

Івано-Франківський національний технічний університет нафти і газу
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

Копей Володимир Богданович

УДК 622.276.054:[004.89+004.94]

ДИСЕРТАЦІЯ

Науково-методологічні основи автоматизованого проектування обладнання
штангової свердловинної насосної установки

05.05.12 – машини нафтової та газової промисловості

Частина 1

Подається на здобуття наукового ступеня доктора технічних наук

Дисертація містить результати власних досліджень. Використання ідей,
результатів і текстів інших авторів мають посилання на відповідне джерело

 В. Б. Копей

Науковий консультант Петрина Юрій Дмитрович, д-р техн. наук, професор

Івано-Франківськ – 2020

Світлій пам'яті

Петрини Юрія Дмитровича

– засновника та лідера наукової
школи "Науково-прикладні основи
розробки технологічних методів
підвищення довговічності
нафтогазового обладнання"

АНОТАЦІЯ

Копей В. Б. Науково-методологічні основи автоматизованого проектування обладнання штангової свердловинної насосної установки. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.05.12 – машини нафтової та газової промисловості. – Івано-Франківський національний технічний університет нафти і газу, МОН України, Івано-Франківськ, 2020.

Дисертація присвячена вирішенню актуальної науково-методологічної проблеми ефективного проектування обладнання комплексно-працевдатної штангової свердловинної насосної установки (ШСНУ).

Метою роботи є удосконалення методології автоматизованого проектування обладнання ШСНУ на основі системного підходу та її використання для комплексного та ефективного дослідження і забезпечення працевдатності ШСНУ.

Об'єкт дослідження – проектування обладнання ШСНУ.

Предмет дослідження – принципи побудови, інтеграції та використання компонентів інформаційної системи (ІС) для автоматизованого проектування обладнання комплексно-працевдатної ШСНУ.

У роботі за допомогою системного підходу та методів комп'ютерного моделювання запропоновано цілісну множину науково-методологічних рішень для розв'язання важливої проблеми ефективного дослідження та проектування обладнання комплексно-працевдатної ШСНУ. На основі цих рішень досліджено роботу обладнання ШСНУ в умовах статичного, циклічного та вібраційного навантаження; отримано залежності показників його працевдатності від конструкційних, технологічних і експлуатаційних параметрів і, на основі цього, подані рекомендації для комплексного забезпечення його працевдатності. Результати досліджень володіють здатністю до простого впровадження.

На основі індуктивних методів самоорганізації моделей розроблено принципи побудови точних і робастних статистичних моделей відмов колон насосних штанг і полірованого штока: моделей густини імовірності відмов як функції від відносної глибини обриву та моделей класу аварійності колони, які враховують множину факторів, пов'язаних з параметрами ШСНУ та її відмови. Обґрунтовано ефективність застосування ансамблів дерев рішень для прогнозування частоти відмов колон насосних штанг за статистичними промисловими даними про відмови.

Розроблено принципи компонентно-орієнтованого моделювання ШСНУ, направлені на моделювання факторів, які розподілені нерівномірно вздовж колони штанг, спрощення створення моделей, їхнього розвитку, використання більшою спільнотою користувачів та інтеграції в ІС. Зокрема розроблено параметричні моделі ШСНУ – модель на основі абстрактних автоматів та мови Python з можливістю моделювання явищ, які важко сформулювати за допомогою диференціальних рівнянь; модель мовою Modelica, яка для моделювання колони штанг використовує стандартні поступальні механічні компоненти. Розроблено алгоритмічні основи та реалізацію програмного каркасу мовою Python для створення компонентно-орієнтованих, подібних на Modelica-моделі, акаузальних моделей ШСНУ та інших динамічних систем без необхідності застосування спеціалізованих мов моделювання. Показані адекватність моделей та приклади їхнього застосування для симуляції склопластикових колон і білярезонансних частот ходів.

На основі відомих САПР і систем моделювання гідродинаміки і механіки деформівного твердого тіла та розроблених автором програмних компонентів запропоновані принципи побудови та використання параметричних геометричних і скінченно-елементних моделей та прикладних САПР проблемних деталей та вузлів ШСНУ, які володіють здатністю простої інтеграції в ІС і можуть бути використані для різностороннього аналізу та оптимізації під час дослідження працездатності та проектування. Зокрема розроблено: гідродинамічні скінченно-елементні моделі (СЕМ) кульового зворотного клапана свердловинного

штангового насоса та протектора насосних штанг; геометричні моделі та СЕМ різьбових з'єднань – муфтового з'єднання насосних штанг, двоопорного з'єднання порожнистих штанг, муфтового з'єднання НКТ, модель різьбових з'єднань з довільними геометричними відхиленнями; СЕМ пресового з'єднання тіла склопластикової насосної штанги зі сталеву головою; геометричні моделі та СЕМ штанг і НКТ з корозійними та втомними дефектами і склопластиковими бандажами. За допомогою розроблених моделей досліджено та уточнено оцінку впливу конструкційних, технологічних і експлуатаційних параметрів свердловинного обладнання ШСНУ на його показники працездатності, зокрема, які характеризують його статичну та втомну міцність, герметичність і вібростійкість. Запропоновано шляхи дослідження і удосконалення обладнання ШСНУ, оптимізації та раціонального вибору значень його параметрів. Зокрема, за критеріями статичної та втомної міцності удосконалено конструкцію з'єднання склопластикового тіла з головою штанги, різьбового з'єднання штанг, з'єднання порожнистих штанг, отримані залежності для вибору оптимального моменту згвинчування різьбових з'єднань штанг та НКТ, виконано аналіз різьбового з'єднання НКТ в умовах гармонічного високочастотного навантажування.

Запропонований підхід до побудови експертних систем з проблем надійності різьбових з'єднань ШСНУ, які описують фактори, що впливають на надійність, і дозволяють логічне виведення нових знань. Підхід оснований на семантичних мережах, логіці предикатів, мові OWL і використанні об'єктно-орієнтованих конструктів мови Python для створення класів, індивідів, атрибутів і відношень онтології. Основні його переваги – проста інтеграція в іншу систему, універсальність і широкі можливості мови Python, легке розширення функціональності системи, зокрема можливість створення власних способів подання та виведення знань.

На основі міждисциплінарного аналізу закономірностей складних систем, розроблено абстрактну модель інформаційної системи для проектування обладнання та підтримки життєвого циклу ШСНУ. Модель шляхом дихотомічного ділення життєвого циклу виробу виділяє класи компонентів ІС

(моделі, результати симуляції, факти бази знань та ін.) та їхню ієрархію, а також володіє основними закономірностями складних систем. На основі цієї моделі розвинуто методологічні основи автоматизованого проектування обладнання та інформаційної підтримки життєвого циклу ШСНУ та запропоновані принципи побудови ІС, яка належить до класу мультиагентних систем, володіє закономірностями складних систем, зокрема здатна до простого свого розширення та інтеграції гетерогенних елементів.

Наукова новизна отриманих результатів полягає в розвитку методології автоматизованого проектування обладнання комплексно-працевдатної ШСНУ, яке опирається на принципи теорії систем та інтегровані в ІС імітаційні моделі обладнання, статистичні моделі відмов і бази знань. Зокрема:

– вперше, на основі міждисциплінарного аналізу закономірностей складних систем, розроблено ізоморфну до складних систем абстрактну модель ІС, яка шляхом дихотомічного ділення життєвого циклу (ЖЦ) виробу виділяє класи функціональних елементів ІС, їхню ієрархію і відношення та є базою для реалізації ефективних ІС;

– набули подальшого розвитку принципи створення ІС, яка відрізняється тим, що належить до класів мультиагентних і гібридних систем, володіє закономірностями складних систем та здатністю до простого розширення та інтеграції гетерогенних компонентів;

– вперше розроблено модель ШСНУ, в якій для спрощення моделювання локальних явищ пружна багатосекційна колона штанг моделюється системою абстрактних автоматів мовою Python;

– вперше розроблено просту для модифікації компонентно-орієнтовану модель ШСНУ мовою Modelica та інтерфейс до неї мовою Python, яка основана на використанні стандартних компонентів "маса", "поступальна пружина з демпфером" і "поступальна сила" для побудови моделі багатосекційної пружної колони штанг;

– набули подальшого розвитку алгоритмічні основи та реалізація програмного каркасу для створення компонентно-орієнтованих акаузальних

моделей динамічних систем, який є гнучким у застосуванні завдяки використанню мови загального призначення Python і опису компонентів різницеви́ми або диференціальними рівняннями засобами пакету SymPy, та розроблено моделі штангових колон на його основі;

– набули подальшого розвитку принципи побудови та використання параметричних моделей та прикладних САПР обладнання ШСНУ, які володіють здатністю до системного дослідження працездатності обладнання, інтеграції в ІС, простого розвитку та базуються на відомих системах CAD/FEA і розроблених автором програмних компонентах;

– досліджено та уточнено вплив конструкційних, технологічних і експлуатаційних параметрів свердловинного обладнання ШСНУ на його показники працездатності, зокрема, які характеризують його статичну та втомну міцність, герметичність та вібростійкість;

– вперше доведено ефективність застосування ансамблів дерев рішень для прогнозування частоти відмов колон насосних штанг за статистичними промисловими даними про відмови.

Практичне значення отриманих результатів полягає в розробленні компонентів ІС та рекомендацій щодо удосконалення обладнання ШСНУ. Зокрема:

– розроблена методика побудови робастних статистичних моделей відмов штангової колони, яка дозволить ефективно ідентифікувати високоаварійні колони і їхні ділянки ще на етапі проектування та виявляти причини відмов;

– розроблено множину параметричних геометричних і скінченно-елементних моделей обладнання ШСНУ, які можуть інтегруватися в ІС: РЗ штанг і НКТ, з'єднань склопластикових штанг, бандажів для ШН і НКТ, протектора, клапана насоса, верстата-качалки (ВК), піделеваторного бурта ШН;

– запропоновано шляхи дослідження та удосконалення обладнання ШСНУ, оптимізації та раціонального вибору значень його параметрів. Зокрема за критеріями статичної та втомної міцності удосконалено конструкцію з'єднання склопластикового тіла з ніпелем, РЗ штанг, з'єднання порожнистих ШН, отримані

залежності для вибору оптимального моменту згвинчування РЗ штанг та НКТ, виконано аналіз РЗ НКТ в умовах гармонічного високочастотного навантажування;

– створено базу знань з проблем надійності РЗ ШСНУ, у якій для подання знань використовується логіка предикатів і мова Python;

– інформаційна система підтримки життєвого циклу ШСНУ, як відкрите програмне забезпечення, впроваджена в НГВУ "Долинанафтогаз" ПАТ "Укрнафта" з метою підвищення якості експлуатації ШСНУ та комплексного забезпечення її працездатності;

– основні результати роботи впроваджено у навчальний процес Івано-Франківського національного технічного університету нафти і газу для підготовки фахівців освітньо-кваліфікаційного рівня магістр за спеціальністю 131 "Прикладна механіка".

Ключові слова: штангова свердловинна насосна установка, працездатність, насосні штанги, насосно-компресорні труби, різьбове з'єднання, статистична модель, імітаційна модель, компонентно-орієнтоване моделювання, метод скінченних елементів, напружено-деформований стан, втомна міцність, управління життєвим циклом виробу, автоматизоване проектування, інформаційна система, мова програмування Python.

ANNOTATION

Kopei V. B. Scientific and methodological bases of computer-aided design of equipment for a sucker rod pumping unit. – Qualification scientific work with the manuscript copyright.

The dissertation for a doctor technical sciences degree in speciality 05.05.12 – machines of oil and gas industry. – Ivano-Frankivsk National Technical University of Oil and Gas, Ministry of Education and Science of Ukraine, Ivano-Frankivsk, 2020.

The dissertation is devoted to solving the current scientific and methodological problem of the effective design of the equipment of a sucker-rod pumping unit (SRPU)

with integrated operability.

The **purpose a research** is to improve, on the basis of a systems approach, the methodology of computer-aided design of SRPU equipment and its use for comprehensive and effective research and ensuring the operability of SRPU.

The object of research is the SRPU equipment design.

The subject of research is principles of creation, integration and use of information system (IS) components for computer-aided design of equipment of SRPU with integrated operability.

Using a systems approach and computer modeling methods, an integral set of scientific and methodological solutions, to solve the important problem of effective research and design of equipment of SRPU with integrated operability, has been proposed. On the basis of these solutions, the operation of the SRPU equipment in the conditions of static, cyclic and vibrational loading was investigated; dependences of its operability indicators on design, technological and operational parameters are obtained, and based on this, recommendations for the integrated ensuring of its operability are given. The results of the research could be easily implemented.

On the basis of inductive methods of self-organization of models, principles for constructing accurate and robust statistical models of failures of sucker rods and polished rods have been developed: models of the probability density of failures as a function of the relative depth of breakage and models of the sucker rod string accident rate, taking into account many factors associated with the parameters of the SRPU and its failures. The efficiency of application of decision trees ensembles for predicting the failure rate of sucker rod strings by oilfield failure statistics has been substantiated.

The principles of component-oriented modeling of SRPU have been developed. They aimed at: modeling factors that are distributed non-uniformly along the rod string; simplifying the creation of models and their development; using by a large community of users; integrating into the IS. In particular, parametric models of SRPU have been developed: a model based on abstract automata and the Python language with the ability to simulate phenomena that are difficult to formulate with differential equations; a Modelica-model that uses standard translational mechanical components to model the

rod string. Algorithmic foundations and implementation of a software framework in the Python language for creating component-oriented, like Modelica-models, acausal models of SRPU and other dynamic systems without the need for specialized modeling languages have been developed. The adequacy of the models and examples of their application for simulating fiberglass strings and near-resonance frequencies of strokes are shown.

On the basis of well-known free and commercial CAD-systems and modeling systems for hydrodynamics and solid mechanics and software components developed by the author, principles for the construction and the use of parametric geometric and finite element models and applied CAD-systems of problem parts and subassemblies of SRPU, which have the ability to easily integrate into an information system and can be used for comprehensive analysis and optimization in operability research and design, are proposed. In particular, the following have been developed: hydrodynamic finite element models (FE-models) of a ball check valve of a downhole sucker rod pump and a sucker rod protector; geometric models and FE-models of threaded connections – coupling connection of sucker rods, double-shoulder connection of hollow sucker rods, tubing coupling, models of threaded connections with arbitrary geometric deviations; FE-model of the press connection of the fiberglass sucker rod body with a steel head; geometric models and FE-models of rods and tubing with corrosion and fatigue defects and fiberglass bandages. With the help of the developed models, the assessment of the influence of the design, technological and operational parameters of the SRPU downhole equipment on its operability indicators, in particular, characterizing its static and fatigue strength, leaktightness and vibration resistance, was investigated and refined. The ways of research and improvement of the SRPU equipment, optimization and rational choice of the values of its parameters are proposed. In particular, according to the criteria of static and fatigue strength, the design of the connection of the fiberglass rod body with the rod head, threaded connections of rods, connections of hollow rods was improved; dependencies for choosing the optimal make-up torque for rods and tubing connections were obtained; an analysis of threaded connections of tubing under conditions of harmonic high-frequency loading was performed.

An approach to the construction of expert systems on the problems of the reliability of threaded connections of SRPU, which describe the factors, affecting reliability, and allow a logical inference of new knowledge, is proposed. The approach is based on semantic networks, predicate logic, OWL language and the use of object-oriented constructs of the Python language to create classes, individuals, attributes, and relations of an ontology. Its main advantages are simple integration into another system, the flexibility and wide capabilities of the Python language, easy expansion of the system's functionality, in particular, the ability to create your own ways of knowledge representation and reasoning.

Based on the interdisciplinary analysis of regularities of complex systems, an abstract model of the information system for equipment design and SRPU life cycle support has been developed. The model by dichotomous division of the product lifecycle identifies the classes of IS components (models, simulation results, knowledge base facts, etc.) and their hierarchy, and also has the basic regularities of complex systems. On the basis of this model, the methodological foundations of computer-aided design of equipment and information support of the SRPU lifecycle, as well as the principles of building an IS, which belongs to the class of multi-agent systems, has the regularities of complex systems, in particular, capable of its easy expansion and integration of heterogeneous elements, have been developed.

The scientific novelty of the results consist in the development of a methodology for the computer-aided design of equipment for a SRPU with integrated operability, which is based on the principles of systems theory and integrated in the IS simulation models of equipment, statistical models of failures and a knowledge base. In particular:

- for the first time, on the basis of an interdisciplinary analysis of regularities of complex systems, an isomorphic to complex systems abstract model of IS, which, by dichotomous division of the product lifecycle, identifies classes of functional elements of IS, their hierarchy and relationships and is the basis for the implementation of effective IS, has been developed;

– the principles of creation of IS, which differs in that it belongs to the classes of multi-agent and hybrid systems, has regularities of complex systems and the ability to easily expand itself and integrate heterogeneous components, were further developed;

– for the first time, a model of a SRPU, in which, to simplify the modeling of local phenomena, an elastic multisectional string of rods is modeled by a system of abstract automata in Python, was developed;

– for the first time, an easy-to-modify component-oriented model of the SRPU in Modelica and an interface to it in Python, which is based on the use of standard components "mass", "translational spring with damper" and "translational force" to build a model of a multi-section elastic rod string, was developed;

– algorithmic foundations and implementation of a software framework for creating component-oriented acausal models of dynamic systems, which is flexible in application due to the use of the general-purpose Python language and the description of components by difference equations or differential equations using the SymPy package, were further developed, and also models of rod strings, based on it, were developed;

– the principles of constructing and using parametric models and applied CAD-systems of SRPU equipment, which have the ability to system research of equipment operability, integration into IS, simple development and based on well-known CAD/FEA systems and software components developed by the author, have been further developed;

– the influence of the design, technological and operational parameters of the SRPU downhole equipment on its operability indicators, in particular, characterizing its static and fatigue strength, leaktightness and vibration resistance, was investigated and refined;

– for the first time, the effectiveness of using decision trees ensembles for predicting the failure rate of sucker rod strings, based on oilfield statistical data on failures, has been proven.

The practical significance of the obtained results consists in the development of IS components and recommendations for improving the equipment of SRPU. In particular:

- a technique has been developed for constructing robust statistical models of sucker rod string failures, which will make it possible to effectively identify high-accident strings and their sections even at the design stage and to reveal the causes of failures;

- a set of parametric geometric and finite element models of SRPU equipment, which can be integrated into IS, have been developed: threaded connections of rods and tubing, connections of fiberglass rods, bandages for rods and tubing, protector, pump valve, pumping unit, rod shoulder for elevator;

- the ways of research and improvement of SRPU equipment, optimization and rational choice of the values of its parameters are proposed. In particular, according to the criteria of static and fatigue strength, the designs of the connection of the fiberglass rod body with the head, the threaded connection of the rods, the connections of the hollow rods were improved, the dependences for choosing the optimal make-up moment of the threaded connections of the rods and tubing were obtained, the analysis of the threaded connection of tubing under conditions of harmonic high-frequency load was performed;

- a knowledge base on the problems of the reliability of threaded connections of SRPU, in which predicate logic and the Python language are used for knowledge representation, has been created;

- the information system for the SRPU lifecycle support, as open source software, was introduced at "Dolynanaftogaz" Oil and Gas Production Department in order to improve the quality of the SRPU's operation and ensure its integrated operability;

- the main results of the work were introduced into the educational process of Ivano-Frankivsk National Technical University of Oil and Gas for the training of specialists of Master's educational and qualification level in specialty 131 "Applied Mechanics".

Keywords: sucker rod pumping unit, operability, sucker rods, tubing, threaded connection, statistical model, simulation model, component-oriented modeling, finite element method, stress-strain state, fatigue strength, product lifecycle management, computer aided design, information system, Python programming language.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Праці, в яких опубліковані основні наукові результати дисертації

1. Experimental study of the reinforcement of damaged steel pipe by composite bandage / Kopey B., Rozgonjuk V., **Kopey V.**, Maksymuk O., Scherbina N., Nayda A. // *Wiertnictwo Nafta Gaz*. 2004. r.21/1. P. 125-134.
2. Finite-element analysis of the tubing thread / Kopey B., **Kopey V.**, Bebnarz S., Savula S. // *Wiertnictwo Nafta Gaz*. 2006. r.23/2. P. 681-685.
3. Оптимізація товщини композитних бандажів при ремонті трубопроводів з дефектами / Копей Б. В., **Копей В. Б.**, Максимук О. В., Щербина Н. М., Найда А. М. // *Науковий вісник Івано-Франківського національного технічного університету нафти і газу*. 2007. № 2(16). С. 101-107.
4. Копей Б. В., Зінченко Ю. С., **Копей В. Б.** Аналіз поломок насосних штанг в промислових умовах // *Науковий вісник Івано-Франківського національного технічного університету нафти і газу*. 2008. № 2(18). С. 49-56.
5. **Копей В. Б.** Аналіз способів підвищення ресурсу муфтового різьбового з'єднання насосних штанг // *Розвідка та розробка нафтових і газових родовищ*. 2008. № 4(29). С. 66-72.
6. Копей Б. В., Кузьмін О. О., **Копей В. Б.** Механічні методи зняття відкладень парафіну та асфальто-смолистих речовин з поверхні свердловинного обладнання // *Нафтогазова енергетика*. 2008. № 3(8). С. 10-14.
7. Сучасні методи боротьби з корозією глибинного обладнання ШСНУ / Копей Б. В., Онищук О. О., Онищук С. Ю., **Копей В. Б.** // *Нафтогазова енергетика*. 2008. № 2(7). С. 13-16.
8. **Копей В. Б.** Застосування системи CAD/FEA для розрахунку і оптимізації різьбових з'єднань нафтогазового обладнання // *Розвідка та розробка нафтових і газових родовищ*. 2009. № 3(32). С. 43-49.
9. Копей Б. В., Михайлюк В. В., **Копей В. Б.** Моделювання різьб насосних штанг методом скінченних елементів // *Науковий вісник Івано-Франківського*

національного технічного університету нафти і газу. 2009. № 2(20). С. 61-67.

10. Копей Б. В., Кузьмін О. О., **Копей В. Б.** Розроблення з'єднань склопластикових порожнистих насосних штанг та визначення навантажень на них // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2010. № 1(23). С. 77-83.

11. **Копей В. Б.** Розробляння та аналіз параметричних скінченно-елементних моделей різьбових з'єднань в Abaqus® // Нафтогазова енергетика. 2010. № 1(12). С. 31-36.

12. **Копей В. Б.** Скінченно-елементний аналіз та оптимізація різьбових з'єднань // Вісник Національного технічного університету України "Київський політехнічний інститут". Серія машинобудування. 2010. № 58. С. 42-47.

13. **Копей В. Б.** Дослідження впливу геометричних параметрів муфтового різьбового з'єднання насосних штанг на напруження у впадинах різьби ніпеля // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2010. № 2(24). С. 81-85.

14. **Копей В. Б.**, Петрина Ю. Д. Принципи розробки бази знань з проблем надійності і довговічності різьбових з'єднань // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2010. № 4(26). С. 66-69.

15. **Копей В. Б.** Автоматизоване проектування з'єднання тіла склопластикової насосної штанги зі сталеву головою // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2011. №5. С. 142-147.

16. Використання явища резонансу для комплектування колони насосних штанг / Олійник А. П., Копей Б. В., Зінченко Ю. С., **Копей В. Б.** // Розвідка та розробка нафтових і газових родовищ. 2011. №1 (38). С. 69-75.

17. **Копей В. Б.**, Палійчук І. І. Застосування мови програмування Python для побудови баз знань з проблем надійності і довговічності штангових свердловинних насосних установок // Нафтогазова енергетика. 2011. №2(15). С. 12-18.

18. **Копей В. Б.** Імітаційна модель свердловинної штангової насосної установки на основі абстрактних автоматів // Розвідка та розробка нафтових і газових родовищ. 2017. №3(64). С. 40-49.

19. **Копей В. Б.,** Копей Б. В., Кузьмін О. О. Принципи побудови моделі свердловинної штангової насосної установки для середовища Maplesoft MapleSim 7 // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2017. №2(43). С. 42-52.

20. **Копей В. Б.** Абстрактна модель інформаційної системи підтримки життєвого циклу виробу // Прикарпатський вісник НТШ. Число. 2017. №2(38). С. 71-96.

21. **Kopei V. B.,** Onysko O. R., Panchuk V. G. Computerized system based on FreeCAD for geometric simulation of the oil and gas equipment thread turning // IOP Conf. Ser.: Mater. Sci. Eng. 2019. 477:012032. DOI: 10.1088/1757-899X/477/1/012032. (*Scopus*)

22. **Kopei V. B.,** Onysko O. R., Panchuk V. G. Component-oriented acausal modeling of the dynamical systems in Python language on the example of the model of the sucker rod string // PeerJ Computer Science. 2019. 5:e227. DOI: 10.7717/peerj-cs.227. (*Scopus, Q1*)

23. **Копей В. Б.,** Онисько О. Р., Жигуц Ю. Ю. Обґрунтування застосування двоопорних різьбових з'єднань пустотілих насосних штанг // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2019. №1(46). С. 7-15. DOI: 10.31471/1993-9965-2019-1(46)-7-15.

24. **Kopei V.,** Onysko O., Panchuk V. The application of the uncorrected tool with a negative rake angle for tapered thread turning // Ivanov V. et al. (eds) Advances in Design, Simulation and Manufacturing II. DSMIE 2019. Lecture Notes in Mechanical Engineering. Cham : Springer, 2020. P. 149-158. DOI: 10.1007/978-3-030-22365-6_15. (*Scopus*)

25. **Kopei V. B.,** Onysko O. R., Panchuk V. G. Principles of development of product lifecycle management system for threaded connections based on the Python

programming language // J. Phys.: Conf. Ser. 2020. 1426:12033. DOI: 10.1088/1742-6596/1426/1/012033. (*Scopus*)

26. Onysko O. R., **Копей В. В.**, Panchuk V. G. Theoretical investigation of the tapered thread joint surface contact pressure in the dependence on the profile and the geometric parameters of the threading turning tool // IOP Conf. Ser.: Mater. Sci. Eng. 2020. 749:012007. DOI: 10.1088/1757-899X/749/1/012007. (*Scopus*)

Праці, які засвідчують апробацію матеріалів дисертації

27. **Копей В. В.** Аналіз способів підвищення ресурсу муфтового різьбового з'єднання насосних штанг // Анотації Міжнародної науково-практичної конференції молодих вчених "Техніка і прогресивні технології у нафтогазовій інженерії", м. Івано-Франківськ, 16-20 вересня 2008 р. Івано-Франківськ : Факел, 2008. С. 55. (*форма участі – очна*)

28. Скінчено-елементний аналіз насосних штанг з зарізьбовими канавками / Копей В. В., **Копей В. В.**, Петрина Ю. Д, Михайлюк В. В. // Анотації Міжнародної науково-технічної конференції "Нафтогазова енергетика: проблеми і перспективи". м. Івано-Франківськ, 20-23 жовтня 2009 р. Івано-Франківськ : ІФНТУНГ, 2009. С. 67. (*форма участі – очна*)

29. **Копей В. В.**, Панчук А. Г. Дослідження залежності напружень в муфтовому різьбовому з'єднанні насосних штанг від характеристик матеріалів деталей з'єднання // Сучасні технології в промисловому виробництві : матеріали Всеукраїнської міжвузівської науково-технічної конференції, м. Суми, 19-23 квітня 2010 р. Ч. II. Суми : Вид-во СумДУ, 2010. С. 122-123. (*форма участі – заочна*)

30. **Копей В. В.** Скінченно-елементний аналіз та оптимізація різьбових з'єднань // Тези доповідей XI Міжнародної науково-технічної конференції "Прогресивна техніка і технологія - 2010", м. Київ, 18-21 травня 2010 р. Київ : НТУУ "КПІ", 2010. С. 99. (*форма участі – очна*)

31. **Копей В.** Скінченно-елементне моделювання та оптимізація параметрів муфтового різьбового з'єднання насосних штанг за критерієм втомної міцності //

Комп'ютерні технології: наука і освіта : тези доповідей V Всеукраїнської науково-практичної конференції, м. Івано-Франківськ, 29.09-3.10.2010 р. Київ : Університет "Україна", 2010. С. 109-112. *(форма участі – очна)*

32. **Копей В. Б.**, Панчук А. Г. Оптимізація параметрів з'єднання тіла склопластикової насосної штанги зі сталевую головкою // Инновационные технологии в машиностроении : материалы Международной научно-практической конференции, г. Запорожье, 17-21 мая 2011 г. Том 2. Запорожье : Запорожская торгово-промышленная палата, 2011. С. 58-60. *(форма участі – заочна)*

33. **Копей В. Б.**, Венгрынюк Т. П. Моделирование дефектов труб в SolidWorks® // Надежность и безопасность магистрального трубопроводного транспорта : материалы VII Междунар. науч.-техн. конф., Новополоцк, 22 – 25 ноября 2011 г. / под общ. ред. д-ра техн. наук, проф. В. К. Липского. Новополоцк : Полоц. гос. ун-т, 2011. С. 250-251. *(форма участі – заочна)*

34. **Копей В. Б.**, Петрина Ю. Д., Венгрынюк Т. П. Конечно-элементное моделирование ремонта труб с дефектами стеклопластиковыми бандажами в SolidWorks® // Надежность и безопасность магистрального трубопроводного транспорта : материалы VII Междунар. науч.-техн. конф., Новополоцк, 22 – 25 ноября 2011 г. / под общ. ред. д-ра техн. наук, проф. В. К. Липского. Новополоцк : Полоц. гос. ун-т, 2011. С. 248-250. *(форма участі – заочна)*

35. **Копей В. Б.** Створення експертної системи з проблем надійності і довговічності різьбових з'єднань в редакторі онтологій Protégé-OWL 3.5 // Збірка наукових праць за матеріалами Міжнародної науково-практичної конференції "Наукові дослідження сучасності. Випуск 4", м. Київ, 30 травня 2012 р. Частина 1. Київ : НАІРІ, 2012. С. 99-101. *(форма участі – заочна)*

36. **Копей В. Б.** Створення експертної системи з проблем надійності і довговічності різьбових з'єднань мовою Python // Збірка наукових праць за матеріалами Міжнародної наукової конференції "Наука - XXI століття. Випуск 2", м. Київ, 27 червня 2012 р. Київ : НАІРІ, 2012. С. 117-122. *(форма участі – заочна)*

37. **Копей В. Б.** Скінченно-елементне моделювання та аналіз втомної міцності насосних штанг в зоні скруглення між піделеваторним буртом і тілом штанги //

Тези доповідей Міжнародної науково-практичної конференції молодих учених та студентів "Техніка і прогресивні технології у нафтогазовій інженерії - 2012", м. Івано-Франківськ, 5-7 листопада 2012 р. Івано-Франківськ : ІФНТУНГ, 2012. С. 123-126. *(форма участі – очна)*

38. **Копей В. Б.** Обґрунтування доцільності збільшення довжини розвантажувальної канавки ніпеля насосної штанги // Технологічний аудит та резерви виробництва (Спецвипуск. Матеріали науково-практичної конференції "Наукові підсумки 2012р.", м. Харків, 2012 р.). 2012. № 6/2 (8). С. 7-8. *(форма участі – заочна)*

39. **Копей В. Б.** Моделювання клапана свердловинного штангового насоса методом обчислювальної гідродинаміки в Abaqus/CAE® // Матеріали Міжнародної науково-технічної конференції "Математичне моделювання прикладних задач математики, фізики, механіки ММАР-2013", м. Харків, 5 – 25 травня 2013 р. Харків : ХНАДУ, 2013. URL: <http://files.khadi.kharkov.ua/images/Fizika.pdf> (дата звернення: 29.02.2016). *(форма участі – заочна)*

40. **Kopey V.** Development of model of sucker-rod pumping system by using Maplesim™ software // International scientific and technical conference "Oil and Gas Power Engineering 2013" : abstracts, Ivano-Frankivsk, October 7-11, 2013. Ivano-Frankivsk : IFNTUOG, 2013. P. 116-118. *(форма участі – очна)*

41. **Копей В. Б.** Принципи побудови тривимірної параметричної моделі верстата-гойдалки в SolidWorks® 2012 // Всеукраїнський науково-практичний семінар "Графічна освіта у ВНЗ: стан та перспективи" : збірник тез доповідей, м. Івано-Франківськ, 19-20 вересня 2013 р. Івано-Франківськ : ІФНТУНГ, 2013. С. 87-89. *(форма участі – очна)*

42. **Копей В. Б.** Використання вільного програмного забезпечення для розробки системи скінченно-елементного аналізу різьбових з'єднань нафтогазового обладнання // Матеріали Міжнародної науково-технічної конференції "Машини, обладнання і матеріали для нарощування вітчизняного видобутку та диверсифікації постачання нафти і газу" ПМ – 2016, м. Івано-

Франківськ, 16-20 травня 2016 р. Івано-Франківськ : ІФНТУНГ, 2016. С. 277-280.
(форма участі – очна)

43. **Копей В. Б.** Моделювання гармонічного осьового навантажування пресового з'єднання склопластикової насосної штанги // Соціально-економічний розвиток в умовах глобалізації : матеріали XLIX Міжнародної науково-практичної конференції, м. Чернівці, 29-30 листопада 2016 р. Т. 1. Київ : Науково-видавничий центр "Лабораторія думки", 2016. С. 7-10. (форма участі – заочна)

44. **Копей В. Б.** Моделювання втомної міцності ніпеля склопластикової насосної штанги // Матеріали II-ї Міжнародної науково-практичної конференції "Теоретичні та практичні аспекти розвитку науки", м. Київ, 29 – 30 листопада 2016 р. Ч. 2. Київ : МЦНД, 2016. С. 22-24. (форма участі – заочна)

45. **Копей В. Б.** Компонентно-орієнтоване моделювання кінематики механізмів мовою Python на прикладі механізму верстата-гойдалки // Тези доповідей Міжнародної науково-технічної конференції "Нафтогазова енергетика-2017", м. Івано-Франківськ, 15-19 травня 2017 р. Івано-Франківськ : ІФНТУНГ, 2017. С. 362-364. (форма участі – очна)

46. **Копей В. Б.,** Копей Б. В., Кузьмін О. О. Принципи побудови моделі свердловинної штангової насосної установки для середовища Maplesoft MapleSim 7 // Тези доповідей Міжнародної науково-технічної конференції "Нафтогазова енергетика-2017", м. Івано-Франківськ, 15-19 травня 2017 р. Івано-Франківськ : ІФНТУНГ, 2017. С. 364-365. (форма участі – очна)

47. **Копей В. Б.** Імітаційна модель свердловинної штангової насосної установки на основі абстрактних автоматів // Тези доповідей Міжнародної науково-технічної конференції "Нафтогазова енергетика-2017", м. Івано-Франківськ, 15-19 травня 2017 р. Івано-Франківськ : ІФНТУНГ, 2017. С. 365-366. (форма участі – очна)

48. **Копей В. Б.,** Воробець М. І. Система автоматизованого проектування металополімерних з'єднань на основі вільного програмного забезпечення // Машинобудування очима молодих: прогресивні ідеї – наука – виробництво (МОМ – 2017) : матеріали тез доповідей XVII Міжнародної науково-практичної

конференції, м. Чернігів, 01 – 03 листопада 2017 р. Чернігів : ЧНТУ, 2017. С. 31-33. *(форма участі – заочна)*

49. **Копей V.**, Panchuk V., Onysko O. Component-oriented modelling of dynamical systems in Python language on the example of the model of the sucker rod string // International conference Innovative Ideas in Science 2017 : book of abstracts, Banja Luka, 02 - 03 November 2017. Banja Luka : University of Business Engineering and Management, 2017. P. 33. *(форма участі – очна)*

50. **Копей В. Б.** Статистичні моделі відмов колон насосних штанг // Матеріали II Міжнародної науково-технічної конференції "Машини, обладнання і матеріали для нарощування вітчизняного видобутку нафти і газу PGE – 2018", м. Івано-Франківськ, 24-27 квітня 2018 р. Івано-Франківськ : ІФНТУНГ, 2018. С. 310-313. *(форма участі – очна)*

51. **Копей В. Б.** Прогнозування частоти відмов колон насосних штанг за допомогою ансамблів дерев рішень // Комплексне забезпечення якості технологічних процесів та систем (КЗЯТПС – 2018) : матеріали тез доповідей VIII Міжнародної науково-практичної конференції, м. Чернігів, 10–12 травня 2018 р. : у 2-х т. / Чернігівський національний технологічний університет [та ін.]; відп. за вип.: Єрошенко Андрій Михайлович [та ін.]. Т. 2. Чернігів : ЧНТУ, 2018. С. 198-200. *(форма участі – заочна)*

52. **Копей V.**, Onysko O., Panchuk V. Computerized system based on FreeCAD for geometric simulation of thread turning of oil and gas pipes // 6th International Conference of Applied Science, May 9-11, 2018, Banja Luka, Bosnia and Herzegovina : book of abstracts. Banja Luka : University of Banja Luka, 2018. P. 108. *(форма участі – очна)*

53. **Копей В. Б.** Алгоритм інтелектуальної системи на основі міждисциплінарних досліджень загальносистемних закономірностей // Комп'ютерне моделювання та оптимізація складних систем (КМОСС-2018) : матеріали IV Міжнародної науково-технічної конференції, м. Дніпро, 1-2 листопада 2018 р. / Міністерство освіти і науки України, Державний вищий навчальний заклад "Український державний хіміко-технологічний університет". Дніпро : Баланс-клуб, 2018. С. 246-248. *(форма участі – заочна)*

54. **Копей В. Б.**, Онисько О. Р., Жигуц Ю. Ю. Обґрунтування застосування двоопорних муфтових різьбових з'єднань пустотілих насосних штанг // Матеріали доповідей VIII Міжнародної науково-технічної конференції "Прогресивні технології у машинобудуванні РТМЕ-2019", 4-8 лютого 2019 р., м. Івано-Франківськ-Яремче. Івано-Франківськ : ІФНТУНГ, 2019. С. 146-149. (*форма участі – очна*)

55. **Kopei V.**, Onysko O., Panchuk V. Principles of the product lifecycle management system development for threaded connections based on the Python programming language // International Conference of Applied Science ICAS 2019 : book of abstracts, May 9-11, 2019, Hunedoara, Romania. Timisoara : Editura EUROBIT, 2019. P. 42. (*форма участі – очна*)

56. **Kopei V. B.**, Kopei B. V. Harmonic axial loading analysis of the tubing threaded connection // Modern achievements of science and education : proceedings of XIV International scientific conference, September 26 - October 3, 2019, Netanya, Israel. Khmelnytskyi : KhNU, 2019. P. 29-37. (*форма участі – очна*)

Праці, які додатково відображають наукові результати дисертації

57. Комп'ютерна програма "Програма для побудови баз знань з проблем надійності і довговічності штангових свердловинних насосних установок" ("SuckerRodPumpingUnitKB") : свідоцтво про реєстрацію авторського права на твір № 42157 / **Копей Володимир Богданович**. Дата реєстрації 08.02.2012 // Каталог державної реєстрації. Вип.16/2012. С. 42.

58. Комп'ютерна програма "Модуль для компонентно-орієнтованого моделювання динамічних систем та приклади його застосування для побудови моделей колони насосних штанг" : свідоцтво про реєстрацію авторського права на твір № 73098 / **Копей Володимир Богданович**. Дата реєстрації 25.07.17 // Бюл. "Авторське право і суміжні права" № 46/2017. С. 154.

ЗМІСТ

с.

Частина 1

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	29
ВСТУП.....	31
РОЗДІЛ 1 СТАН ПРОБЛЕМИ І ЗАДАЧІ ДОСЛІДЖЕНЬ.....	39
1.1 Принципи функціонування ШСНУ.....	39
1.2 Аналіз умов роботи та відмов свердловинного обладнання ШСНУ	40
1.2.1 Колона насосних штанг.....	40
1.2.2 Насосно-компресорні труби	49
1.2.3 Свердловинні штангові насоси.....	53
1.3 Огляд ремонту та зміцнення труб і штанг склопластиковими бандажами.....	55
1.4 Проблеми статистичних моделей відмов колон насосних штанг	59
1.5 Проблеми імітаційних моделей ШСНУ.....	61
1.5.1 Огляд математичних моделей ШСНУ	61
1.5.2 Огляд методів компонентно-орієнтованого моделювання	62
1.5.3 Можливості імітаційного моделювання на основі абстрактних автоматів	64
1.6 Огляд можливостей систем CAD/FEA для автоматизованого проектування обладнання ШСНУ.....	66
1.6.1 Інтеграція CAD і FEA для проектування обладнання ШСНУ.....	66
1.6.2 Аналіз можливостей систем CAD/FEA для моделювання різьбових з'єднань	69
1.7 Огляд методів обчислення циклічної довговічності за результатами моделювання МСЕ	72
1.8 Проблеми інформаційної системи підтримки життєвого циклу ШСНУ	78
1.8.1 Ізоморфізм закономірностей складних систем.....	78
1.8.2 Обґрунтування архітектури ІС, мови системної інтеграції та методів подання знань	87
1.9 Вибір напрямку досліджень	92
РОЗДІЛ 2 СТАТИСТИЧНІ МОДЕЛІ ВІДМОВ КОЛОН НАСОСНИХ ШТАНГ... ..	95

2.1 Залежності частоти відмов колони від відносної глибини обриву	95
2.2 Залежності частоти відмов колони від відносної глибини обриву та діаметра плунжера свердловинного насоса.....	107
2.3 Багатовимірні поліноміальні моделі частоти відмов.....	112
2.4 Моделі частоти відмов на основі ансамблів дерев рішень	117
2.5 Висновки до розділу.....	120
РОЗДІЛ 3 КОМПОНЕНТНО-ОРІЄНТОВАНЕ МОДЕЛЮВАННЯ ШСНУ	122
3.1 Імітаційна модель ШСНУ на основі абстрактних автоматів	122
3.2 Імітаційна модель ШСНУ мовою Modelica	136
3.2.1 Принципи побудови моделі ШСНУ мовою Modelica	136
3.2.2 Застосування і верифікація моделей	153
3.2.3 Обчислення циклічних навантажень, що діють на штанги і НКТ в умовах парафіноутворення.....	163
3.3 Автоматизована симуляція гідродинаміки клапана свердловинного штангового насоса для різної висоти підйому кульки	165
3.4 Компонентно-орієнтоване акаузальне моделювання динамічних систем мовою Python на прикладі моделі колони насосних штанг	169
3.4.1 Опис моделі мовою Modelica.....	169
3.4.2 Опис принципів моделювання в Python	172
3.4.3 Симуляція вільних коливань штангової колони.....	177
3.4.4 Симуляція процесу відкачування.....	180
3.4.5 Аналіз результатів симуляцій.....	183
3.5 Компонентно-орієнтоване моделювання кінематики механізмів мовою Python на прикладі механізму верстата-качалки	185
3.6 Принципи побудови тривимірної параметричної моделі верстата-качалки в SOLIDWORKS	187
3.7 Висновки до розділу.....	190
РОЗДІЛ 4 ПАРАМЕТРИЧНІ СКІНЧЕННО-ЕЛЕМЕНТНІ МОДЕЛІ РІЗЬБОВИХ З'ЄДНАНЬ ШСНУ	193

4.1 Принципи побудови параметричних скінченно-елементних моделей різьбових з'єднань в Abaqus/CAE	193
4.2 Принципи побудови параметричних скінченно-елементних моделей різьбових з'єднань на основі вільного програмного забезпечення	200
4.3 Геометричні моделі різьбових з'єднань з відхиленнями на основі FreeCAD	204
4.4 Опис методики оптимізації різьбових з'єднань	208
4.5 Удосконалення різьбового з'єднання насосних штанг за критерієм статичної міцності	211
4.6 Удосконалення різьбового з'єднання насосних штанг за критеріями втомної міцності	219
4.7 Скінченно-елементний аналіз різьбових з'єднань ШН з відхиленнями	225
4.7.1 Відхилення кута профілю різьби	225
4.7.2 Відхилення кроку різьби	226
4.7.3 Відхилення зовнішнього діаметра різьби ніпеля штанги	229
4.8 Обґрунтування застосування двоопорних різьбових з'єднань порожнистих насосних штанг	232
4.9 Аналіз з'єднання насосно-компресорних труб за критерієм герметичності ..	241
4.10 Обґрунтування застосування різьб НКТ, які виготовлені некоригованими різцями з від'ємним переднім кутом	247
4.11 Моделювання гармонічного осьового навантажування різьбового з'єднання насосно-компресорних труб	249
4.12 Висновки до розділу	259
РОЗДІЛ 5 СКІНЧЕННО-ЕЛЕМЕНТНІ МОДЕЛІ ЕЛЕМЕНТІВ КОЛОНИ ШТАНГ ТА НКТ	263
5.1 Моделі пресового з'єднання тіла склопластикової насосної штанги зі сталеву головою	263
5.1.1 Принципи побудови моделі на основі Abaqus/CAE	263
5.1.2 Принципи побудови моделі на основі вільного програмного забезпечення	269

5.1.3 Залежності міцності з'єднання тіла склопластикової насосної штанги зі сталеву головою від його параметрів.....	272
5.1.4 Моделювання гармонічного осьового навантажування пресового з'єднання склопластикової насосної штанги	278
5.1.5 Моделювання втомної міцності ніпеля склопластикової насосної штанги	283
5.2 Моделі штанг і НКТ з дефектами та склопластиковими бандажами	286
5.2.1 Геометричні моделі дефектів штанг і НКТ	286
5.2.2 Вплив склопластикового бандажа на напруження в НКТ	289
5.2.3 Оптимізація конструкції фаски склопластикового бандажа	296
5.2.4 Прогнозування ресурсу НКТ з тріщиною і бандажем за допомогою коефіцієнта інтенсивності напружень	303
5.2.5 Вплив радіуса і глибини сферичного дефекту на напруження в штанзі... ..	308
5.2.6 Вплив діаметра і довжини бандажа на напруження в штанзі зі сферичним дефектом	314
5.2.7 Прогнозування ресурсу штанги з тріщиною і бандажем за допомогою коефіцієнта інтенсивності напружень	315
5.3 Скінченно-елементне моделювання та аналіз втомної міцності насосних штанг в зоні скруглення між піделеваторним буртом і тілом штанги	319
5.4 Гідродинамічна модель протектора для насосних штанг	324
5.5 Висновки до розділу.....	335
РОЗДІЛ 6 ПРИНЦИПИ ПОБУДОВИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПРОЕКТУВАННЯ І ПІДТРИМКИ ЖИТТЄВОГО ЦИКЛУ ОБЛАДНАННЯ ШСНУ	339
6.1 Абстрактна модель інформаційної системи	339
6.1.1 Модель інформаційної системи та її відповідність загальносистемним закономірностям.....	339
6.1.2 Приклад класифікації елементів інформаційної системи.....	353
6.1.3 Алгоритм інформаційної системи на основі загальносистемних закономірностей	356

6.2 Принципи створення баз знань з проблем надійності обладнання ШСНУ	358
6.2.1 Використання мови OWL	358
6.2.2 Використання мови Python	364
6.3 Принципи реалізації багатоагентної інформаційної системи для проектування і підтримки життєвого циклу обладнання ШСНУ	372
6.4 Висновки до розділу	381
ЗАГАЛЬНІ ВИСНОВКИ	383
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	387
Частина 2	
ДОДАТОК А Список публікацій здобувача	432
ДОДАТОК Б Статистичні моделі відмов штангових колон.....	441
ДОДАТОК В Модель ШСНУ на основі абстрактних автоматів.....	451
ДОДАТОК Г Модель ШСНУ мовою Modelica	473
ДОДАТОК Д Графіки циклічних навантажень на колони ШН і НКТ	485
ДОДАТОК Е Пакет для моделювання кулькового клапана методами обчислювальної гідродинаміки.....	487
ДОДАТОК Є ruscodyn – пакет для компонентно-орієнтованого акаузального моделювання мовою Python	490
ДОДАТОК Ж Компонентно-орієнтована модель верстата-качалки для симуляції кінематики.....	509
ДОДАТОК З Python-класи для перебудови параметричної моделі верстата-качалки у SOLIDWORKS	512
ДОДАТОК И Моделі P3 для Abaqus/CAE.....	523
ДОДАТОК І Пакет ThreadsOSS – прикладна САПР різьбових з'єднань.....	567
ДОДАТОК Й Пакет для геометричного моделювання різьб з відхиленнями за допомогою FreeCAD API.....	590
ДОДАТОК К Результати моделювання замкового P3	597
ДОДАТОК Л Результати моделювання P3 гладких НКТ умовним діаметром 114 мм (ГОСТ 633-80) з пружною моделлю матеріалу.....	599

ДОДАТОК М Input-файли Abaqus для відтворення гармонічного і динамічного аналізу	601
ДОДАТОК Н VBA-макрос для обчислення коефіцієнта запасу втомної міцності за критерієм Сайнса в SOLIDWORKS Simulation	603
ДОДАТОК О Варіанти конструкції пресового з'єднання полімерного стержня зі сталеву оболонкою	605
ДОДАТОК П Макрос Abaqus/CAE для розрахунку з'єднання полімерного стержня зі сталеву оболонкою	606
ДОДАТОК Р Система автоматизованого проектування металополімерних з'єднань на основі вільного програмного забезпечення	612
ДОДАТОК С Програма для обчислення КІН за результатами моделювання МСЕ	621
ДОДАТОК Т Макрос Abaqus/CAE для оптимізації конструкції ШН за критерієм втомної міцності з fe-safe.....	626
ДОДАТОК У Засоби оптимізації конструкції протектора.....	632
ДОДАТОК Ф Абстрактна модель ІС.....	636
ДОДАТОК Х Код експертної системи з проблем надійності та довговічності різьбових з'єднань	640
ДОДАТОК Ц Приклад PLM системи різьбових з'єднань	661
ДОДАТОК Ч Акти впровадження результатів роботи.....	680

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

A – осьовий натяг різьбового з'єднання НКТ за ГОСТ 633-80, мм

API – Application Programming Interface (інтерфейс прикладного програмування)

CAD – Computer Aided Design (автоматизоване проектування)

D – коефіцієнт запасу втомної міцності

DAE – Differential-Algebraic Equations (диференціально-алгебраїчні рівняння)

E – модуль пружності, МПа

E_r , E_θ , E_z – модулі пружності в радіальному, тангенціальному та осьовому напрямку, МПа

$F(h)$ – кумулятивна функцією розподілу імовірностей відмов для випадкової величини h

$f(h)$ – функція густини імовірності відмов для випадкової величини h

FEA – Finite Element Analysis (скінченно-елементний аналіз)

G – модуль зсуву, МПа

h – відносна глибина відмови (обриву)

K_v – показник аварійності штангової колони

$\lg N$ – логарифм циклічної довговічності

N – циклічна довговічність

n_i – відносна частота відмов

R – коефіцієнт асиметрії циклу

R^2 – коефіцієнт детермінації

Δ – видовження ніпеля або скорочення муфти під час згвинчування (величина згвинчування), мм

ν – коефіцієнт Пуассона

σ_0 – границя витривалості для $R = 0$, МПа

σ_{-1} – границя витривалості для $R = -1$, МПа

σ_a – амплітуда напружень циклу, МПа

σ_m – середнє напруження циклу, МПа

σ_r , σ_θ , σ_z – радіальні, кільцеві та осьові напруження в трубі, МПа

σ_b – границя міцності, МПа

$\sigma_{\text{екв}}$ – еквівалентні напруження, МПа

σ_m – еквівалентні напруження за критерієм Мізеса-Губера, МПа

σ_p – осьові напруження розтягу, що виникають в тілі штанги або НКТ, МПа

$\sigma_{\text{руйн}}$ – напруження в тілі штанги, що відповідають зусиллю руйнування пресового з'єднання, МПа

σ_t – границя плинності, МПа

ψ_σ – коефіцієнт чутливості матеріалу до асиметрії циклу

БС – бандаж склопластиковий

ВК – верстат-качалка

ЖЦ – життєвий цикл виробу

ІС – інформаційна система для проектування та підтримки життєвого циклу обладнання ШСНУ

КІН – коефіцієнт інтенсивності напружень

ККН – коефіцієнт концентрації напружень

МАІС – методи активізації інтуїції спеціалістів

МСЕ – метод скінченних елементів

МФПС – методи формалізованого представлення систем

НКТ – насосно-компресорна труба

ООП – об'єктно-орієнтоване програмування

ПКМ – полімерний композиційний матеріал

РЗ – різьбове з'єднання

СА – системний аналіз

САПР – система автоматизованого проектування

СЕМ – скінченно-елементна модель

СПО – спуско-підймальні операції

СПУ – смоло-парафінові утворення

ШН – штанга насосна

ШСНУ – штангова свердловинна насосна установка

ВСТУП

Обґрунтування вибору теми дослідження. Застосування штангових свердловинних насосних установок (ШСНУ) є одним з найбільш поширених у світовій та вітчизняній практиці видобування нафти. Водночас обладнання ШСНУ працює в складних умовах і нерідко відмовляє. Основними факторами, які спричинюють відмови ШСНУ є дія складних циклічних навантажень на штангову колону (розтяг, стиск, згин і кручення) та її значні деформації, підвищений тиск в насосно-компресорних трубах (НКТ), важкодоступність свердловинного обладнання та його робота в умовах тертя ковзання і корозійного середовища. В окремих випадках до них додається підвищений вміст газу, наявність піску в продукції, смоло-парафінові утворення (СПУ), викривлені та похило спрямовані свердловини. Як засвідчує практика, найбільш складними відмовами ШСНУ є відмови її важкодоступної підземної частини, зокрема колони насосних штанг і свердловинного плунжерного насоса, що призводить до значних витрат часу та коштів. Тому основні дослідження потрібно спрямовувати на забезпечення працездатності свердловинного обладнання. Вказані проблеми часто намагаються вирішувати окремо, без системного підходу, що призводить до значних витрат. Наприклад для боротьби з СПУ застосовуються дорогі хімічні реагенти або теплові методи. Тоді як більш ефективним є застосування недорогих скребків-протекторів для насосних штанг (ШН) та штангообертача, які разом вирішують комплекс проблем: СПУ, зношування штанг і НКТ, а також згин, втому та самовідгвинчування штанг [1]. Необхідне комплексне і ефективне забезпечення працездатності ШСНУ, яке спрямовано на вирішення множини проблем її працездатності з найменшими витратами та опирається на системний підхід.

Роботу ШСНУ достатньо глибоко дослідили вчені: Вірновський А. С., Пірвердян А. М., Адонін А. Н., Богомольний Г. І., Грузінов Я. А., Драготеску Н. Д., Белов І. Г., Мищенко І. Т., Івановський В. Н., Гіббс С. Г. Надійність свердловинного обладнання, зокрема штангової колони, досліджували: Баграмов Р. О., Вірновський О. С., Фаерман І. Л., Круман Б. Б., Протасов В. Н., Сулейманов

А. Б., Уразаков К. Р., Копей Б. В., Тараєвський С. Й., Федорович Я. Т., Зубаіров С. Г. та інші вчені.

Сучасні досягнення імітаційного моделювання, автоматизованого проектування (CAD), аналізу даних, штучного інтелекту, теорії систем і системного аналізу (СА) уможливають побудову інформаційних систем для автоматизованого проектування та інформаційної підтримки життєвого циклу (ЖЦ) обладнання ШСНУ, за допомогою яких можна ефективно вирішувати ці проблеми на системному рівні. На даний момент дослідження в цьому напрямі не достатньо розвинено. Тому проблема ефективного дослідження та проектування обладнання комплексно-працевдатної ШСНУ є актуальною.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційну роботу виконано у відповідності до програм науково-дослідних робіт: "Оптимізація параметрів різьбових з'єднань" (номер державної реєстрації 0110U002631), яку було профінансовано завдяки гранту Президента України для підтримки наукових досліджень молодих учених на 2009 рік; "Експертна система з проблем надійності і довговічності штангових свердловинних насосних установок з модулями автоматизованого проектування та моделювання", яка фінансувалась стипендією Кабінету Міністрів України для молодих вчених; "Розробка наукових основ створення з'єднань з металополімерних композитних матеріалів та керування їх зносо-фрикційними та втомними властивостями" (номер державної реєстрації 0115U002279); "Експериментальні дослідження властивостей дослідних зразків бандажів нафтогазопроводів в умовах експлуатації" (номер державної реєстрації 0118U004335).

Мета і завдання дослідження. Метою роботи є удосконалення методології автоматизованого проектування обладнання ШСНУ на основі системного підходу та її використання для комплексного та ефективного дослідження і забезпечення працевдатності ШСНУ.

Досягнення поставленої мети вимагає вирішення наступних **завдань**:

1) розроблення принципів побудови робастних статистичних моделей відмов штангових колон на прикладі відмов штангових колон в НГВУ

"Долина нафтогаз";

2) розроблення принципів компонентно-орієнтованого моделювання ШСНУ та розроблення параметричних імітаційних моделей ШСНУ зі здатністю до простої модифікації та інтеграції в інформаційну систему;

3) розроблення множини параметричних геометричних і скінченно-елементних моделей (СЕМ) та прикладних САПР проблемних деталей та вузлів ШСНУ, які дозволяють системне дослідження їхньої працездатності, зокрема різьбових з'єднань (РЗ) ШН та НКТ, з'єднань склопластикових штанг, ШН і НКТ з дефектами та склопластиковими бандажами (БС), протекторів для ШН, клапана свердловинного насоса;

4) дослідження, на основі моделей, працездатності різьбових з'єднань ШН та НКТ в умовах статичного та динамічного навантаження, у тому числі вібраційного, і розроблення принципів удосконалення та оптимізації їхньої конструкції;

5) дослідження, на основі моделей, працездатності елементів колон штанг і НКТ (пресових з'єднань склопластикових штанг, піделеваторного бурта штанги, штанг та НКТ з склопластиковими бандажами, протектора) та розроблення принципів удосконалення та оптимізації їхньої конструкції;

6) розроблення принципів побудови гнучкої бази знань, яка описує різні фактори, що впливають на надійність різьбових з'єднань ШСНУ;

7) розроблення абстрактної моделі та принципів реалізації інформаційної системи для проектування обладнання та підтримки життєвого циклу ШСНУ (далі – ІС), яка базується на множині моделей обладнання і володіє закономірностями складних систем.

Об'єкт дослідження – проектування обладнання ШСНУ.

Предмет дослідження – принципи побудови, інтеграції та використання компонентів інформаційної системи для автоматизованого проектування обладнання комплексно-працездатної ШСНУ.

Методи дослідження. У роботі застосовано добре відомі та перевірені:

- методи математичної статистики та індуктивного машинного навчання для прогнозування відмов штангової колони;
- метод граничного подання для геометричного моделювання обладнання ШСНУ;
- метод скінченних елементів (МСЕ) задач механіки деформівного твердого тіла та гідродинаміки для моделювання обладнання ШСНУ;
- методи обчислення циклічної довговічності за багатоосьового напруженого стану для відносного порівняння різних конструкцій обладнання ШСНУ;
- основні положення лінійної механіки руйнування для прогнозування ресурсу деталей з втомними тріщинами;
- загальносистемні принципи та закони теорії систем для побудови абстрактної моделі ІС;
- об'єктно-орієнтований та мультиагентний підходи до побудови ІС;
- методи дедуктивного машинного навчання для створення баз знань, у якій різні фактори, що впливають на надійність обладнання, пов'язані причинно-наслідковими зв'язками;
- методи розрахунку навантажень, що діють на штангові колони, для побудови їхніх динамічних моделей;
- методи агентного та компонентно-орієнтованого акаузального моделювання складних динамічних систем для побудови моделей ШСНУ.

Вибір обраних методів зумовлений поставленими завданнями та доведеною їхньою ефективністю в САПР та інших інформаційних системах.

Наукова новизна отриманих результатів полягає в розвитку методології автоматизованого проектування обладнання комплексно-працездатної ШСНУ, яке опирається на принципи теорії систем та інтегровані в ІС імітаційні моделі обладнання, статистичні моделі відмов і бази знань. Зокрема:

- вперше, на основі міждисциплінарного аналізу закономірностей складних систем, розроблено ізоморфну до складних систем абстрактну модель ІС, яка шляхом дихотомічного ділення життєвого циклу (ЖЦ) виробу виділяє класи

функціональних елементів ІС, їхню ієрархію і відношення та є базою для реалізації ефективних ІС;

– набули подальшого розвитку принципи створення ІС, яка відрізняється тим, що належить до класів мультиагентних і гібридних систем, володіє закономірностями складних систем та здатністю до простого розширення та інтеграції гетерогенних компонентів;

– вперше розроблено модель ШСНУ, в якій для спрощення моделювання локальних явищ пружна багатосекційна колона штанг моделюється системою абстрактних автоматів мовою Python;

– вперше розроблено просту для модифікації компонентно-орієнтовану модель ШСНУ мовою Modelica та інтерфейс до неї мовою Python, яка основана на використанні стандартних компонентів "маса", "поступальна пружина з демпфером" і "поступальна сила" для побудови моделі багатосекційної пружної колони штанг;

– набули подальшого розвитку алгоритмічні основи та реалізація програмного каркасу для створення компонентно-орієнтованих акаузальних моделей динамічних систем, який є гнучким у застосуванні завдяки використанню мови загального призначення Python і опису компонентів різницевиими або диференціальними рівняннями засобами пакету SymPy, та розроблено моделі штангових колон на його основі;

– набули подальшого розвитку принципи побудови та використання параметричних моделей та прикладних САПР обладнання ШСНУ, які володіють здатністю до системного дослідження працездатності обладнання, інтеграції в ІС, простого розвитку та базуються на відомих системах CAD/FEA і розроблених автором програмних компонентах;

– досліджено та уточнено вплив конструкційних, технологічних і експлуатаційних параметрів свердловинного обладнання ШСНУ на його показники працездатності, зокрема, які характеризують його статичну та втомну міцність, герметичність та вібростійкість;

– вперше доведено ефективність застосування ансамблів дерев рішень для прогнозування частоти відмов колон насосних штанг за статистичними промисловими даними про відмови.

Практичне значення отриманих результатів полягає в розробленні компонентів ІС та рекомендацій щодо удосконалення обладнання ШСНУ. Зокрема:

– розроблено методику побудови робастних статистичних моделей відмов штангової колони, яка дозволить ефективно ідентифікувати високоаварійні колони і їхні ділянки ще на етапі проектування та виявляти причини відмов;

– розроблено множину параметричних геометричних і скінченно-елементних моделей обладнання ШСНУ, які можуть інтегруватися в ІС: РЗ штанг і НКТ, з'єднань склопластикових штанг, бандажів для ШН і НКТ, протектора, клапана насоса, верстата-качалки (ВК), піделеваторного бурта ШН;

– запропоновано шляхи дослідження та удосконалення обладнання ШСНУ, оптимізації та раціонального вибору значень його параметрів. Зокрема за критеріями статичної та втомної міцності удосконалено конструкцію з'єднання склопластикового тіла з ніпелем, РЗ штанг, з'єднання порожнистих ШН, отримані залежності для вибору оптимального моменту згвинчування РЗ штанг та НКТ, виконано аналіз РЗ НКТ в умовах гармонічного високочастотного навантажування;

– створено базу знань з проблем надійності РЗ ШСНУ, у якій для подання знань використовується логіка предикатів і мова Python;

– інформаційна система підтримки життєвого циклу ШСНУ, як відкрите програмне забезпечення, впроваджена в НГВУ "Долина нафтогаз" ПАТ "Укрнафта" з метою підвищення якості експлуатації ШСНУ та комплексного забезпечення її працездатності (додаток Ч);

– основні результати роботи впроваджено у навчальний процес Івано-Франківського національного технічного університету нафти і газу для підготовки фахівців освітньо-кваліфікаційного рівня магістр за спеціальністю 131 "Прикладна механіка" (додаток Ч).

Положення, що виносяться на захист:

1. Розвиток методології автоматизованого проектування обладнання ШСНУ, що опирається на принципи теорії систем, використовує основу на Python інформаційну систему з гетерогенними елементами (база знань, статистичні моделі відмов, імітаційні моделі, результати симуляції та інші) та застосовується для комплексного та ефективного дослідження і забезпечення працездатності ШСНУ.

2. Розвиток методів комп'ютерного моделювання обладнання ШСНУ, які спрямовані на підвищення адекватності, спрощення використання, розвитку та інтеграції моделей, забезпечення можливості системного дослідження працездатності та оптимізації параметрів обладнання.

Особистий внесок здобувача. Результати роботи отримані автором самостійно. У працях, опублікованих у співавторстві, особисто автором: виконано аналіз відмов і виділено базові класи відмов колон ШН [2]; виконано моделювання та експериментальне дослідження напружень в трубах з бандажами [3-5] та теоретично обґрунтовано доцільність зміцнення і ремонту труб і штанг бандажами [6]; виконано моделювання РЗ ШСНУ [7-15] та елементів колон ШН і НКТ [16-18]; показано необхідність оптимізації моменту згвинчування РЗ ШН з подовженою зарізьбовою канавкою [19, 20]; запропоновано напрямки удосконалення конструкції протекторів [21]; виконано огляд конструкцій з'єднання склопластикової штанги [22]; запропоновано способи моделювання руху колони ШН за допомогою звичайних диференціальних рівнянь [23] та стандартних компонентів мови Modelica [24-26]; розроблено систему компонентно-орієнтованого моделювання мовою Python [27, 28]; розроблено принципи та побудовано базу знань [29, 30]; розроблено принципи побудови ІС для підтримки життєвого циклу РЗ ШСНУ [31, 32].

Апробація результатів досліджень. Основні результати роботи доповідалися і обговорювалися на: міжнародних наукових конференціях "Техніка і прогресивні технології у нафтогазовій інженерії" (2008, 2012, Івано-Франківськ), "Нафтогазова енергетика" (2009, 2013, 2017, Івано-Франківськ), "Прогресивна

техніка і технологія" (2010, Київ), "Машини, обладнання і матеріали для нарощування вітчизняного видобутку та диверсифікації постачання нафти і газу" (2016, Івано-Франківськ), "Innovative ideas in science" (2017, Banja Luka, Bosnia and Herzegovina), "International Conference of Applied Science" (2018, Banja Luka, Bosnia and Herzegovina; 2019, Hunedoara, Romania), "Машини, обладнання і матеріали для нарощування вітчизняного видобутку нафти і газу" (2018, Івано-Франківськ), "Design, Simulation, Manufacturing: The Innovation Exchange" (2019, Луцьк), "Прогресивні технології у машинобудуванні" (2019, Івано-Франківськ-Яремче), "Modern achievements of science and education" (2019, Netanya, Israel) та наукових семінарах кафедри нафтогазових машин та обладнання Івано-Франківського національного технічного університету нафти і газу (2019, 2020, Івано-Франківськ).

Публікації. За результатами досліджень, які викладені в дисертації, опубліковано 58 робіт. Основні наукові результати дисертації висвітлено у 26 наукових публікаціях, які розкривають основний зміст дисертації, з яких 19 – у наукових виданнях, включених до Переліку наукових фахових видань України; 7 – у наукових періодичних виданнях інших держав із напрямку, з якого підготовлено дисертацію; 5 – у закордонних виданнях, проіндексованих у базі даних Scopus (у тому числі 1 – у виданні, віднесеному до першого квартилю (Q1) відповідно до класифікації SCImago Journal and Country Rank).

Структура та обсяг дисертації. Дисертація складається з двох частин загальним обсягом 683 сторінки. Перша частина обсягом 428 сторінок (основний текст – 291 сторінка) містить вступ, шість розділів, висновки та список використаних джерел (378 найменувань) і включає 128 рисунків і 29 таблиць. Друга частина обсягом 255 сторінок містить 26 додатків.

РОЗДІЛ 1

СТАН ПРОБЛЕМИ І ЗАДАЧІ ДОСЛІДЖЕНЬ

1.1 Принципи функціонування ШСНУ

Більше 70% нафтових свердловин України оснащені ШСНУ, за допомогою яких видобувається 50% всієї нафти. Це пояснюється відносною простотою конструкції, простим обслуговуванням і ремонтом, зручністю налагодження, високим коефіцієнтом корисної дії, можливістю експлуатації різнотипних свердловин (у першу чергу мало- та середньодібних). Крім того набутий величезний досвід експлуатації ШСНУ в різних умовах. Застосування ШСНУ буде актуальним і в майбутньому у зв'язку зі поступовим збільшенням малодібних свердловин. Крім того, під час видобутку ШСНУ високов'язких нафт та експлуатації похило-спрямованих свердловин значно зростають навантаження та інтенсифікуються процеси зношування. Це ставить високі вимоги щодо працездатності ШСНУ. Як вказують дослідники [33] ШСНУ рентабельні та часто безальтернативні.

В ШСНУ наземний привід передає зворотно-поступальний рух плунжеру глибинного насоса за допомогою колони ШН, які з'єднані муфтовим РЗ. Під час руху вниз відкривається нагнітальний клапан (установлений в плунжері), закривається впускний клапан (установлений внизу циліндра) і рідина перетікає в надплунжерний простір. Під час руху вверх нагнітальний клапан закривається і рідина піднімається по НКТ, які з'єднані муфтами. В цей же час впускний клапан відкривається і рідина наповнює підплунжерний простір. Колона ШН з'єднана з полірованим гирловим штоком, який з'єднаний з підвіскою приводу. Обладнання гирла призначене для підвішування колони НКТ, герметизації штока за допомогою сальника, та передачі продукції споживачу. На колоні ШН можуть бути розташовані нерухомі протектори (для захисту ШН і НКТ від спрацювання) і рухомі скребки (для очищення поверхні ШН від СПУ) [34]. Як правило вони виготовляються з зносостійких полімерних матеріалів, з малим коефіцієнтом

тертя. Штангообертачі [35] використовуються для повільного обертання колони ШН з метою запобігання самовідгвинчування, рівномірного зношування ШН і НКТ, ефективного очищення НКТ від СПУ скребками. Для відділення рідини від газу і пуску можуть використовуватись газові або пісчані якорі, що встановлюються нижче насоса.

1.2 Аналіз умов роботи та відмов свердловинного обладнання ШСНУ

1.2.1 Колона насосних штанг

Відмова глибинного обладнання ШСНУ призводить до необхідності проведення тривалого і дорогого підземного ремонту свердловин. Більшість відмов ШСНУ пов'язана з поломками насоса, ШН або НКТ [36-38]. Нижче описані типові пошкодження і відмови ШСНУ.

ШН працюють в складних експлуатаційних умовах – циклічні навантаження, тертя і зношування, згин і кручення, корозійне середовище. Обриви колони ШН складають 30-40% усіх відмов ШСНУ. Відмова колони ШН вимагає проведення дорогих та тривалих ремонтних операцій, що пов'язано з необхідністю проведення ловильних робіт і СПО. В розробленій автором таблиці 1.1 розглянуто основні види і причини відмов ШН [2]. Види класифікуються за фізичним механізмом відмови і відмовленим елементом колони. Причини відмов, за ознакою етапу ЖЦ колони, на якому вони виникають, можна поділити на такі базові класи: технологічні (дефекти під час виготовлення прокату і ШН); конструкторські (помилки під час проектування колони ШН та її елементів); монтажні та транспортні (порушення правил зберігання, транспортування і монтажу); експлуатаційні (поломки, які виникають переважно на етапі експлуатації колони ШН).

Поломки як наслідок **виробничих дефектів** (дефекти прокату (заливини, струпи, закати); дефекти штампування головки; дефекти обробки [39]) легко розпізнаються. Засіб запобігання таких відмов – дефектоскопія тіла ШН.

Таблиця 1.1 – Причини відмов тіла ШН (Т), ніпеля (Н), муфти (М), різьби (Р), з'єднання (З): I – статичне і корозійно втомне руйнування, II – механічне спрацювання, III – корозійні дефекти, IV відгвинчування

№	Причини	Види відмов			
		I	II	III	IV
	Технологічні. Дефекти під час виготовлення:				
1	Прокату заготовки	ТНМ			
2	Штампування головки	ТН			
3	Механічної обробки	НМР			3
4	Термообробки	ТНМР	ТМ	ТНМ	
	Конструкторські (помилки проектування колони та її елементів):				
5	Завищені навантаження на колону	ТНМР			3
6	Неправильна компоновка колони	ТНМР	ТМ		3
7	Неправильний вибір матеріалу і термообробки	ТНМР	ТМ	ТНМ	
8	Неправильно вибраний діаметр плунжера насоса і НКТ	ТНМР	ТМ		3
	Монтажні (порушення правил зберігання, транспортування і монтажу):				
9	Пошкодження поверхонь з утворенням концентраторів напружень	ТНМР			
10	Згин ШН	Т	ТМ		
11	Неналежно закріплені НКТ	ТНМР	ТМ		3
12	Неоптимальний момент згвинчування	НМР		Н	3
13	Згвинчування забрудненої муфти, відсутність змащування	НМР		Н	3
14	Неправильна посадка плунжера	ТНМР			3
15	Неоптимальне занурення під динамічний рівень	ТНМР			3
	Експлуатаційні (причини, які виникають переважно на етапі експлуатації):				
16	Корозійно-втомні дефекти	ТНМР			
17	Дефекти механічного спрацювання	ТНМР		М	3
18	Часткове відгвинчування	НР		Н	3
19	Корозійні дефекти	ТНМР	ТМ		
20	Викривлена свердловина	ТНМР	ТМ		3
21	Згин тіла ШН	Т	Т		
22	Згин нижньої частини колони	ТНМР	ТМ		3
23	Навантаження стиску	ТНМ	ТМ		3
24	Падіння обірваної частини колони	ТР			
25	Спрацювання внаслідок тертя об НКТ	ТНМР	ТМ	ТМ	3
26	Удари муфти по НКТ	ТМ	М	М	3
27	Вібрації, динамічні навантаження, удари	ТНМР			3
28	В'язка рідина	ТНМР	ТМ		3
29	Наявність піску	ТНМР	ТМ	ТМ	3
30	Висока концентрація напружень	ТНМР			
31	Інші експлуатаційні дефекти	ТНМР			3
32	СПУ	ТНМР	ТМ		3
33	Заїдання (прихоплення) плунжера	ТНМР	М		3
34	Корозія, проникнення корозійного середовища. Висока обводненість.	ТНМР	ТМ	ТНМ	
35	Високий газовий фактор	ТНМР			3
36	Дефекти насоса	ТНМР	Т		3

Помилки проектування колони та її елементів можуть бути причиною завищених навантажень розтягу, згину, стиску, ударних навантажень, корозії. Їхнім наслідком є відмови колони та іншого обладнання ШСНУ. **Монтажні** пошкодження тіла ШН призводять до концентраторів напружень і можуть бути причиною втомних тріщин. Найбільш небезпечні поперечні концентратори. Тертя ШН і муфт із НКТ спричиняє їхнє взаємне **зношування** (особливо в корозійному і обводненому середовищі [37, 38, 40-42]), що призводить до зменшення площі їхнього поперечного перетину. Значне спрацювання зовнішньої поверхні муфти призводить до зростання концентрації напружень у різьбі [1]. **Удари** муфт в НКТ є результатом удару плунжера насоса об рідину, неналежно закріплених НКТ, заклинювання плунжера чи будь-якої комбінації цих причин [39]. **Циклічні навантаження** в корозійному середовищі призводять до появи і росту втомних тріщин в ШН. Втомні сприяють корозійні та технологічні дефекти, спрацювання тіла. Близько 90% втомних відмов ШН по тілу відбуваються в зоні скруглення між піделеваторним буртом і тілом [43], що пояснюється підвищеною концентрацією напружень та напруженнями згину в цій зоні. Останні можуть виникати під час ходу колони ШН униз. Скруглення між піделеваторним буртом і тілом передбачено стандартом ГОСТ 13877-96. Для штанг ШН22 його радіус 67 мм. Не зважаючи на це, ШН нерідко ламаються в цій зоні, що свідчить про необхідність збільшення цієї величини або захисту цієї ділянки. Напруження **згину** нерідко виникають внаслідок викривлення ШН і часто є причиною втомних поломок ШН. Характерною ознакою такої поломки є коса, неперпендикулярна до осі поверхня перелому [39]. Як правило, причиною згину ШН є недбале транспортування, зберігання, обслуговування або удар внаслідок падіння обірваної частини колони. У разі застосування насосів великого діаметра згин низу колони виникає внаслідок збільшення навантажень стиску і втрати стійкості [44].

Корозія, як результат електрохімічної реакції сталі та корозійно-активного середовища, є причиною майже половини всіх поломок ШН. Внаслідок взаємодії з водою або кислотами утворюються такі сполуки як оксид заліза, сульфід заліза,

карбонат заліза тощо. Вода в продукції нерідко містить значну кількість розчинених домішок і газів, як кислотні гази – вуглекислий газ (CO_2) і сірководень (H_2S). Вони легко розчиняються у воді, яка знижує рН і стає корозійною до сталі. Наявний корозійний шар часто захищає тіло від поширення корозії, але він може періодично пошкоджуватись згином, тертям чи впливом абразиву, що інтенсифікує процес корозії [44]. Термічне зміцнення ШН змешує їхню корозійну стійкість і вимагає нанесення захисних покриттів (цинкові, алюмінієві, полімерні тощо). Спостерігають наступні види корозії штянг [2, 44]: кислотна (процес витравлювання металу кислотою, після якого на поверхні залишаються з гострі, гребенеподібні або гратчасті виступи), хлоридна (дрібні, плоскодонні раковини нерегулярної форми з крутими стінками і гострими краями), CO_2 -корозія (глибокі, з'єднані в довгі ланцюжки раковини з круглою основою, крутими стінками і гострими краями), корозія різнорідних металів (внаслідок з'єднання двох металів з різними електродними потенціалами), H_2S -корозія (сприяє водневому окрихченню, утворює чорну плівку сульфїду заліза і неширокі та розміщені хаотично глибокі раковини, що мають дно округлої форми, круті стінки та фаски на краях), корозія від впливу мікроорганізмів (деякі бактерії виробляють H_2S , органічні кислоти або ензими, окисляють розчинне залізо у воді), киснева корозія (дрібні, плоскодонні, з широкою основою і з тенденцією до об'єднання раковини, які, за наявності CO_2 , можуть мати гострі краї та круті стінки або широкі, гладкі кратери з скошеними краями, якщо наявний H_2S), корозія від солевідкладення (великі локальні раковини внаслідок відколу солевідкладень), корозія спричинена паразитними струмами (глибокі, неправильної форми раковини з гладкими стінками, гострими краями і невеликим конусом в основі раковини).

Пропонувались різні технологічні, конструктивні та експлуатаційні способи підвищення ресурсу колони ШН: захисні покриття (дифузійно-цинкові, полімерні, силікатно-емалеві та інші) [44-52], поверхневе пластичне деформування [45, 53, 54], гартування струмами високої частоти [45, 46, 55, 56], удосконалення конструкції [1, 44, 57], застосування легованих сталей [44], протекторного захисту

та інгібіторів корозії [41, 42], застосування важкого низу колони [44]. Їхнім загальним недоліком є неспроможність комплексно і ефективно вирішити проблеми експлуатації. Так інгібітори, леговані сталі та сілікатно-емалеві покриття характеризуються високою вартістю, склопластикові труби і полімерні покриття володіють низькою зносостійкістю, дифузійно-цинкові покриття не можуть застосовуватись в лужному середовищі [6]. Окремі методи підвищення ресурсу колони ШН вирішують тільки частину проблем, наприклад корозію, або їхнє корозійно-втомне руйнування чи спрацювання. Більш ефективний системний підхід. Наприклад застосування комплексу обладнання, який включає штангообертач, склопластикові ШН, скребки-протектори водночас вирішує проблеми СПУ, корозії, корозійно-втомного руйнування, спрацювання і згину ШН [1]. На даний час не розвинуто методологію ефективного поєднання різних методів підвищення ресурсу. Така методологія не можлива без інформаційної системи, що описує ці методи і зв'язки між ними.

Склопластикові ШН містять стержень з полімерного композиційного матеріалу (ПКМ) зі скляними волокнами та сталеві головки з різьбовими ніпелями [58, 59]. Основні їхні переваги: стійкість до корозії, мала вага, вищий опір корозійній втомі, малий модуль пружності, стійкість поверхні до СПУ. Їхні недоліки: вища вартість, експлуатаційна температура до 100°C, навантаження стиску недопустимі, труднощі в проектуванні колони, не підходить для похилих свердловин та свердловин з в'язкою продукцією, труднощі ловильних операцій після обриву. Малий модуль пружності склопластикової колони разом з застосуванням важкого сталевого низу колони дозволяє збільшити хід плунжера насоса на 20-50% [60] і досягти вищої продуктивності під час коливань колони з білярезонансними частотами [23]. Але ці питання мало вивчені. Не достатньо вивчена також динаміка штангової колони з порожнистими і поплавковими штангами [61]. Важливою задачею є розробка надійного з'єднання сталеві головки зі склопластиковим тілом ШН. Максимальне осьове навантаження розтягу, яке витримує з'єднання є основним параметром під час його проектування, але потрібно враховувати також втомні характеристики з'єднання,

складність конструкції та технології виготовлення, можливість зміни механічних характеристик склопластику під впливом високої температури і агресивного середовища. Гладке клейове з'єднання володіє низькою міцністю адгезиву, а різьбове і клепане з'єднання створюють передумови зменшення міцності ПКМ [22]. Найбільше практичне використання знайшли клиново-клейове [62] і пресове з'єднання [1], яке утворюється шляхом пластичного деформування сталеві оболонки ніпеля таким чином, що її внутрішня частина вдавлюється в склопластикове тіло і утворює з ним натяг. Пресове може мати кращу працездатність (у тому числі під час навантажень стиску) завдяки тому, що навантаження сприймають також внутрішні волокна стержня [22]. Воно також краще підходить для порожнистих штанг [61]. На рис. 0.1 показані можливі варіанти з'єднання. Верхній ряд: пресове гладке, пресове різьбове, пресове ступінчасте, клиново-клейове [62], пресове клейове [22]. Нижній ряд: пресове хвилясте, пресове хвилясте клейове, пресове порожнистих штанг [22, 61], пресове з хвилястими штампами, пресове ступінчасте хвилясте клейове. На даний час не існує ефективних методик автоматизованого проектування таких з'єднань.

Для захисту колон від зношування, зменшення сил тертя, видалення СПУ зі стінок НКТ використовують **протектори ШН** (центратори, скребки-протектори) – пластмасові деталі, які нерухомо кріпляться на тілі ШН [1, 34]. Проблемою їхнього застосування, особливо на склопластикових ШН, є значна сила гідродинамічного опору, яка може призвести до появи небажаних зусиль стиску в колоні [63]. Аналіз відомих конструкцій протекторів з урахуванням їхніх гідродинамічних та зносостійких характеристик показав, що найкращі експлуатаційні характеристики мають протектори з малою площею поперечного перетину, великою площею тертя (контакту) з НКТ та достатньою площею зчеплення з ШН [21]. Площа тертя визначає зносостійкість лопаті – збільшення площі призводить до зменшення контактних тисків в зоні тертя. Тому, конструкція протектора повинна бути оптимальною за критеріями мінімальної сили гідродинамічного опору та максимальної площі тертя зі стінкою НКТ. Додатковий показник якості – міцність кріплення протектора на штанзі, яка

залежить від площі їхнього зчеплення [61]. Під час ходу колони вниз протектор з гвинтовими канавкам додатково створює крутний момент навколо осі. У випадку достатньої кількості таких протекторів на колоні цей момент дозволяє запобігати самовідгвинчуванню РЗ і зменшити навантаження на штангообертач [64, 65]. Гвинтова лінія канавки повинна мати напрямок проти годинникової стрілки. В праці [65-67] на основі методики розрахунку осьових турбін отримано формули для обчислення крутного моменту, за якими виявлено, що впродовж циклу роботи ШСНУ багатолопатеви протектор може створювати максимальне миттєве значення моменту 0,27 Нм. В праці [68] на основі цих формул описано методику вибору раціональних параметрів такого протектора за критерієм максимального крутного моменту. Моделювання МСЕ спроектованого за цією методикою шестилопатевого протектора показало, що він створює крутний момент 0,072 Нм [65]. Монтаж 100 таких протекторів на колоні дозволить зменшити крутний момент, потрібний штангообертачу (100-200 Нм), мінімум на 7-27 Нм. Проте на даний час не запропоновано методики оптимізації параметрів протекторів за усіма цими критеріями.

Основними проблемами експлуатації стандартних муфтових **різьбових з'єднань ШН** є їхні втомні поломки та самовідгвинчування [37]. Найчастіше втомні руйнування упорного РЗ ШН відбуваються в зонах перших витків різьби ніпеля та розвантажувальної канавки ніпеля [37, 43, 44]. Це пояснюється підвищеною концентрацією напружень в цих зонах. Втомні поломки спричинені нерівномірним розподілом навантаження по витках різьби та значною концентрацією напружень у зарізьбовій канавці. Їм також сприяє низький опір РЗ навантаженням згину, послаблення натягу РЗ, корозія та зношування зовнішньої поверхні муфти [43, 44]. Причиною самовідгвинчувань є нерівномірний розподіл навантаження по витках, недостатня площа опори, навантаження стиску, згину і кручення, які можуть виникати у нижній частині колони ШН, недотримання правил згвинчування (недостатній момент згвинчування, згвинчування забрудненої муфти [43]). Часто підвищення відгвинчувань РЗ спостерігають у свердловинах з інтенсивним СПУ. Відомі також деформації та спрацювання

витків різьби, яке відбувається в результаті неправильного змащування і необережних виконань монтажних операцій. Втомні поломки по тілу муфти зумовлені високою концентрацією напружень в зоні останньої робочої впадини різьби муфти, спрацюванням муфти та недотриманням правил експлуатації, які забороняють удари ключем по муфті. Стандартні РЗ порожнистих ШН мають додаткову проблему – низьку герметичність. В працях [69, 70] виконано скінченно-елементний аналіз способів вирівнювання навантаження на витки різьби муфтового РЗ ШН без зміни параметрів профілю різьби: застосування муфти розтягу-стиску із змінним перетином розтягнутої частини; зменшення модуля пружності матеріалу муфти; застосування покриття різьби муфти пластичним матеріалом; зміна модуля пружності матеріалу ніпеля; вибір оптимальної довжини згвинчування; застосування різьби ніпеля, утопленої в різьбі муфти; застосування зрізу перших витків різьби муфти; застосування розвантажувальної канавки оптимальної форми і розміру; розтиск перших витків різьби муфти у радіальному напрямку; обтиск останніх витків різьби муфти; розтиск останніх витків різьби ніпеля; попереднє пластичне деформування перших витків ніпеля або муфти. Перспективним є поєднання різних способів, наприклад зменшення модуля пружності матеріалу муфти і попереднє пластичне деформування перших витків високим моментом згвинчування. Для зменшення концентрації напружень в ніпелі стандартом ГОСТ 13877-96 передбачено в його конструкції розвантажувальну канавку. Відомо, що збільшення довжини такої канавки може підвищити втомну міцність РЗ [20, 71, 72], але проблемою є визначення довжини канавки, яка б забезпечила задану циклічну довговічність ніпеля. Результати досліджень [19, 20] показали, що збільшення довжини розвантажувальної канавки ніпеля ШН зменшує концентрацію напружень в ньому. Додатково, у комплексі з застосуванням довгих протекторних вставок [73], така конструкція ніпеля дозволить захистити муфту від зношування.

Відомі також двоопорні конічні замкові РЗ бурильних труб, які застосовуються в складних умовах буріння [74, 75]. У порівнянні з одноопорними вони витримують більший крутний момент, володіють більш рівномірним

навантаженням на витки [74] та вищою герметичністю. Відомі дослідження таких РЗ методом скінченних елементів (МСЕ), в тому числі на основі 3D моделей з гвинтовою різьбою, що дозволяє моделювати згин і кручення [76-80]. Виконано огляд досліджень і проаналізовано переваги і недоліки наступних РЗ: циліндричного і конічного, одноопорного і двоопорного, з зарізьбовими канавками і без них. Перевагою конічних РЗ є швидкість і зручність згвинчування. Відповідно зменшується імовірність пошкодження різьби під час згвинчування. Операції згвинчування-розгвинчування для ШН виконуються не так часто, тому конічне РЗ не є обов'язковим. Недоліком конічного РЗ є нижча втомна міцність ніпеля внаслідок зменшення його діаметра в зоні торця. Втомна міцність муфти в зоні останнього витка може бути вищою внаслідок збільшення її товщини. Зменшення конусності дозволяє збільшити площі опор і збільшити міцність під час кручення, згину і стиску [78]. В одноопорному РЗ розподіл напружень по впадинах різьби ніпеля дуже нерівномірний – навантажені тільки перші витки [1, 76]. Розподіл напружень в двоопорному РЗ більш рівномірний, але з певним недовантаженням середніх витків [77, 79, 80]. Розподіл навантажень по витках є важливою характеристикою РЗ і визначає його опір самовідгвинчуванню і герметичність. Однак за ним не можна робити висновок про втомну міцність РЗ. Двоопорне РЗ володіє вищою міцністю опор до великого крутного моменту [78]. Збільшення крутного моменту не спричиняє різкого зростання напружень в перших витках ніпеля [79]. Міцність на кручення залежить від площ опор [75, 78]. На основі 3D моделей з гвинтовою різьбою виявлено, що таке РЗ володіє кращою статичною і втомною міцністю під час кручення, стиску, розтягу та згину [78-80]. Недоліком двоопорного РЗ є суттєва залежність втомної міцності від відношення натягів на основній (зовнішній) і додатковій (внутрішній) опорах [76, 81]. Доведено, що використання додаткової опори може покращити розподіл еквівалентних напружень по впадинах витків, але завеликий натяг на внутрішньому торці може спричинити руйнування муфти [76]. Це потребує малих допусків довжини ніпеля і глибини муфти. Необхідно також контролювати ці розміри перед згвинчуванням, оскільки торці можуть бути пошкоджені під час

експлуатації. Для подолання цього недоліку в РЗ можна застосовувати пружні або пластичні елементи, наприклад постійні пружні або змінні пластичні шайби в зоні додаткової опори. Їхня жорсткість або пластичність повинні залежати від потрібної величини допуску натягу і контактного тиску на додатковій опорі. Збільшення довжини зарізьбової (розвантажувальної) канавки призводить до зменшення концентрації напружень у зоні перших робочих витків різьби [1]. 3D моделювання двоопорних РЗ показало, що збільшення довжини плечей (без канавок) та зменшення конусності призводить до вищої статичної та втомної міцності, особливо під час кручення [78]. В двоопорних РЗ необхідні зарізьбові канавки в ніпелі та муфті. Їхній недолік – збільшення довжини РЗ. Крім того, значне збільшення довжини зарізьбової канавки спричиняє зменшення її жорсткості та вимагає оптимізації величин натягів. Необхідне обґрунтування доцільності застосування двоопорних РЗ порожнистих ШН шляхом моделювання напружено-деформованого стану їхніх різних варіантів та обчислення значень еквівалентних напружень і коефіцієнта запасу втомної міцності.

В праці [72] виявлено, що зменшення висоти витків різьби збільшує напруження у впадинах витків ніпеля в середній частині РЗ та зменшує їх в зоні торця ніпеля, а в напруження в останній западині різьби муфти значно зростають. Проте не дослідженим є вплив висоти витків на коефіцієнт запасу втомної міцності та контактні напруження в різьбі.

1.2.2 Насосно-компресорні труби

Колона НКТ сприймає власну вагу і вагу рідини. Додатково на неї діє змінне навантаження, спричинене процесом відкачування. Відмови свердловинного обладнання можуть бути спричинені технологічними дефектами, дефектами, які утворились під час СПО, та експлуатаційними дефектами. Статичні та втомні обриви по тілу і різьбі можуть бути наслідком: значних статичних і циклічних навантажень, концентрації напружень в останніх витках ніпеля та в місцях технологічних чи експлуатаційних дефектів, пошкодження тіла

і різьби під час СПО (наприклад пошкодження тіла ключами), тертя і зношування колоною ШН, корозії [42, 82-84]. Одним з найбільш небезпечних видів корозії є водневе окрихчування та сульфідне розтріскування під напруженням, внаслідок проникнення атомарного водню [85]. Часто проблемою є відкладення парафіну і солей на стінках НКТ [82, 84]. Вони призводять до збільшення навантаження на ШСНУ, зменшення пропускної здатності НКТ та необхідності проведення підземних ремонтів, пов'язаних з очищенням стінок НКТ. Так, нафти родовищ Прикарпаття характеризуються високим вмістом парафіну (9,5-12,5%) і смол (14,5-17,5%). Сучасні методи боротьби з СПУ (теплові, хімічні) дорогі та часто неефективні. За даними [42] 80% відмов НКТ пов'язані зі зношуванням колоною ШН. Найбільш ґрунтовно закономірності корозійно-механічного зношування колони ШН і НКТ вивчені в працях [41, 42]. Експериментальні дослідження [42] показали, що знос НКТ у прісній воді в 11 раз більший ніж в чистій нафті. В працях [38, 40] виявлено, що високий відсоток води в продукції (більше 90%) може бути причиною різкого скорочення терміну служби НКТ (в 4-5 раз). Це можна пояснити інтенсифікацією процесу зношування колоною ШН. Для підвищення ресурсу НКТ застосовують такі методи: захисні та антипарафінові покриття [86], закачування інгібіторів корозії [41, 82], застосування НКТ з ПКМ [87], застосування протекторів на колоні ШН та штангообертачів [1, 42], або обертачів НКТ. Штангообертач інтенсифікує зношування муфт на 30%, але значно підвищує ресурс муфт і НКТ внаслідок рівномірного спрацювання їхніх поверхонь [42].

Різьбове з'єднання НКТ. Більшість аварій НКТ пов'язані з відмовами РЗ [82-84, 88-90]. Основними видами відмов РЗ є втомне руйнування ніпеля, вирив ніпеля з муфти, порушення герметичності, заїдання різьби, знос муфти і НКТ в зоні РЗ, самовідгвинчування, корозія, знос і зминання різьби, задири на перших витках, зрив витків [82-84]. Негерметичність РЗ може бути спричинена неправильним моментом згвинчування, вібраціями, забрудненням різьби, невідповідним змащенням, пошкодженням різьби під час СПО, спрацюванням та промивами різьби. Муфтове з'єднання гладких НКТ за ГОСТ 633-80 має

незахищену від корозії та СПУ центральну частину і володіє низькою герметичністю. Внаслідок цього можуть виникати корозійні пошкодження різьби на кінці ніпеля і в середині муфти [84]. Ущільнюючі кільця, рекомендовані стандартами ГОСТ Р 52203-2004 та API-5СТ, не забезпечують захисту центральної частини ЗР. Відмови РЗ можуть спричинити такі фактори як концентрація напружень у перших витках [7, 91-94], згин НКТ, знос НКТ колоною ШН, багатократне згвинчування, недбале використання (недостатній момент згвинчування, згвинчування з перекосом, відсутність попереднього згвинчування вручну, удари під час посадки НКТ в муфту, згвинчування забрудненої різьби, неправильне змащування, відсутність запобіжних деталей), циклічні навантаження, обводненість, корозія, зношування різьби (корозійно-механічне, контактано-втомне, ерозійне) [83, 84, 88, 95]. У випадку застосування занурюваних відцентрових електронасосів руйнування РЗ найчастіше відбувається в нижній частині колони внаслідок її осьового і радіального навантажування і вібрацій насоса [95]. У випадку фонтанного або глибинонасосного способів видобування РЗ частіше руйнуються вверху колони, де більші статичні та циклічні навантаження [95].

Причиною вказаних відмов можуть бути і вібраційні навантаження на колону НКТ. Характер цих навантажень, насамперед, залежить від способу видобування нафти. У випадку застосування занурюваних відцентрових електронасосів можливі вібрації частотами 0-8 кГц [96]. У випадку нормальної роботи штангового глибинного насоса спектр коливань колони ШН містить частоти, які рівні частоті ходу підвіски (0,1-0,5 Гц) та частоті вільних коливань колони ШН (0,5-10 Гц). Остання наближено може бути розрахована за формулою $1204/L$ (Гц) [97], де L – довжина колони в метрах. Рідина і колона НКТ теж є пружними тілами і можуть мати вільні коливання певної частоти. Наявність вищих частот у спектрі, як правило, спричинена різноманітними порушеннями нормальної роботи насоса. Аналіз промислових динамограм, які наведені у праці [98] показує, що такі частоти спостерігаються під час: заїдання плунжера, прихоплення плунжера у вставному насосі, витоків рідині в нагнітальній частині,

відкачування рідини з високим вмістом газу, високої посадки плунжера, запізнення посадки кульки клапана. На величину і характер коливань значний вплив мають сили тертя, які виникають під час експлуатації або спуско-підіймальних операцій (СПО) [88, 98-100]. Вільні високочастотні коливання НКТ можуть виникати і після різноманітних ударних навантажень, наприклад ударів торців муфт ШН об незахищені торці НКТ під час руху колони ШН.

В праці [93] виконано статичний аналіз моделі муфтового РЗ НКТ для різних значень натягу і зовнішнього осьового навантаження. В працях [7, 94] таке РЗ проаналізовано в діапазоні частот навантаження 0-20 кГц за допомогою програмного комплексу Ansys. Однак було проведено тільки спрощений частотний (modal) і гармонічний (harmonic) динамічні аналізи.

В праці [101] запропонований спосіб герметизації НКТ шляхом газопломеневого порошкового напилення шару цинку на різьбових поверхнях ніпеля і муфти товщиною від 0,03 мм і більше. Коефіцієнт тертя спокою поверхонь цинк-цинк в умовах змащування може досягати дуже малих значень (до 0,04). Тоді як поверхонь сталь-сталь – 0,1-0,2. Під час згвинчування менший коефіцієнт тертя покращує згвинчування РЗ і дозволяє досягти його вищої міцності. Високі пластичні, антифрикційні та антикорозійні властивості цинку забезпечують надійну герметизацію РЗ НКТ. Проте найбільш ефективним є напилений герметизуючий шар товщиною 0,10...0,15 мм з цинково-алюмінієвого сплаву у співвідношенні 6:1 [101]. У зв'язку з низьким коефіцієнтом тертя в такому РЗ актуальним є аналіз його відклику на гармонічні навантаження. Було виявлено залежність значень зазорів в різьбі від коефіцієнта тертя для різних значень зовнішнього навантаження розтягу [92]. Але недослідженою є поведінка РЗ з низьким коефіцієнтом тертя в умовах гармонічного навантажування. Отже потрібно виконати аналіз відклику СЕМ муфтового РЗ НКТ з різними значеннями коефіцієнта тертя на гармонічне зовнішнє осьове навантаження частотами 0-20 кГц.

Перспективним є застосування НКТ з високоміцних сталей. Класична технологія різьбонарізання різцями вимагає застосування різців з нульовим

переднім кутом γ . Але це значно ускладнює умов різання високоміцних сталей НКТ [10]. Для таких сталей необхідні від'ємні значення переднього кута [102]. Відомо, що поворот стандартної різбонарізної пластини на такий кут призводить до зміни кінематичних кутів різання та спотворення профілю різьби (зміни кута профілю) [102]. В праці [10], на основі FreeCAD-моделі [8], розроблено геометричну модель різбонарізання ніпелів НКТ, яка дозволяє довільні значення переднього кута γ , заднього кута α та кута нахилу різальної кромки λ . За допомогою алгоритму для FreeCAD API [10] можливо обчислювати значення кінематичних задніх α_k і передніх γ_k кутів в різних точках різальної кромки різця. Зокрема для моделей різбонарізання ніпелів гладких НКТ з зовнішніми діаметрами 33 мм і 114 мм відповідно ГОСТ 633-80 (еквіваленти ніпелів 1.315 і 4-1/2 гладких НКТ за API Spec. 5CT) отримано залежності α_k і γ_k від α , γ , λ [10] в різних зонах різьби. Цей алгоритм і залежності можуть бути використані для вибору значень α , γ , λ за допустимими значеннями α_k і γ_k [10]. Проте зміна кута профілю різьби може вплинути і на працездатність РЗ, зокрема на герметичність і міцність. Для обґрунтування можливості застосування некоригованих різців з від'ємним переднім кутом постає додаткове завдання побудови відповідних СЕМ з'єднань і аналізу їхнього напружено-деформованого стану за критеріями втомної міцності та герметичності.

1.2.3 Свердловинні штангові насоси

В ШСНУ використовують вставні (найчастіше) і трубні свердловинні штангові насоси. Перші більш зручні в експлуатації, але і менш продуктивні. Циліндри насосів бувають втулкові та безвтулкові. У більшості випадків застосовують прості за конструкцією і надійні кулькові клапани. Вибір типу насосу, зокрема клапанного вузла, залежить від експлуатаційних факторів [103, 104]. Міжремонтний період сучасних насосів може бути 500 діб і більше [105]. Найбільш розповсюдженими типами відмов насосів є спрацювання клапана, спрацювання плунжера і циліндра, СПУ в насосі, руйнування клапана,

руйнування клапанної клітки [104, 106]. Найчастіше зустрічаються відмови клапанних вузлів [104]. В основному ці відмови спричинені різними видами зношування (механічне, ерозійне, корозійне), наявністю абразивних частинок в рідині, відкладенням парафіну і солей, згином низу колони. Рідше зустрічається деформація штока, деформація корпусу, перекіс втулок, заклинювання плунжера, заклинювання клапана. Ці відмови спричинюються збільшенням навантажень на шток, згином низу колони, відкладеннями або абразивними частинками, неправильною посадкою плунжера. В праці [106] виявлено, що в свердловинах з великою обводненістю продукції (50-90%) середній наробіток на відмову насосів з втулковими циліндрами більш, як в три рази перевищує середній наробіток на відмову насосів з безвтулковими циліндрами. Отже обводненість значно інтенсифікує процеси зношування в парі "плунжер-циліндр". Зі збільшенням діаметра насоса імовірність безвідмовної роботи та середній наробіток на відмову його зменшується. Середній наробіток на відмову вставних насосів більший ніж невставних [36, 107].

У більшості випадків зворотні клапани свердловинних штангових насосів для видобування нафти виконані у вигляді самодіючих кульових клапанів [108]. Їхня надійність визначає ефективність видобування нафти штанговими установками. Такі клапани володіють запізненням посадки, значними контактними тисками в сідлі, не підходить для добування в'язких нафт і для похило-скерованих свердловин. Замість них можуть використовуватись керовані кулькові клапани, клапани з примусовим закриттям (кулькові, каплевидні або тарілчасті), клапани з напрямним штоком, самоустановлюючі клапани, демпфуючі клапани та інші [104, 105, 109]. Математичні моделі кульових клапанів досліджували: Пірвердян А.М., Давлетшин Х.Г., Степанова І.С., Івановский В.Н. Ці моделі для спрощення обчислюють коефіцієнт витрати клапана для статичної системи. Для адекватного моделювання ШСНУ важливим є обчислення коефіцієнта витрати клапана як залежності від висоти підйому кульки [104]. Для цього потрібна гнучка параметрична СЕМ клапана [110, 111]. Принципи створення моделей клапанів для Modelica описані в [112]. Методика

об'єднання Modelica-компонентів і скінченно-елементних гідродинамічних моделей в одній моделі для адекватного моделювання клапанів описана в [113]. Математична динамічна модель клапана дозволить дослідити його функціонування та прийняти рішення щодо удосконалення конструкції. Для побудови адекватної динамічної моделі важливо знати залежність перепаду тисків ΔP в клапані або коефіцієнта витрати клапана μ від висоти підйому кульки h [111, 114]. На даний час існують кілька таких емпіричних залежностей [108, 115], які не враховують конструктивні особливості кульових клапанів різних типів. Цю проблему можна вирішити шляхом застосування методів обчислювальної гідродинаміки та відповідних програмних продуктів, які їх реалізують. Таким чином необхідним є створення параметричної СЕМ кульового клапана для автоматизованого отримання залежностей $\Delta P(h)$ або $\mu(h)$.

1.3 Огляд ремонту та зміцнення труб і штанг склопластиковими бандажами

Відомо чимало методів ремонту труб [116, 117] – заварювання дефектів, сталеві муфти, ремонт лейнерами, композитно-муфтова технологія (сталева муфта та ПКМ між муфтою і трубою), полімерна муфта зі сталевим вузлом затягування, полімерні муфти без сталевих елементів (бандажі з скляними або вуглецевими волокнами, муфти «Clock Spring», термоусадочні муфти). Більшість цих методів застосовують для ремонту трубопроводів без зупинки їхньої експлуатації та рідко застосовують для ремонту НКТ чи ШН. Це можна пояснити відсутністю ефективних методів обґрунтування доцільності ремонту. Проте зміцнення і ремонт НКТ і ШН ПКМ-бандажами може бути ефективним завдяки їхнім наступним особливостям: на відміну від захисних покриттів одночасно забезпечують корозійну стійкість і зменшують напруження в зоні дефекту (у разі достатньої їхньої товщини); стійкі до СПУ на їхніх поверхнях; не створюють значних концентраторів напружень після ремонту; менша маса і розміри бандажів; відсутній термічний вплив на трубу під час ремонту; можуть бути

застосовані для більшої кількості типів дефектів, у томі числі тріщин; мають універсальну конструкцію для будь-яких дефектів, зокрема підходять для труб з відхиленнями круглості; наносяться в заводських умовах, що забезпечує їхню вищу якість; можуть застосовуватись разом з іншими методами зміцнення штанг (поверхневим пластичним деформуванням, гартуванням струмами високої частоти, металізаційними покриттями) [118]; для додаткового захисту на місці нанесення бандажа можна додатково сформувати пластиковий протектор-центратор для ШН або НКТ.

Для ремонту дефектів трубопроводів застосовуються два типи ПКМ: заливний (для заповнення дефектів, пов'язаних з втратою металу) і муфтовий (для відновлення несучої здатності) [119]. Вони можуть бути також поєднані. До заливних ПКМ відноситься клей «Моноліт» (ВНИИГАЗ) и молекулярметали фірми «Діамант-Металопластик» ГМБХ (Німеччина), які показали найкращі адгезійні та міцнісні характеристики [119]. Усесторонні випробування довели також хороші характеристики композитних матеріалів РЭМ-Сталь і РЭМ-Алюміній (ТУ 2257-005-00396558-98) фірми «Порсіл лтд» м. Санкт-Петербург [120] на епоксидній основі, які містять $65\% \pm 5\%$ дрібнодисперсного металічного компонента (порошок з нержавіючої сталі типу 18-8, або порошок з алюмінію АСД-4). Основні характеристики матеріалу РЭМ-Сталь: напруження руйнування σ_B під час стистку, розтягу, згину і зрізу – 60, 25, 40, 25 МПа відповідно; модуль пружності під час розтягу $E=1600$ МПа; адгезійна міцність до сталеві поверхні 15 МПа; теплостійкість $-60...130$ °С; низьке водопоглинення і хороша хімічна стійкість (в тому числі до нафти і сірководню); гарантія – не менше 10 років. Ці матеріали рекомендуються також для ремонту промислових нафтопроводів з наскрізними дефектами (отворами) з застосуванням закладних деталей, накладок і бандажів [120]. Аналіз характеристик показує, що вони підходять і для ремонту НКТ. Для кращої адгезії рекомендують видалення іржі ортофосфорною кислотою і забезпечення шорсткості поверхні Rz 175 мкм [120]. Відомий також ПКМ Belzona 1111 (Super Metal) на основі епоксидної смоли без розчинників і порошку кремнієвої сталі з адгезією до сталі 22 МПа і температуростійкістю 49 °С в

вологодому середовищі [121]. До муфтових ПКМ відноситься гнучкий анізотропний рулонований склопластик (ГАРС) за ТУ 2296-152-05786904-99 ($\sigma_{\text{в}}=950-1390$ МПа, $E=24,8-52$ ГПа), композиційна спіральна муфта (КСМ) за ТУ 92-115-14-98 ($\sigma_{\text{в}}=538-684$ МПа, $E=21,4-28$ ГПа) та вуглецева однонаправлена стрічка (УОЛ-300-1). Для ремонту небезпечних дефектів рекомендують застосовувати бандажі зі значеннями модулів пружності $E \geq 0,1E_{\text{тр}}$ і $\sigma_{\text{в}} \geq \frac{3}{4}\sigma_{\text{втр}}$ ($E_{\text{тр}}$, $\sigma_{\text{втр}}$ – модуль пружності та границя міцності матеріалу труби) [120]. Як зв'язуюче використовують конструкційний композитний адгезив (ККА) типу ПГР-4 за ТУ 2225-009-00396558-99 з міцністю на розрив $\sigma_{\text{в}}$ 45 МПа. ПКМ [119] застосовуються для ремонту труб з такими дефектами як загальна корозія (зовнішня і внутрішня); корозійні виразки; задири, подряпини, відколи; каверни; вм'ятини глибиною до 5% діаметра труби. Не підлягають ремонту ПКМ такі дефекти як тріщини; наскрізні дефекти; стрес-корозійні дефекти; гофри; вм'ятини в поєднанні з додатковим концентратором [119]. Допустимість ремонту і оптимізація шарів муфти виконується за допомогою спеціального програмного забезпечення в залежності від параметрів труби і дефекту [119]. Була виконана оцінка несучої здатності трубопроводів діаметром 530 – 1420 мм, відремонтованих із застосуванням таких композитних матеріалів [120]. Ізоляційний композитний бандаж "ІКП" за ТУ У 26.1–02070855.003–2010, розроблений в ІФНТУНГ [122, 123], являє собою трьохшарову конструкцію: епоксидна ґрунтовка, бандаж склопластиковий (БС) на основі скловолокна марки Т-10-80 та зв'язуючого марки ЕДТ-10П (епоксидна модифікована смола ЕД-10 і модифікований затверджувач на основі амінів), поліуретанова двохкомпонентна композиція 3М Scotchkote 352. Останній поліуретановий шар дозволяє експлуатувати покриття також під водою. Усесторонні випробування БС показали його добрі характеристики, у тому числі тривалу водостійкість за 65°C, адгезію до сталі 7,8 МПа за 20 °С, міцність під час удару 15 Дж [123]. Експериментально доведена ефективність покриття для гальмування росту коротких тріщин в трубопроводах [122]. Промислові випробування експериментально-дослідних зразків БС проводили в НПС «Долина» Дрогобицького НУ філії МН «Дружба», а також на нафтохімічних

підприємствах ЗАТ «ЛУКОР» і ТОВ «Карпатнафтохім» [124]. Проведено намотування бандажів на дефектні ділянки трубопроводів, якими транспортували конденсат, утворений з пари, що поступає в цех під тиском 1 МПа та температурою 80°C (дефект – вм'ятина глибиною 0,5 мм, шириною 7,0 мм, довжиною 40 мм), а також повітря технологічне з тиском 0,5-0,8 МПа та температурою -20...+30 °C (дефект – вм'ятина глибиною 0,6 мм, шириною 6,0 мм, довжиною 30 мм). За 4 роки їхньої експлуатації порушень цілності БС, розшарувань, порушень адгезійного зв'язку не спостерігали.

В працях [125, 126] експериментально обґрунтована доцільність ремонту та зміцнення штанг БС. Виявлено, що циклічна довговічність ШН з БС невеликої товщини в корозійному середовищі (3% розчин NaCl) дещо вища ніж циклічна довговічність ШН на повітрі – БС підвищує ресурс на 80% для амплітуди 240 МПа [126]. Розвиток тріщини в таких ШН починається в момент, який складає 80% її повного ресурсу, і ШН ламаються під час більшого критичного розміру тріщини [126]. В ШН без БС тріщина починає розвиватися в момент, який складає 50% її повного ресурсу. Це свідчить про добрі ізолюючі властивості БС протягом 80% від повного ресурсу.

Стандарт ГОСТ 633-80 дозволяє ремонт забракованих НКТ з наступним їхнім випробуванням. Як правило ремонт НКТ містить такі операції: мийка; візуальний контроль, сортування і відбраковування; перевірка труб на відповідність стандарту; очищення внутрішньої порожнини труб від відкладень; дефектоскопія; товщинометрія; відрізка дефектних ділянок; ремонт різьбових частин і муфт; заміна муфт; гідровипробування [82, 127]. Перед гідровипробуваннями можуть бути додані операції для ремонту НКТ бандажами. Сучасні методи дефектоскопії (як правило електромагнітні та ультразвукові) дозволяють автоматизоване визначення величини, координат і характеру дефектів з високою точністю [128-130]. Стандарт ГОСТ 633-80 рекомендує відбраковувати труби з дефектами, якщо сигнал від ультразвукового дефектоскопа перевищує сигнал від повздовжнього прямокутного дефекту з глибиною $5 \pm 0,75\%$ від номінальної товщини стінки, довжиною мінімум 50 мм та

шириною максимум 1 мм. Найбільш сучасні промислові методи магнітної дефектоскопії дозволяють виявляти тонкі поверхневі та внутрішні дефекти висотою більше 5% від товщини стінки, а також об'ємні дефекти в тілі труби, порівнювані зі свердлінням діаметром 0,79 мм [130].

За виробничими даними традиційний ремонт (без БС) однієї НКТ коштує в 5-7 раз дешевше ніж купівля нової (~20-30 тис. грн. за тону, ціна металобрухту 6 тис. грн. за тону), а ресурс відремонтованої НКТ досягає 80%. Ремонт НКТ лейнерами коштує в 2 рази дешевше нових, а ресурс збільшується в 2-3 рази [85]. Ремонт НКТ і ШН бандажами буде мати найбільшу економічну ефективність під час ремонту локальних дефектів, які є недопустимими для експлуатації та обумовлюють відбракування виробу, наприклад окремих ненаскрізних корозійних язв, вм'ятин, тріщин [84]. Збільшення площі та кількості дефектів призводить до збільшення витрат часу на ремонт і витрат на матеріали. Теоретично можливим є нанесення БС на усю зовнішню поверхню виробу. В комплексі з ремонтом внутрішньої поверхні НКТ лейнерами [85] це дозволить отримати виріб зі значно вищим ресурсом. Такі НКТ можна буде застосовувати для ШСНУ, що не можна сказати про склопластикові НКТ без внутрішньої сталеві частини. Отже доцільність ремонту БС повинна бути розрахована в кожному окремому випадку за таким загальним автоматизованим алгоритмом: побудова параметричної геометричної і скінченно-елементної моделі виробу з дефектами, БС і без нього (за даними різних взаємодоповнюючих методів дефектоскопії); оптимізація розмірів БС за критеріями міцності та витрат; прийняття рішення про доцільність ремонту. Для реалізації цього алгоритму потрібно розробити методіку побудови моделей та оптимізації БС.

1.4 Проблеми статистичних моделей відмов колон насосних штанг

Для виявлення залежностей частоти відмов колон від різних факторів часто використовують статистичні методи обробки даних про відмови [1, 44, 45, 88, 131-133]. На основі цих методів розроблені: емпіричні залежності частоти

втомних відмов колон ШН від діаметра насоса і штанг, довжини колони, режимів відкачування, матеріалу ШН і властивостей робочого середовища [131, 134, 135], від приведенного напруження [44, 88, 136] та від відносної глибини обриву для насосів великого і малого діаметра [1, 44, 131, 137]; залежності кількості відмов від викривленості свердловини і обводненості продукції [88, 135]; залежності кількості відмов у в'язкому середовищі від співвідношення динамічних і статичних навантажень [88]; залежності кількості відмов від глибини спуску насоса [138]; складені за експлуатаційними даними криві втоми ШН [139]. Відомі також системи прогнозування відмов на основі аналізу динамограм або ватметрограми [140, 141], а також система діагностики відмов ШСНУ на основі методу аналізу першопричин відмов (Failure Root Cause Analysis) [142].

В праці [2] проведений аналіз відмов ШН в умовах НГВУ «Долина нафтогаз» за 1999-2007 рік. На основі даних про поломки ШН виведено залежність ймовірності безвідмовної роботи ШН у 2006 році $P_1(N)$ та у 2007 році $P_2(N)$ від числа циклів навантаження N (млн. циклів) [2]. Остання показує дещо менші значення:

$$P_1(t) = e^{-\left(\frac{N}{16,75}\right)^{1,66}} ; P_2(t) = e^{-\left(\frac{N}{13,51}\right)^{1,65}} .$$

Загальна кількість відмов по тілу – 445 (35%), по різьбі – 311 (24%), по муфті – 133 (10%), по штоку – 268 (21%), відгвинчування – 129 (10%) [2]. Спостерігається тенденція до зменшення загальної кількості відмов за один рік – з 165 (1999 рік) до 110 (2007 рік), що в основному спричинено зменшенням кількості відмов по тілу – з 93 (1999 рік) до 41 (2007 рік). Отримані також залежності частоти обривів від відносної глибини обриву h ШН у 2006 і 2007 роках [2]. Найбільше обривів відбувається внизу колони, дещо менше вверху і найменше для $h=0,5..0,7$. Кількість відмов вверху колони за ці роки зменшилась.

Як вказують дослідники [136, 143] такі моделі найбільш успішно можуть використовуватись тільки для свердловин, які подібні за складом продукції, обводненістю, СПУ, типом обладнання, режимами експлуатації та іншими параметрами на свердловини, які використовувались для побудови моделей. Були

спроби створення узагальнених моделей [131, 136], але вони враховують небагато факторів і володіють низькою точністю. Тому необхідно запропонувати саму методику побудови більш точних статистичних моделей. Сьогодні збільшення швидкодії обчислювальної техніки дозволяє застосувати нові методи побудови більш складних, точних і робастних статистичних моделей, зокрема індуктивні методи самоорганізації моделей [144] та інші методи машинного навчання [145], які дозволяють враховувати багато факторів.

1.5 Проблеми імітаційних моделей ШСНУ

1.5.1 Огляд математичних моделей ШСНУ

Математичні моделі ШСНУ дозволяють підвищити ефективність їхнього проектування та експлуатації. Побудова адекватної моделі ШСНУ є складним завданням, оскільки на її роботу впливає велика кількість різноманітних чинників, які іноді важко описати математично. Зі складністю математичних моделей ШСНУ зростає складність їхньої реалізації на комп'ютері, а також складність їхньої модифікації та удосконалення. Основною проблемою застосування існуючих адекватних моделей ШСНУ є складність їхньої модифікації під час удосконалення чи внесення інших змін. Це перешкоджає еволюції цих моделей.

Найпростішими математичними моделями ШСНУ можна вважати формули для розрахунку екстремальних навантажень на колону. Найчастіше використовуються формули К. Мілса [139], Д. Слоннеджера, А. С. Вірновського [97, 131], стандарту АНІ PR11L [146]. Ці формули не дають уявлення про рух компонентів ШСНУ, але їх можна використовувати для попередньої перевірки правильності побудови більш складних моделей.

Адекватність динамічних моделей ШСНУ, у першу чергу, залежить від механічної моделі руху колони ШН та гідродинамічної моделі насоса. У більшості випадків менш важливими є модель приводу ШСНУ, гідродинамічна модель руху рідини в НКТ, модель руху пружної НКТ. Найбільш адекватні динамічні моделі колони ШН як пружного стержня з розподіленою масою будують на основі

хвильового рівняння – лінійного гіперболічного рівняння з частинними похідними другого порядку [147-155]. Але застосовують також системи рівнянь першого порядку [156]. Разом з поздовжніми коливаннями колони можна моделювати і її крутильні коливання [157]. Моделювати колону ШН можна також як пружний стержень з зосередженою масою за допомогою звичайних диференціальних рівнянь другого порядку [23, 158-160]. Моделювання гідродинаміки насоса можливе за допомогою звичайних диференціальних рівнянь першого порядку [161]. Усі ці рівняння розв’язуються чисельними методами і навіть були запропоновані їхні аналітичні розв’язки. Але основною проблемою застосування таких моделей є складність їхньої зміни і перебудови під час додання в модель нових можливостей чи модифікації її частин. Не кожен спеціаліст в галузі нафтовидобування володіє необхідними для модифікації моделі математичними знаннями. Це суттєво ускладнює застосування і розвиток таких моделей. Одним з шляхів вирішення цієї проблеми є застосування компонентно-орієнтованого підходу до побудови моделей, зокрема за допомогою мови Modelica [162] та основаних на цій мові середовищ моделювання [163]. Тому необхідна розробка принципів побудови адекватних компонентно-орієнтованих моделей ШСНУ з можливістю простої їхньої модифікації та удосконалення.

1.5.2 Огляд методів компонентно-орієнтованого моделювання

Імітаційне компонентно-орієнтоване моделювання ґрунтується на побудові моделі складної системи з простих компонентів. Компонент описує математичну модель відповідного фізичного об’єкта (маса, пружина, електричний опір, гідравлічний опір, гідравлічний мотор та ін.), яка формулюється у вигляді алгебраїчного, диференціального чи різницевого рівняння. Компоненти з’єднуються один з одним через порти, які визначають множину змінних для взаємодії між компонентами [164]. Компоненти і порти зберігаються в програмних бібліотеках. Під мультидоменним моделюванням розуміють можливість використання в моделі компонентів, які відрізняються своєю

фізичною природою (механічних, гідравлічних, електричних і т. п.). Адекватне моделювання ШСНУ вимагає застосування компонентів як мінімум з двох областей – механіки та гідравліки. Компонентно-орієнтоване моделювання може базуватись на причинному моделюванні (Causal Modeling) або непричинному моделюванні (Acausal Modeling) [165]. У першому випадку компонент отримує на вході сигнал x , виконує на ним визначену математичну операцію $f(x)$ і повертає результат y на вихід. У цьому випадку моделювання реалізується імперативним програмуванням шляхом операції присвоювання змінній y значення виразу $f(x)$. У другому випадку сигнал y двох з'єднаних компонентів може передаватись у двох напрямках. Таке моделювання реалізується декларативним програмуванням шляхом розв'язування рівняння $y=f(x)$, де невідомим може бути x або y . Тут змінні x і y являють собою певні фізичні величини, а рівняння $y=f(x)$ – фізичний закон, який описує їхній зв'язок. Це дозволяє спростити побудову моделі, зосередитись на фізичному формулюванні задачі, а не на розроблянні алгоритму її розв'язання. Можливо також уникнути помилок, які характерні для імперативного програмування. Найчастіше поведінка цих моделей описується системою диференціальних рівнянь, які розв'язуються методом скінченних різниць – чисельним методом, оснований на заміні диференціальних операторів різницевиими схемами. Розв'язування нестационарних задач методом скінченних різниць являє собою ітераційний процес – на кожній ітерації знаходиться рішення стаціонарної задачі для заданого моменту часу. Для цього використовуються явні та неявні різницеві схеми [163, 166].

Найбільш відомими засобами імітаційного моделювання складних динамічних мультидоменних систем можна назвати Simscape для Simulink, Dymola, EcosimPro [164], 20-sim, Modelica. Серед них Modelica є найбільш популярною вільною мовою компонентно-орієнтованого моделювання таких систем [162, 166, 167]. Основні її особливості: вільна, об'єктно-орієнтована, декларативна, орієнтована на гібридне (неперервне і дискретне) компонентно-орієнтоване моделювання складних мультидоменних фізичних систем, підтримує побудову ієрархічних моделей, адаптована для візуального програмування, широко використовується для досліджень у різних галузях. Її вільна стандартна бібліотека (Modelica Standard Library) налічує близько 1280 компонентів. Існують

вільні та комерційні середовища моделювання мовою Modelica – OpenModelica [163], JModelica.org, Wolfram SystemModeler, SimulationX, MapleSim [165], Dymola, LMS Imagine.Lab AMESim. Вони містять засоби полегшення побудови та аналізу моделей, такі як візуалізація результатів, помічники для створення нових компонентів, оптимізація параметрів моделі, генерація захищеного виконуваного коду моделі, інтерфейси з іншими програмними продуктами, засоби символьних обчислень. Усі вони дозволяють побудову моделей за допомогою графічних блок-схем, яка полягає у перетягуванні потрібних компонентів з бібліотеки на графічну область, введення значень їхніх атрибутів і з'єднання їхніх портів. Наприклад середовище імітаційного моделювання мовою Modelica MapleSim [165], ефективно взаємодіє з системою комп'ютерної алгебри Maple, що дозволяє поєднувати переваги чисельних та символьних обчислень.

Основними недоліками мови Modelica є її вузька спеціалізація і складність вивчення. Проблему вузької спеціалізації можливо розв'язати шляхом застосування інтерфейсів між Modelica і мовами загального призначення [163, 166]. Але опанувати нову мову зазвичай важче, аніж вивчити компонент чи бібліотеку знайомої мови програмування. Цих проблем у користувача не виникає, якщо для побудови моделі він застосовує знайому йому мову програмування загального призначення. Чимало високорівневих мов загального призначення придатні для реалізації компонентно-орієнтованого моделювання, оскільки вони мають зручні імперативні та об'єктно-орієнтовані конструкції, а також дозволяють декларативне програмування [168]. Зокрема подібну на Modelica функціональність можна реалізувати на Python з її математичними бібліотеками SymPy [169] і SciPy [170].

1.5.3 Можливості імітаційного моделювання на основі абстрактних автоматів

Ще одним шляхом вирішення проблеми складності побудови моделі ШСНУ є застосування імітаційних моделей на основі абстрактних автоматів та автоматного програмування для їхньої реалізації. Як відомо, комп'ютерне імітаційне моделювання застосовують у випадку складності побудови аналітичної

моделі. Можна виділити два підходи до імітаційного моделювання складних динамічних систем – на основі моделей типу "чорний ящик" і моделей типу "білий ящик". Моделі типу "чорний ящик" (функціональні) відображають тільки зовнішнє функціонування об'єкта. Динаміка функціонування визначається глобальними правилами і законами. Це створює труднощі моделювання певних локальних явищ. Моделі типу "білий ящик" (структурні) відображають об'єкт як систему з певною структурою і механізмом взаємодії елементів. Динаміка функціонування визначається локальними правилами і законами, які можна задати окремо для кожного елемента. Тому такі моделі будуються "знизу вверху". Сьогодні такий напрямок імітаційного моделювання відомий як агентне моделювання, яке включає моделювання на основі автоматів [171-173].

Абстрактним автоматом називають математичну модель пристрою, який має множини входів, виходів та внутрішніх станів і здатний переходити в новий стан та генерувати сигнали на виходах в залежності від значення вхідних сигналів та поточного стану автомата. В дискретних автоматах час розбитий на частини однакової тривалості, які називають тактами. Протягом такту стан, вхідні та вихідні сигнали не змінюються. Скінченні автомати володіють скінченними множинами станів і вхідних значень [174]. Основною перевагою імітаційного моделювання на основі автоматів є можливість моделювання різноманітних складних систем шляхом опису простих правил поведінки їхніх елементів. Тому існує думка, що використання автоматів може бути універсальним способом моделювання динамічних систем [175, 176]. Моделювання на основі автоматів дає змогу також полегшити розробку алгоритму і програми, які його реалізують. Таку програму краще створити в стилі автоматного програмування. Автоматне програмування – це парадигма програмування, під час використання якої програма чи її фрагмент формуються як модель якого-небудь формального автомату. На відміну від класичного структурного програмування автоматне програмування дозволяє сформувати у розробника цілісну картину поведінки сутності та просто описувати системи зі складною поведінкою. Легко також організувати паралельні обчислення. Автоматне програмування може

використовуватись для полегшення розробки будь-яких програм [177]. Автоматне програмування може бути легко реалізоване на основі об'єктно-орієнтованого програмування. Тоді модель автомата може бути описана класом, стан автомата – полями (атрибутами) класу, функція переходу – методом класу. Нові автомати можна легко створювати шляхом успадкування класів інших автоматів. Програмування ще більше спрощується, якщо опис поведінки автомата виконується декларативним методом, а не імперативним. Тобто вказується ціль автомата, а не алгоритм досягнення цієї цілі. Таким чином необхідна розробка принципів побудови імітаційної моделі ШСНУ на основі абстрактних автоматів та програми об'єктно-орієнтованою мовою Python для її реалізації. Необхідно, щоб розроблену модель можна було легко модифікувати, а програма повинна бути виконана в стилі автоматного програмування і використовувати тільки стандартну бібліотеку Python.

1.6 Огляд можливостей систем CAD/FEA для автоматизованого проектування обладнання ШСНУ

1.6.1 Інтеграція CAD і FEA для проектування обладнання ШСНУ

Системи автоматизованого проектування (САПР, CAD-системи) застосовувались і є перспективними для дослідження і проектування обладнання ШСНУ: РЗ штанг [20, 57, 72, 178] та труб [7, 91, 92, 94], з'єднань склопластикових штанг [1], штанг з тріщинами [1], труб з тріщинами і БС [179], протекторів для ШН [63], деталей штангообертача [35, 180], зубчастих передач редуктора [181-183]. Популярними є такі комерційні CAD-системи як CATIA, NX, Creo, SOLIDWORKS, Inventor, Компас 3D та вільні FreeCAD, SALOME, OpenSCAD. SOLIDWORKS є однією з найпопулярніших САПР в загальному машинобудуванні, вона легка в застосуванні, має широкі можливості створення параметричних моделей та містить багато додаткових модулів, зокрема Simulation, який реалізує МСЕ для задач теорії пружності та пластичності. Поєднання CAD з комп'ютеризованою системою скінченно-елементного аналізу

(FEA) дає широкі можливості дослідження та параметричної оптимізації конструкції. FEA реалізує MCE – ефективний чисельний метод розв’язування рівнянь математичної фізики (механіки деформівного твердого тіла, електромагнетизму, гідрогазодинаміки та термодинаміки). На практиці MCE може бути ефективно реалізованим програмою, в якій автоматизуються всі етапи розв’язування задачі, починаючи з формування сітки скінченних елементів, закінчуючи обчисленням напружень, деформацій та інших величин. Відомі комерційні (Ansys, Nastran, Abaqus, Comsol, SOLIDWORKS Simulation) та вільні (CalculiX, FEniCS, Code_Aster, OpenFoam, Z88, Elmer) FEA-системи.

САПР часто мають інтерфейси прикладного програмування (API), які використовують для створення прикладних програм. Так SOLIDWORKS API застосовує об’єктно-орієнтований підхід і включає багато програмних компонентів для роботи з прикладною програмою, моделлю, ескізом, кресленням, елементами, інтерфейсом користувача та ін. Системи CAD і FEA можуть взаємодіяти шляхом обміну моделями в нейтральних форматах даних (IGES, STEP) або за допомогою API. Так Abaqus API може забезпечити доступ до SOLIDWORKS API, використовуючи модель компонентного об’єкта COM. Існують також повністю інтегровані з CAD модулі FEA (наприклад SOLIDWORKS/Simulation) або асоціативні інтерфейси FEA/CAD.

Abaqus/CAE – система FEA [184, 185] для задач механіки суцільних середовищ. Abaqus дає змогу розв’язувати статичні та динамічні задачі, а також сильно нелінійні перехідні швидкоплинні динамічні задачі. Має зручний інтерфейс користувача, підтримує велику кількість типів скінченних елементів, дає змогу розв’язувати контактні та інші нелінійні задачі, має інтерфейси для відомих CAD, модулі для розв’язування вузькоспеціалізованих задач, наприклад механіки руйнування. Основною перевагою, яка відрізняє його від інших програм, є наявність API популярною мовою Python. Python API надає Abaqus широкі можливості ефективної автоматизації роботи і створення прикладних програм для побудови СЕМ, їхнього аналізу і розв’язування оптимізаційних задач. Розробка і аналіз моделей в системах FEA, як правило, виконується в такій послідовності:

побудова геометричної моделі; введення характеристик матеріалів, кроків навантаження, граничних умов, параметрів контакту і навантажень; побудова сітки скінченних елементів; обчислення і аналіз результатів [186]. Розробляти і аналізувати модель в Abaqus можна вручну і автоматизовано за допомогою Python програм, що дозволяє створювати моделі з можливістю легкої зміни будь-якого її параметра (параметричні моделі). В Abaqus API об'єкти програми (геометричні моделі деталей, матеріали, збирання, кроки навантаження, контактні взаємодії, зовнішні навантаження, мережа скінченних елементів, двовимірні ескізи тощо) описані у вигляді класів та їхніх ієрархій мовою Python, що знаходяться в модулях `part`, `material`, `section`, `assembly`, `step`, `interaction`, `load`, `mesh`, `job`, `sketch`, `visualization` та інших.

Висока ціна комерційних систем САПР робить їх недоступними для багатьох потенційних користувачів. На даний час розроблено чимало якісних вільних програмних продуктів, які можна використати для побудови спеціалізованих систем CAD/FEA. Зазвичай вони програють комерційним аналогам в можливостях і якості документації. Однак інтеграція сучасного різнотипного вільного програмного забезпечення дозволяє розробляти ефективні спеціалізовані системи CAD/FEA, які багато в чому можуть конкурувати з комерційними аналогами. В працях [187, 188] описані принципи побудови прикладних САПР шляхом інтеграції таких вільних продуктів як Python 2.7, Open CASCADE Technology 6.8.0, FreeCAD 0.16, pythonOCC 0.16, Gmsh 2.11, CalculiX 2.9. Ці програми володіють вільним ліцензіями, сумісними з GNU GPL, доступні для операційних систем Linux та Windows та активно розвиваються.

Open CASCADE Technology (OCCT) – платформа розробки програмного забезпечення, яка забезпечує сервіси для 3D поверхневого і твердотільного моделювання, обміну даними з CAD та візуалізації [189]. OCCT використовує метод подання форми шляхом опису її границь (граничного подання або BREP). Форком (відгалуженням) OCCT є C++ бібліотека Open CASCADE Community Edition (OCE). Програмісти мовою Python можуть використовувати бібліотеку `pythonOCC` [190, 191], яка побудована на основі OCE за допомогою SWIG –

інструмента для пов'язування коду C++ та Python. FreeCAD 0.17 [192] – це вільна параметрична 3D САПР, яка базується на геометричному ядрі OCCT 7.2.0 і володіє API мовою Python. В працях [193, 194] показані можливості FreeCAD для швидкої розробки прикладних програм, що працюють з геометричними моделями. Прямим зв'язком FreeCAD з OCCT є її Python-модуль Part, використання якого набагато зручніше ніж pythonOCC. В праці [194] наведено абстрактний приклад використання модуля Part в FreeCAD 0.17x64. Gmsh – вільний генератор скінченно-елементної сітки з вбудованими можливостями пре-і постпроцесора [195]. Gmsh може працювати в інтерактивному режимі з графічним інтерфейсом користувача і в режимі командного рядка. CalculiX – популярний вільний програмний пакет для FEA. Спектр задач, які дозволяє розв'язувати CalculiX, надзвичайно широкий. Дозволяє розв'язувати лінійні та нелінійні задачі механіки суцільних середовищ МСЕ [186]. Можна моделювати анізотропні матеріали, розв'язувати різні типи контактних і динамічних задач. Основним компонентом CalculiX є розв'язувач CCX, який використовує формат вхідних файлів, подібний на формат вхідних файлів Abaqus.

Для досягнення системного ефекту під час дослідження і проектування обладнання ШСНУ важливим є можливість інтеграції в єдиній інформаційній системі різнотипних моделей і прикладних САПР, побудованих за допомогою вказаних CAD/FEA.

1.6.2 Аналіз можливостей систем CAD/FEA для моделювання різьбових з'єднань

Для дослідження РЗ ШСНУ системи CAD/FEA уможливають розв'язання таких задач [196]:

а) оптимізація геометричних параметрів РЗ (оптимізація профілю різьби [57], довжини згвинчування, довжини і форми розвантажувальних канавок [72, 178], радіусів скруглення, фасок), оптимізація нових конструкцій (зі вставним

витком, муфтою розтягу-стиску [1, 70], спеціальним профілем), обґрунтування допусків РЗ [10, 197];

б) оптимізація матеріалів РЗ, в тому числі характеристик пластичності [12] і ортотропії матеріалу (оптимізація відношення модуля пружності ніпеля до модуля пружності муфти, оптимізація залишкових напружень різьби ніпеля та матеріалу покриття, оптимізація технології формування різьби, оптимізація РЗ з композиційних матеріалів); дослідження коефіцієнта тертя [92];

в) оптимізація зусилля згвинчування [198];

г) обчислення довговічності РЗ з втомною тріщиною та іншими дефектами.

Ці оптимізаційні обчислення можуть виконуватись за критеріями: статичної та втомної міцності РЗ [57], зменшення концентрації напружень в різьбі, рівномірності навантаження вздовж витків різьби [1], величини навантаження на перші витки, критеріями механіки руйнування, міцності до ударних навантажень; зносостійкості, захисту від механічного спрацювання різьби та інших поверхонь РЗ [1]; зусилля згвинчування, вібрацій і динамічних навантажень, які призводять до самовідгвинчування або заїдання РЗ [15]; герметичності РЗ [93].

З метою аналізу і оптимізації РЗ системи FEA уможливають розв'язання задач: статичних, обчислення власних частот, динамічних і дуже нелінійних перехідних динамічних задач. Моделі матеріалів РЗ можуть бути пружними (лінійними) [7] і пружно-пластичними (нелінійними) [15]. Моделі контакту поверхонь РЗ можуть бути спрощені (лінійні) і реалістичні (нелінійні). Моделі РЗ в системі CAD/FEA можна поділити на тривимірні, плоскі та осесиметричні (табл. 1.2) [93]. У тривимірній моделі МСЕ розв'язується просторова задача, а в плоскій і осесиметричній – двовимірною. Тривимірною моделлю є більш реалістичною, наприклад дає змогу моделювати асиметричні навантаження і геометрію [1, 80]. Деколи для спрощення тривимірної моделі замість гвинтової поверхні різьби використовується квазірізьба – поверхня обертання. Плоска і осесиметрична моделі потребують меншої кількості скінченних елементів, тому володіють меншою обчислювальною трудомісткістю і дають змогу створювати сітку з дрібним кроком, що значно підвищує точність [57].

Таблиця 1.2 – СЕМ різьбових з'єднань

Характеристики	Моделі РЗ			
	тривимірні		двовимірні	
	гвинтова різьба	квазірізьба	плоскі	осесиметричні
Реалістичність	дуже висока	висока	середня	низька
Моделі навантажень	будь-які	окрім тангенціальних	в площині	симетричні відносно осі
Відношення обчислювальна трудомісткість/точність	велике	велике	мале	найменше

Спосіб моделювання згвинчування РЗ залежить від мети аналізу, виду самого РЗ, моделі (тривимірна чи осесиметрична) і можливостей CAD/FEA системи. Моделювати згвинчування можна, наприклад:

а) моментом згвинчування в тривимірній моделі з гвинтовою різьбою. Рекомендований момент згвинчування може бути обчислений за відомою формулою:

$$M \approx 0,5Fd_2 \left(\frac{p_p}{\pi d_2} + f / \cos(\alpha / 2) \right), \quad (1.1)$$

де: F – сила згвинчування: $F = \sigma \frac{\pi d_1^2}{4}$, $\sigma = 0,6\sigma_t$ – напруження в ніпелі, d_1 , d_2 – внутрішній і середній діаметр різьби, p_p – крок, α – кут профілю, f – коефіцієнт тертя.

б) зміщенням контактних поверхонь. Для цього виконується осьове видовження опорної поверхні РЗ, або зміщення різьбової поверхні на відстань, кратну кроку різьби [57]:

$$\Delta = \frac{p_p \cdot d_c}{\pi \cdot D_f}, \quad (1.2)$$

де d_c – колове зміщення муфти відносно штанги під час згвинчування, D_f – діаметр кола, на якому фіксується зміщення d_c .

в) болтовим навантаженням (bolt load), яке прикладається до поперечного перетину ніпеля (або муфти) та імітує силу затягування чи видовження ніпеля (скорочення муфти) Δ . Існує в багатьох системах FEA.

г) температурною деформацією Δ частини РЗ з заданим коефіцієнтом лінійного розширення в осьовому напрямку [1];

д) шляхом симуляції поля деформації згвинченого РЗ.

Для масштабного дослідження і оптимізації РЗ ШСНУ важливими є усі перелічені можливості систем FEA. Проте принципова схема такого дослідження включає ще й ітераційний процес: зміна параметричної моделі в CAD, експорт моделі в FEA, симуляція моделі в FEA. Тому під час вибору систем CAD/FEA, слід виходити з наявності в них інтерфейсів програмування або асоціативного інтерфейсу. Враховуючи це, ефективним є використання таких систем як SolidWorks/Abaqus, FreeCAD/CalculiX та мови їхньої системної інтеграції Python. Дослідники існуючих СЕМ РЗ нафтового обладнання [72, 76, 91] рідко акцентують увагу на можливостях їхньої параметризації, простої модифікації та інтеграції в іншу систему.

1.7 Огляд методів обчислення циклічної довговічності за результатами моделювання МСЕ

Найчастіше обчислення циклічної довговічності базується на експериментальних даних, які отримані для випадку одноосьового навантаження та симетричного циклу навантаження ($R=-1$) [199]. Тому застосовують різні методи переходу від асиметричного циклу навантажування до симетричного та від багатоосьового напружено-деформованого стану до одноосьового. Багато з них проаналізовано в працях [200-202].

Для переходу від асиметричного циклу навантажування до симетричного розраховують амплітуду напружень еквівалентного по пошкоджуваності симетричного циклу навантажування $\sigma_{a \text{ екв}}$. Тоді умова втомної міцності $\sigma_{a \text{ екв}} \leq \sigma_{-1}$. Для крихких матеріалів використовують лінійну залежність Гудмена. Для пластичних – нелінійну залежність Гербера або лінійну залежність Серенсена-Кінасшвілі [203] (в зарубіжній літературі відома як залежність Морроу), що має

експериментальне підтвердження для $-1 < R < 0 \dots 0,5$ та задовільна для більшості практичних випадків розрахунків на втомну міцність $\sigma_m < (0,5 \dots 0,6) \sigma_B$:

$$\sigma_a = \sigma_{-1} - \psi_\sigma \sigma_m, \quad (1.3)$$

де ψ_σ – коефіцієнт чутливості матеріалу до асиметрії циклу (mean stress sensitivity factor). Для сталей середньої міцності ($\sigma_B = 650-1150$ МПа) приймають $\psi_\sigma = 0,15-0,25$. В цій роботі автор використовував значення $0,33$. Для $\psi_\sigma = \sigma_{-1}/\sigma_B$ отримаємо залежність Гудмена. Цей коефіцієнт може бути отриманий експериментально:

$$\psi_\sigma = \frac{2\sigma_{-1} - \sigma_0}{\sigma_0}, \quad (1.4)$$

де σ_{-1} – границя витривалості для симетричного циклу.

σ_0 – границя витривалості для віднульового циклу.

Якщо підставити умову втомного руйнування $\sigma_{a \text{ екв}} = \sigma_{-1}$ в (1.3) то отримаємо:

$$\sigma_{a \text{ екв}} = \sigma_a + \psi_\sigma \sigma_m. \quad (1.5)$$

Після обчислення $\sigma_{a \text{ екв}}$ кількість циклів до втомного руйнування N може бути знайдена з рівняння кривої втоми, отриманої для $R = -1$:

$$\sigma_{a \text{ екв}} = AN^B,$$

де A та B – константи.

Часто використовують наступне рівняння кривої втоми [199]:

$$N = \frac{\sigma_R^m N_G}{\sigma^m}, \quad (1.6)$$

де N_G – абсциса точки перелому кривої втоми в логарифмічних координатах;

σ_R – ордината точки перелому (границя витривалості). Зазвичай відомо $\sigma_R = \sigma_{-1}$, тоді $\sigma = \sigma_{a \text{ екв}}$.

m – показник нахилу кривої втоми [199] $m = (\lg N_1 - \lg N_2) / (\lg \sigma_2 - \lg \sigma_1)$.

Запас втомної міцності (або коефіцієнт запасу) D визначає на скільки навантаження може бути збільшене, щоб досягти заданої циклічної довговічності [202]. Використовують запас втомної міцності за подібним циклом [204], якщо умова втомного руйнування має вигляд

$$D\sigma_a / K_D + \psi_\sigma D\sigma_m = \sigma_{-1},$$

де σ_a і σ_m – діючі напруження, а $D\sigma_a$ і $D\sigma_m$ – граничні напруження, K_d – коефіцієнт перерахунку границі витривалості, який враховує такі фактори як абсолютні розміри, шорсткість, корозійне середовище, поверхневе зміцнення та інші. Тоді

$$D = \frac{\sigma_{-1}}{\sigma_a / K_d + \psi_\sigma \sigma_m}. \quad (1.7)$$

Також використовують запас втомної міцності за змінними напруженнями [204], якщо умова втомного руйнування має вигляд

$$D\sigma_a / K_d + \psi_\sigma \sigma_m = \sigma_{-1},$$

де σ_a і σ_m – діючі напруження, а $D\sigma_a$ – гранична амплітуда напруження. Тоді

$$D = \frac{\sigma_{-1} - \psi_\sigma \sigma_m}{\sigma_a / K_d}. \quad (1.8)$$

Якщо $D < 1$, то деталь буде мати обмежену циклічну довговічність.

Для багатоосьового напруженого стану використовується формула (1.5), в якій [205]:

$$\sigma_a = \sqrt{\frac{1}{2} [(\sigma_{a1} - \sigma_{a2})^2 + (\sigma_{a2} - \sigma_{a3})^2 + (\sigma_{a3} - \sigma_{a1})^2]}, \quad (1.9)$$

$$\sigma_m = \sigma_{m1} + \sigma_{m2} + \sigma_{m3} = \sigma_{mx} + \sigma_{my} + \sigma_{mz},$$

де σ_{ai} – амплітуда компонента нормального напруження, $\sigma_{ai} = (\sigma_{i \max} - \sigma_{i \min})/2$;

σ_{mi} – середнє компонента нормального напруження, $\sigma_{mi} = (\sigma_{i \max} + \sigma_{i \min})/2$.

Допустимо також прийняти:

$$\sigma_m = \sqrt{\frac{1}{2} [(\sigma_{m1} - \sigma_{m2})^2 + (\sigma_{m2} - \sigma_{m3})^2 + (\sigma_{m3} - \sigma_{m1})^2]}. \quad (1.10)$$

Таким чином отримуємо залежність Сайнса [205]:

$$\sigma_{\text{екв}} = \sqrt{\frac{1}{2} [(\sigma_{a1} - \sigma_{a2})^2 + (\sigma_{a2} - \sigma_{a3})^2 + (\sigma_{a3} - \sigma_{a1})^2]} + \psi_\sigma (\sigma_{m1} + \sigma_{m2} + \sigma_{m3}).$$

Значення коефіцієнта запасу втомної міцності D за критерієм Сайнса може бути обчислене за формулою (1.8) [57]. Критерій Сайнса зручний для моделювання втоми МСЕ, але може бути використаний тільки для умов пропорційного навантажування, де напрямки осей головних напружень залишаються незмінними впродовж циклу навантажування [206]. Величину D можна застосовувати для відносного порівняння різних варіантів конструкції з

точки зору втомної міцності. Якщо для порівняння застосовується МСЕ, то важливо забезпечити однакові умови створення моделі, зокрема розмір сітки. Внаслідок неточності МСЕ та застосування пружних моделей матеріалів замість пружно-пластичних ця величина деколи може приймати від'ємне значення.

Більш точні та універсальні деформаційні критерії втомного руйнування під час багатоосьового навантажування базуються, в основному, на концепції критичної площини [201], оснований на пошуку біля поверхні деталі площини з найбільш небезпечним для утворення тріщини напружено-деформованим станом. В даний час важко віддати однозначну перевагу тому або іншому критерію [201]. Автор для обчислення циклічної довговічності N_f використовував критерій Брауна-Міллера [207], який дає найбільш реалістичні значення довговічності для пластичних металів, підходить для багатоосьового, пропорційного і непропорційного навантажування [201] і часто застосовується в таких програмних продуктах як fe-safe [202]:

$$\frac{\Delta\gamma_{\max}}{2} + \frac{\Delta\varepsilon_n}{2} = 1,65 \frac{\sigma_f'}{E} (2N_f)^b + 1,75\varepsilon_f' (2N_f)^c,$$

де $\frac{\Delta\gamma_{\max}}{2}$ - амплітуда максимальної деформації зсуву в критичній площині,

$\frac{\Delta\varepsilon_n}{2}$ - амплітуда нормальної деформації до площини з γ_{\max} ,

$E, \sigma_f', \varepsilon_f', b, c$ - константи матеріалу [201].

Рівняння Брауна-Міллера може бути модифіковане включенням корекції середнього напруження за Морроу [202]:

$$\frac{\Delta\gamma_{\max}}{2} + \frac{\Delta\varepsilon_n}{2} = 1,65 \frac{(\sigma_f' - \sigma_{nm})}{E} (2N_f)^b + 1,75\varepsilon_f' (2N_f)^c, \quad (1.11)$$

де σ_{nm} - середнє нормальне напруження до площини з γ_{\max} .

В даному випадку під циклічною довговічністю N_f слід розуміти кількість повторів повних циклів втомного навантажування до моменту утворення втомної тріщини, а під відмовою слід розуміти появу втомної тріщини [202]. Для аналізу багатоосьової втоми використовують метод масштабування і комбінування навантажування (multiaxial analysis using scale-and-combine loading) [202]. Він

полягає в тому, що тензор напружень для кожного набору напружень масштабується окремою історією навантажування і результати комбінуються, утворюючи історію напружень для комбінованого навантажування. В цій роботі обчислення втомної міцності за критерієм Брауна-Міллера виконували в fe-safe 6 – програмному комплексі для аналізу втоми матеріалів, який використовує результати моделювання напружено-деформованого стану МСЕ [202, 208].

Класичні випробування на втому не дають повної інформації для побудови закономірностей розвитку втомних тріщин. Для цього потрібно застосовувати методику прогнозування ресурсу на основі **лінійної механіки руйнування**. Вона полягає в обчисленні коефіцієнта інтенсивності напружень (КІН) та використанні діаграми втомного руйнування. Найпростіше значення КІН (для випадку відкриття тріщини $K = K_I$) визначається за допомогою прямих методів – шляхом визначення значення напруження (1.12) або деформації (1.13) в околі тріщини, наприклад за допомогою МСЕ [209, 210]:

$$K = \sigma_y \sqrt{2\pi r}, \quad (1.12)$$

$$K = \sqrt{2\pi} \frac{2G}{1+k} \frac{V_y}{\sqrt{r}}, \quad (1.13)$$

де G – модуль зсуву, Па; $G = E/(2+2\nu)$ (для сталі $G = 7,9 \cdot 10^{10}$ Па);

E – модуль пружності, Па (для сталі $E = 2,1 \cdot 10^{11}$ Па);

ν – коефіцієнт Пуассона (для сталі $\nu = 0,28$);

σ_y – напруження в околі тріщини, Па;

V_y – половина величини розкриття тріщини, м;

r – відстань від вершини тріщини до місця заміру σ_y або V_y , м ($r \rightarrow 0$);

$k = 3 - 4\nu$ – для плоского деформування;

$k = (3 - \nu)/(1 + \nu)$ – для плоского напруження.

Таким чином отримаємо формулу для прямого методу переміщень і плоского деформування [211]:

$$K = \frac{V_y E \sqrt{2\pi / r}}{4 - 4\nu^2}. \quad (1.14)$$

Значення напружень або переміщень, які необхідно підставити у формули, слід вибирати такими, щоб вони були визначальними для даного типу деформування [209]. Зазвичай метод переміщень є більш точним. Для підвищення точності обчислення КІН використовують дуже дрібні елементи в околі тріщини, субмоделі або екстраполують значення КІН для $r \rightarrow 0$ [209, 211]. Існують більш точні, але й більш складні методи обчислення КІН за допомогою МКЕ, наприклад метод J-інтегралу [209-211].

Після обчислення значень КІН знаходять відповідні значення поправочної функції $Y(a)$, що враховує вплив геометрії тріщини і деталі:

$$Y = \frac{K}{\sigma \sqrt{\pi a}}. \quad (1.15)$$

де a – глибина тріщини, м.

Функція $Y(a)$ дозволяє отримати емпіричну формулу для обчислення КІН в залежності від глибини тріщини a :

$$K_{теор} = Y \sigma \sqrt{\pi a}. \quad (1.16)$$

Перша ділянка діаграми втомного руйнування відповідає діапазону значень швидкості росту тріщин $V < 5 \cdot 10^{-9}$ м/цикл, а третя ділянка – $V > 10^{-6}$ м/цикл. Друга ділянка (середня) є прямолінійною в логарифмічних координатах і описується степеневою залежністю Періса:

$$V(\Delta K) = \frac{dl}{dN} = C(\Delta K)^n, \quad (1.17)$$

де C і n – постійні, що залежать від матеріалу.

$\Delta K = K_{\max} - K_{\min}$. Для віднульового циклу ($R=0$) $K_{\min} = 0$, тому $\Delta K = K_{\max}$.

Циклічну довговічність обчислюють шляхом числового інтегрування:

$$N = \int_{a_0}^{a_k} \frac{da}{V(\Delta K)} \approx \sum_{a_0}^{a_k} \frac{\Delta a}{V(\Delta K)_i}, \quad (1.18)$$

де N – довговічність деталі з врахуванням тріщиностійкості матеріалу;

a – довжина тріщини;

a_0, a_k – початковий і критичний (або кінцевий) розміри тріщини;

$V(\Delta K)$ – функція швидкості росту тріщини на повітрі або в корозійному середовищі.

Коефіцієнт запасу визначається так:

$$D = N / N_s, \quad (1.19)$$

де N_s – потрібне число циклів.

Ця методика дозволяє прогнозувати ресурс деталей з експлуатаційними дефектами (штанг, НКТ, їхніх РЗ) для різних умов навантаження без необхідності проведення натурних випробувань на корозійну втому.

1.8 Проблеми інформаційної системи підтримки життєвого циклу ШСНУ

1.8.1 Ізоморфізм закономірностей складних систем

Життєвий цикл (ЖЦ) виробу охоплює етапи його проектування, виготовлення, експлуатації та ремонту [212]. Для управління ЖЦ (product lifecycle management – PLM) виробу призначені **інформаційні системи підтримки життєвого циклу виробу (ІС)**. Вони складаються зі спеціалізованих елементів (інформаційних ресурсів), призначених для досягнення певних цілей на конкретних етапах ЖЦ виробу, і які можуть взаємодіяти один з одним і з середовищем через уніфіковані інформаційні канали [213]. ІС може мати різноманітні ресурси: лінгвістичні ресурси, бази даних, бази знань, прикладні програми, комп'ютерні моделі, апаратне забезпечення, користувачі. Інформаційною продукцією ІС можуть бути технічні вимоги, винаходи, математичні моделі виробу і його частин, конструкторська документація, технологічні процеси, рекомендації для експлуатації, стандарти. Зазвичай такі ІС базуються на концепції CALS-технологій (Continuous Acquisition and Life cycle Support – безперервної інформаційної підтримки процесів ЖЦ виробів) [214-216]. ШСНУ та їхні компоненти можуть поєднувати у собі тисячі наукоємних розробок, які поступово розроблялись і удосконалювались впродовж десятиліть. У зв'язку зі складністю предметної області ІС таких виробів, ці ІС потрібно

відносити до класу складних систем. Тоді, відповідно до принципу ізоморфізму [217], їм повинні бути властиві усі закономірності складних систем. Це треба враховувати під час побудови ІС та їхніх абстрактних моделей, які відображають найбільш загальні, як правило якісні, характеристики ІС. Відомі дослідження і концептуальні моделі таких систем [218-220] слабо акцентують увагу на цьому. Необхідна розробка абстрактної моделі ІС шляхом трансдисциплінарного огляду і аналізу основних загальносистемних закономірностей на прикладах різних природних та штучних складних систем.

Системою називають множину пов'язаних елементів. Відповідно класифікації систем К. Боулдінга **складна система** характеризується вищим проявом властивостей відкритості та стохастичності поведінки, більш яскраво вираженими проявами закономірностей ієрархічності та історичності, а також більш складними "механізмами" функціонування і розвитку [221]. **Закономірності складних систем** характеризують принципові особливості побудови, функціонування і розвитку складних систем [222]. Це закономірності взаємодії частини і цілого (емерджентність, адитивність, прогресуюча систематизація, прогресуюча факторизація, інтегративність), закономірності ієрархічної впорядкованості (ієрархічність та комунікативність), закономірності здійсненності систем (закон "необхідної різноманітності" У. Р. Ешбі, еквіфінальність, закономірність потенціальної ефективності Б. С. Флейшмана [223]), закономірності розвитку систем (історичність та самоорганізація). Один з основних принципів теорії систем **принцип ізоморфізму** [217, 224] (тотожність закономірностей систем) говорить, що ці закономірності характерні для усіх складних систем. Складним системам властива синергія усіх цих закономірностей. В праці [213] автором виконаний міждисциплінарний огляд і аналіз таких закономірностей в різних складних природних та штучних системах, який дозволить їх глибше дослідити та перенести в ще не вивчені системи, зокрема ІС.

Проведений аналіз показує, що багато складних систем володіють вказаними закономірностями, зокрема їхні ЖЦ мають подібну структуру та зміст

етапів. Результати аналізу дозволяють побудувати абстрактну модель ІС шляхом синтезу цих закономірностей. Філософською основою цих закономірностей є **діалектика**. Діалектична логіка доповнює закони класичної логіки законами: "єдності та боротьби протилежностей" (замість закону "виключення третього"), "переходу кількісних змін в якісні", та законом "заперечення заперечення". Відповідно першого закону діалектики будь-який предмет має протилежності, які в процесі взаємодії призводять до протиріччя, що дає поштовх розвитку. Протилежність – риси, сторони, ознаки предмета, які докорінно відрізняються один від одного і, разом з тим, не можуть існувати один без одного, взаємно доповнюють один одного. Діалектика стверджує, що проблема повинна бути поставлена в антиномічній формі, а принцип історизму повинен бути принципом будь-якого мислення [225].

Закономірність цілісності (емерджентність, системний ефект, синергія) – поява у системи властивостей, яких немає у її елементів. Цю закономірність часто називають основною системною проблемою. Цілісності характерна інтегративність – елементи повинні бути протилежними (неоднорідними, суперечливими, взаємодоповнюючими), як цього вимагає закон єдності та боротьби протилежностей. Наприклад, здатність систем до розвитку часто пояснюють наявністю в них протилежних додатного і від'ємного зворотних зв'язків [226, 227]. Перший направлений на зміни, а другий – на стабільність. Емерджентними є основні фактори еволюції [228], психіка [229, 230], типи особистості [231, 232]. В проектуванні систем виділяють два основні протилежні підходи: підхід "зверху" та підхід "знизу". Підхід "зверху" (аксіологічний, ціленаправлений, цільовий, декомпозиції, структуризації) описує систему в термінах цілей і цільових функціоналів. Підхід "знизу" (каузальний, термінальний, композиції, морфологічний, тезаурусний) описує систему в термінах впливу одних елементів на інші, у вигляді "простору станів". Ці підходи доповнюють один одного та існують рекомендації застосовувати їх паралельно [222].

Закономірність адитивності (нецілісності, дисинергії) описує відокремленість елементів системи. **Закономірність прогресуючої систематизації** описує прагнення системи до цілісності, а **закономірність прогресуючої факторизації** описує прагнення системи до адитивності. Будь-яка система, що розвивається, знаходиться між станами цілісності і адитивності [222].

Для складних систем характерна **закономірність ієрархічності** – наявність в системі упорядкованих ієрархій і прояви цілісності рівнів ієрархії. Ієрархії систем відомі в біології, менеджменті, інформатиці та психології [229, 233]. Складну проблему можна розбити на прості, а прості – на ще простіші. Так можливі описи системи з точки зору різних рівнів абстрагування (страт) та рівнів складності рішення (шарів) [234]. На вищих рівнях абстрагування система подається як "чорний ящик" з цілями заданими зовні, а на нижніх – як "білий ящик" з цілями заданими зсередини. З ієрархічністю тісно пов'язана **закономірність комунікативності** – з'єднання системи з середовищем множиною комунікацій.

Закономірності розвитку систем ще не достатньо вивчені сьогодні, але закони діалектики можуть бути філософською базою для таких досліджень. Кожна система та її елемент володіє "життєвим циклом" – періодом часу від їхнього зародження до смерті. Система невіддільна від свого ЖЦ, тому допустимо говорити про єдність системи та її ЖЦ. Очевидно, що в будь-якому циклі, зокрема в циклі цілеспрямованої діяльності, можна виділити протилежні в часі етапи. ЖЦ системи можна розглядати як систему з елементами (етапи, стани) і відношеннями між ними. Відповідно і сама складна система повинна містити елементи, які спеціалізуються на цих етапах. Багатьом складним системам властива **закономірність історичності** [222], яка не тільки вказує на наявність характерних етапів ЖЦ їхніх елементів, але й циклічних повторів цих етапів з переходом на рівень вищої якості. Такі коливальні явища часто спостерігаються в економічних, соціальних чи екологічних системах. Не зважаючи на надзвичайну розповсюдженість коливальних явищ в природі небагато з них вивчені та ще немає переконливої відповіді про глибинні причини цієї розповсюдженості [235],

але помітний їхній зв'язок з закономірністю цілісності [226] та наявністю протилежних елементів, які є їхньою причиною [213]. Зокрема відомі різні біологічні цикли, економічні цикли [236-238], цикли в соціальних системах [239, 240], цикли наукових парадигм [241, 242]. Фундаментальною передумовою тектології О. О. Богданова [226] є принципи емерджентності та історичності. Її можна розглядати також як є універсальну методологію рішення задач [243]. Важливим її поняттям є тектологічний акт, фази якого відповідають діалектичній тріаді (тезис, антитезис, синтез). Початок тектологічного акту – це завжди криза – криза з циклом її наслідків [226]. Наприклад, механізм еволюції – це тріада "мутації"- "природній добір"- "адаптація". Теорія пізнання виділяє взаємодоповнюючі рівні пізнання (теоретичний і емпіричний) та форми пізнання (раціональне пізнання, чуттєве та надчуттєве). Ці рівні та форми є несумісними в один момент часу і повинні послідовно змінювати один одного. Закономірність історичності та ієрархічності помітна в структурі діяльності суб'єкта [229]. В методології діяльність також розглядається як система [244]. Процеси розв'язання творчих задач і проблемних ситуацій можна розглядати як складні системи, які мають закономірність історичності. Вони залежить від багатьох психологічних факторів: мотиваційний стан, знання, інтелект, тип особистості. Існують різні взаємодоповнюючі стратегії розв'язання задачі: абстракція, аналогія, мозкова атака, аналіз, перевірка гіпотез, нестандартне мислення, аналіз засобів і цілей, метод фокальних об'єктів, морфологічний аналіз, доведення неможливості вирішити проблему, зведення до іншої задачі, дослідження, аналіз причин, метод спроб і помилок. Можна виділити таку загальну послідовність розв'язання задачі (цикл розв'язання задачі): проблемна ситуація – постановка задачі – рішення, або таку: пошук – мобілізація – застосування засобів [229]. Емерджентні стадії творчого мислення описували: Г. Уоллес, А. Пуанкаре, Г. Гельмгольц, Б. А. Лезин, К. Дункер, П. К. Енгельмейер, П. М. Якобсон, Я. О. Пономарьов, Ж. Адамар [245], Дж. Гіксон, А. Т. Шумілін, А. де Гроот [246], Г. С. Альтшуллер [247], О. О. Богданов [226]. Процеси і цикли прийняття рішень відомі в теорії прийняття рішень, теорії управління (цикл управління-виконання), системному

аналізі, методах управління проектами. Розроблено цикли прийняття рішення для різноманітних задач: цикл спотереження–гіпотеза–екперимент–оцінка в науковому методі; цикл Демінга PDCA (plan–do–check–act) та «спіраль якості» Джурана в системі управління якістю; підхід DMAIC (define, measure, analyze, improve, control) в методології "шість сігм" для управління виробництвом; метод "Шість Капелюхів Мислення" Едварда де Боно; метод "DO IT!" Р. Олсона; метод "How to Solve It" Джорджа Пойа; метод Eight Disciplines Problem Solving (Вісім дисциплін рішення проблем), розробленого в Ford Motor Company; метод OODA loop Джона Бойда (observe, orient, decide, act) в військовій справі; метод Intelligence–Design–Choice Герберта Саймона; методика Getting Things Done Девіда Алана (Collect–Process–Organize–Do–Review); метод АЗ, розроблений в Toyota. В праці [222] наведено порівняльний аналіз трактувань ЖЦ штучних і технічних систем – методики "ПАТТЕРН", методик Г. С. Поспелова, С. А. Саркисяна, М. М. Четвертакова, Е. Г. Яковенка, В. Н. Спицнаделя. Там же вказані приклади ЖЦ для різних видів продукції або послуг. Поняття ЖЦ штучних систем висвітлено в деяких стандартах [248-251]. У багатьох цих циклах спостерігається чергування двох протилежних емерджентних етапів (наприклад безсвідомого-свідомого, раціонального-ірраціонального, теоретичного-практичного). В програмній та системній інженерії відомі наступні моделі ЖЦ систем [251, 252]: каскадна, ітеративна, гнучка, V-модель, Dual Vee модель, спіральна модель. В праці [253] детально описані моделі ЖЦ сучасної системної інженерії. В одній з них перший етап "розробка концепції" з входами "функціональна недостатність" та "технічні можливості" має виходи "специфікація функціоналу" та "визначення концепції", які є входами другого етапу "технічна розробка". Виходами цього етапу є "специфікація виробництва" та "виробництво системи", які є входами третього етапу "пост-розробка". Його виходами є "робоча документація" та "установлена система". Кожний етап може розбиватись на три підетапи за такою ж схемою.

Закономірність самоорганізації – здатність систем виходити на якісно новий рівень розвитку, адаптуватись до зовнішніх умов. Цю закономірність

вивчає синергетика, яку часто позиціонують як універсальну теорію еволюції. Згідно синергетики нерівноважність системи є необхідною умовою її розвитку. Процеси самоорганізації відбуваються поряд з процесами протилежної направленості та можуть переважати над ними (прогрес) або уступати їм (регрес) [254].

Розглянемо приклади прояву **закономірності адитивності**. В психології відомі такі порушення цілісності психічних процесів як когнітивний дисонанс (конфлікт ідей, цінностей, емоцій), когнітивне упередження (порушення прийняття правильних рішень), психастенія (порушення рішучості), резонерство (порушення ціленаправленості мислення), інерція мислення (порушення уяви) [228, 229]. З точки зору теорії систем допустимо вважати, що ці порушення викликані адитивністю компонентів психіки – їхньою роздільністю, односторонністю, нецілісністю. Глибинна психологія вивчає два протилежні несвідомі потяги: "потяг до життя" і "потяг до смерті" [230]. Їм можна поставити у відповідність дві протилежні загальносистемні закономірності – "прогресуюча систематизація" і "прогресуюча факторизація". В теорії прийняття рішень відомі фактори, які не сумісні з певними етапами творчого мислення. Наприклад конформізм, ригідність, цензура та критика неприпустимі на етапі генерації ідей. Але, все ж, вони потрібні для цілісного мислення і повинні існувати в наступних етапах прийняття рішення. В практиці проектування систем відомі випадки, коли надмірна "зацикленість" на одному етапі проектування призводила по повної зупинки робіт [222]. Або навпаки – коли незавершеність кожного етапу призводила до помилкових проектних рішень.

Еквіфінальність – це динамічна властивість системи здійснювати рух різними шляхами з різних початкових станів в один кінцевий стан, який визначається тільки параметрами системи [217]. Будь-яка система має граничні можливості. Для їх подолання система повинна об'єднуватись з іншими системами. В. Ф. Турчин вважає, що еволюція пояснюється метасистемними переходами – інтеграцією систем в одне ціле з виникненням нового рівня управління [255].

Закономірність "необхідної різноманітності" говорить про необхідність різноманітності елементів системи. Наприклад, відповідно до різних психологічних типологій можна пояснити ефективність методу експертних оцінок, коли для вирішення проблеми залучаються багато спеціалістів навіть однієї спеціальності. Внаслідок різних типів вони мають різні погляди на проблему, збільшується «різноманітність» поглядів та методів і зростає імовірність емерджентності. А. І. Уемов [256] наголосив на необхідності "різноманітності" під час вирішення творчих проблем, чергування етапів виникнення ідей, їхньої розробки, конкретизації та втілення. У. Р. Ешбі довів, що різноманітність системи, яка управляє, повинна бути більшою різноманітності системи, яка управляється [257]. Потрібно збільшувати різноманітність елементів системи, яка управляє, (збільшувати кількість методів, залучати до роботи різних спеціалістів) і зменшувати різноманітність системи, що управляється (зменшувати невизначеність, уніфікувати, стандартизувати). Тоді зростає імовірність синергії елементів управляючої системи. Але, разом з тим, може зрости і диссинергія внаслідок неузгодження певних елементів. Різноманітність є необхідною умовою, але не достатньою. Тому ці елементи повинні бути організовані у систему з максимальним синергетичним ефектом, відповідно організовані в просторі та часі.

Системний аналіз (СА) – один з найбільш конструктивних напрямків системних досліджень і застосовується тоді, коли проблема стає дуже складною, є велика невизначеність проблемної ситуації та багатокритеріальність задачі [222]. СА спирається на теорію систем та використовує мультидисциплінарний підхід. Основним методом СА є розчленовування великої невизначеності на малі, зі збереженням цілісного уявлення про об'єкт дослідження і проблемну ситуацію [222]. Тому СА вимагає розробки методики системного аналізу, яка визначає впорядковані етапи і методи розв'язування задачі. Аналіз таких методик [222] дозволяє виділити в багатьох з них етапи виявлення проблеми і постановки цілей, етап розробки варіантів і моделей прийняття рішення, етап оцінки альтернатив і пошуку рішення, етап прийняття рішення, етап реалізації та оцінки реалізації.

Відповідно до принципу абстрагованого відображення системи Ю. І. Черняк [258] пропонує виділяти наступні рівні (страти) відображення системи: концептуальний, науково-дослідницький, проектний, інженерно-конструкторський, технологічний і рівень реалізації. Кожний з цих рівнів теж поділяється на етапи, на яких застосовуються відповідні методи. За рівнем формалізації методи СА можна поділити на методи активізації інтуїції спеціалістів (МАІС) і методи формалізованого представлення систем (МФПС) [222]. МАІС – це, в основному, якісні методи, такі як морфологічні, "мозкової атаки", експертних оцінок, дерева цілей, сценаріїв. МФПС – це, в основному, кількісні методи, такі як логічні, аналітичні, теоретико-множинні, статистичні. Ці методи можуть поєднуватись, на основі чого був розвинутий метод СА під назвою "поступова формалізація моделі прийняття рішення" [259]. Цей підхід базується на ідеї поступової формалізації задач (проблемних ситуацій) з невизначеністю шляхом почергового використання засобів МАІС і МФПС. Автори методу відмічають, що більшість реальних ситуацій проектування складних систем потрібно відобразити класом самоорганізуючих систем, моделі яких повинні постійно коректуватись і розвиватись. Рекомендують спочатку поділяти весь цикл на два великі етапи: формування моделі прийняття рішення, її оцінка та аналіз. На першому етапі часто спочатку застосовують МАІС, а потім – МФПС. На другому нерідко застосовують експертне оцінювання [222]. В СА важливе значення мають методики формулювання, структуризації та аналізу цілей, які ґрунтуються на закономірностях цілеутворення [222, 260]: 1 – уявлення про ціль залежить від стадії пізнання об'єкта; 2 – ціль залежить від внутрішніх і зовнішніх факторів; 3 – необхідно структуризувати цілі; 4 – спосіб подання цілей залежить від стадії пізнання об'єкта; 5 – в структурі цілей проявляється закономірність цілісності; 6 – для формування ієрархічних структур цілей застосовують підхід "зверху" та підхід "знизу"; 7 – цілі є засобами досягнення цілей вищого рівня і цілями для нижнього рівня; 8 – на нижніх рівнях цілі більш конкретні, а на верхніх більш розмиті; 9 – на одному рівні поділені цілі повинні бути співрозмірними і логічно незалежними, а ознаки декомпозиції єдиними.

Отже трансдисциплінарні дослідження загальносистемних закономірностей на прикладах різних природних та штучних складних систем повинні бути основою побудови таких ІС. У зв'язку з великою кількістю таких досліджень в самій ІС потрібно створити механізми автоматизованої реалізації результатів цих досліджень в алгоритмах ІС. Це можливе, для прикладу, шляхом організації загальнодоступної бази знань з цими результатами, яка наповнюватиметься дослідниками якою-небудь мовою опису систем.

1.8.2 Обґрунтування архітектури ІС, мови системної інтеграції та методів подання знань

Відповідно до закономірностей складних систем, зокрема закономірностей емерджентності та "необхідної різноманітності", ІС повинна бути гібридною системою, що поєднує у собі різні моделі інтелектуальних систем, які доповнюють одне одного. Великий список таких моделей наведено в праці [261]. Відомі технології синергетичного штучного інтелекту основані на синергетичному об'єднанні великої кількості різнотипних інтелектуальних елементів. Зокрема гібридні інтелектуальні системи об'єднують різні концепції подання і обробки знань [262], а багатоагентні системи основані на взаємодії великої кількості інтелектуальних агентів [263]. Такого роду ІС потрібно відносити до класу складних систем і, відповідно до принципу ізоморфізму, їм повинні бути властиві усі закономірності складних систем [213, 222]. В праці [262] наголошується на важливості самоорганізації в гібридних інтелектуальних системах. Найбільший вплив на самоорганізацію мають такі відношення між елементами системи як "координація", "узгодженість" і "спір" [262]. Аналіз цих відношень показав, що існує, як мінімум, два класи мікрорівневих моделей ІС за ознакою наявності координації між координатором і експертами, три класи за ступенем узгодженості експертів і два класи за ознакою наявності спору. Таким чином, може існувати не менше 12 різних класів мікрорівневих моделей ІС [262].

Під час самоорганізації ІС змінює свою мікрорівневу модель так, щоб залишатися релевантною задачам [262].

Щоб бути ізоморфною до складних систем, ІС повинна мати такі базові властивості: велика кількість гетерогенних елементів (інтелектуальних агентів), автономність елементів, децентралізація, взаємодія між елементами, прості локальні правила (поведінка) елемента. Цим вимогам відповідають такі програмні системи як мультиагентні системи (Multi-agent systems – MAS) [264, 265] та агентно-орієнтований підхід до їхньої розробки (agent-oriented programming) [266], який зазвичай реалізується ООП. MAS може спростити розробку і застосування ІС [267] а також, завдяки системному підходу, підвищити їхню ефективність. Для простого опису поведінки елементів перспективною є інтеграція в ІС систем, основаних на правилах (rules engine) [268], наприклад мови логічного програмування Datalog.

Для спрощення розробки системи необхідна проста, розповсюджена, високорівнева мова програмування загального призначення, яка орієнтована на інтеграцію різнотипних компонентів. Цим вимогам відповідають так звані "мови сценаріїв", зокрема Python [269]. Основні особливості Python: працює майже на усіх відомих платформах, є відкритим і вільним програмним забезпеченням, виконується шляхом інтерпретації байт-коду, має інтерактивний режим роботи, підтримує кілька парадигм програмування (в тому числі ООП), код програм компактний і легко читається, мові характерні динамічна типізація, повна інтроспекція, зручні структури даних (кортежі, списки, словники, множини), велика стандартна бібліотека та велика кількість сторонніх бібліотек різноманітного призначення. ООП основане на поданні програми у вигляді сукупності об'єктів (абстрактних моделей), які можуть взаємодіяти, і кожен з яких є екземпляром певного класу (типу даних), а класи є членами певної ієрархії успадкування [270]. Кожен клас описує множину об'єктів певного типу. Основними принципами ООП є абстрагування, інкапсуляція, успадкування і поліморфізм [270]. Python має здатність до системної інтеграції різнотипних компонентів та велику кількість пакетів, зокрема: NumPy (пакет для N-вимірних

масивів), SciPy (фундаментальна бібліотека для наукових обчислень [170]), Matplotlib (2D графіки [271]), SymPy (символьна математика) [169], scikit-learn (машинне навчання [272]), pyDatalog (логічне програмування [273]), pandas (аналіз даних [274]), NetworkX (дослідження складних графів) [275], pythonOCC. Інше ПЗ може бути доступне через API: Abaqus, FreeCAD, CalculiX, Gmsh. Така інтеграція може забезпечити емерджентність і "необхідну різноманітність". Застосування Python може також суттєво спростити розробку MAS [276]. Відомі такі MAS загального призначення для Python як osBrain [277] чи PADE [278].

Важливим компонентом ІС є база знань, яка містить множину фактів і правил логічного висновку (виведення). Відомо, що розвиток науки ґрунтується на отриманні об'єктивних і системно-організованих знань, які допомагають людям раціонально організувати свою діяльність і вирішувати різні проблеми, котрі виникають в її процесі. З розвитком науки об'єм знань у будь-якій галузі діяльності людини поступово збільшується. Разом з тим виникає багато проблем, пов'язаних з їхнім ефективним використанням, наприклад пошуком, коригуванням, аналізом, систематизацією, узагальненням. Комп'ютерні технології зумовили розвиток інженерії знань – області науки про штучний інтелект, яка вивчає методи і засоби отримання, представлення, структурування і використання знань. Інженерія знань пов'язана з розробкою баз знань і експертних систем [279, 280]. База знань – це сукупність фактів і правил логічного висновку (виведення) в обраній предметній області. Правила логічного висновку – це правила перетворення вихідної системи фактів (суджень) в нову систему фактів (висновків). Наявність правил висновку є основною відмінністю баз знань від баз даних. Програма, яка виконує логічний висновок з попередньо побудованої бази фактів і правил у відповідності з законами формальної логіки, називається машиною виведення. База знань і машина виведення є основними компонентами експертної системи – програми, яка призначена для пошуку способів вирішення проблем з певної предметної області. Для такого пошуку користувач подає відповідні запити до бази знань. Для збереження знань в комп'ютерній системі з можливістю їхньої подальшої автоматизованої обробки використовують

різноманітні методи подання (представлення) знань. Прикладами методів подання чітких знань є: логіка першого порядку, мови програмування Prolog/Datalog, реляційні системи, продукційна модель, фреймова модель, семантичні мережі [279-282]. До методів подання нечітких знань відносяться: нечітка логіка і теорія нечітких множин, нейронні мережі, еволюційне моделювання, гібридні інтелектуальні системи [261, 281, 282]. Методи подання знань можна реалізувати різними формальними мовами подання знань, наприклад класу онтологій. Під онтологією в інформатиці розуміють детальну формалізацію деякої області знань за допомогою концептуальної схеми. Онтологія може використовуватись для представлення в базі знань ієрархії понять та їхніх відносин [281]. Семантичною мережею називають інформаційну модель предметної області, що має вигляд орієнтованого графа. Вершинами графа є об'єкти предметної області (поняття, події, властивості, процеси), а дугами графа – відносини між об'єктами [279, 281]. Мова Веб-Онтологій OWL (ontology web language) [283] була розроблена для реалізації концепції семантичної павутини в межах всесвітньої павутини (WWW) і є розширенням мови RDF (Resource Description Framework) [284]– мови розмітки на основі XML, яка використовується для подання тверджень про ресурси в вигляді придатному для машинної обробки. Твердження, що висловлюється про ресурс RDF, має вигляд «суб'єкт – предикат – об'єкт» і називається триплетом. Наприклад: «циклічне навантаження» – «є причиною» – «втоми». Множина RDF-тверджень утворює орієнтований граф, в якому вершини є суб'єктами та об'єктами, а ребра – предикатами. Діалекти OWL Lite і OWL DL основані на описових (дескрипційних) логіках – мовах подання знань, що дозволяють описувати поняття предметної області в недвозначному, формалізованому вигляді [285]. Для полегшення розробки бази знань використовують редактори онтологій, наприклад Protégé [286]. Для створення запитів до даних, поданих за моделлю RDF, використовується мова запитів SPARQL. SPARQL – це мова запитів до даних, поданих за моделлю RDF, а також протокол для передавання цих запитів і відповідей на них [287]. *SWRL (Semantic Web Rule Language)* [288]– це рекомендація мови правил для семантичної павутини, яка ґрунтується на

об'єднанні OWL (OWL Lite і OWL DL) з унарною/бінарною Datalog RuleML. *Datalog* – це мова правил і запитів для дедуктивних баз даних, яка синтаксично є підмножиною мови Пролог. *RuleML (Rule Markup Language)* – це мова розмітки на основі XML для опису правил. SWRL розширює OWL аксіоми правилами на основі диз'юнктивів Хорна. Правила SWRL є формою імплікації антецедент (засновок) \rightarrow консеквент (висновок). Антецедент і консеквент складаються з 0 або більше атомів (предикатів) об'єднаних кон'юнкцією (логічним AND): атом \wedge атом ... \rightarrow атом \wedge атом. Атоми SWRL мають бути одномісними або двомісними. Атоми можуть бути визначені як $C(x)$, $P(x,y)$, $\text{sameAs}(x,y)$, $\text{differentFrom}(x,y)$ або $\text{builtIn}(r,x,\dots)$, де C – OWL опис або діапазон даних; P – OWL властивість; r – вбудоване відношення; x,y – OWL індивіди, OWL значення даних або відповідні змінні. Правила SWRL можуть бути включені в OWL онтологію. Не зважаючи на усі переваги OWL, SPARQL і SWRL є спеціальними мовами, що певною мірою обмежує їхні можливості та розвиток. Для простої інтеграції бази знань в ІС бажано її розробляти тією ж мовою, що і ІС.

Основною ідеєю автора є застосування в якості мови подання знань та створення запитів до бази знань мови програмування Python, яка на відміну від спеціальних мов є мовою загального призначення, і тому дає розробнику експертних систем набагато більше можливостей і дозволяє реалізувати концепцію гібридної системи. Для цього можна використати засоби ООП Python. Автором запропоновано принципи розробки баз знань мовою Python на основі фреймової моделі подання знань [289]. Елементи розробленої автором онтології частково відповідають основним елементам OWL Lite [283]: класи (з можливістю створення підкласів), властивості (які можуть бути функціональними, інверсними, симетричними і транзитивними) та індивіди. Тому можливим є частковий імпорт і експорт таких онтологій в OWL. Модуль Python, який містить базу знань і запити до неї повинен мати блочну структуру – спочатку створюються класи, потім індивіди, потім задаються властивості індивідів, а в кінці виконуються запити. Класи онтології відповідають класам мови Python, а підкласи можна створювати за допомогою механізму успадкування Python. Індивіди онтології відповідають

об'єктам Python. Індивіди можуть мати властивості у вигляді атрибутів Python, які дозволяють описувати відношення між індивідами. Запити до бази знань створюються шляхом пошуку об'єктів в множинах. За допомогою Python користувач може створювати власні способи подання і виведення знань. Зокрема, на відміну від OWL, в класах такої онтології можуть бути властивості, які не зберігають конкретного значення, а дозволяють його обчислити.

1.9 Вибір напрямку досліджень

Проведений аналіз [2, 6] показує, що основними критеріями працездатності ШСНУ є корозійна стійкість свердловинного обладнання, статична і втомна міцність колони ШН, вібростійкість з'єднань колон ШН та НКТ, герметичність НКТ та РЗ, зносостійкість та гідродинамічний опір протекторів, зносостійкість деталей насоса. Існує ряд основних факторів, які визначають працездатність ШСНУ і якими можна управляти без значних витрат часу і коштів. Зокрема перспективним є наступні рішення: удосконалення РЗ ШН комбінованими методами [69, 70], яке можливе навіть без модифікації конструкції ніпеля; оптимізація моменту згвинчування [19, 20]; застосування склопластикових ШН та удосконалення їхніх з'єднань [22]; дослідження роботи з'єднань в умовах циклічного і вібраційного навантаження; ремонт та зміцнення ШН і НКТ склопластиковими бандажами [6]; удосконалення конструкції протекторів [21]; зменшення частоти ходів; експлуатація колони склопластикових ШН в білярезонансних режимах [23]. Комбінації таких рішень на різних етапах ЖЦ ШСНУ можуть створити системний ефект, суттєво і комплексно підвищити ресурс і ефективність роботи ШСНУ без значних витрат часу і коштів.

Сучасні досягнення імітаційного моделювання, автоматизованого проектування, аналізу даних, штучного інтелекту, теорії систем і системного аналізу уможливають побудову ефективних інформаційних систем для проектування та підтримки ЖЦ ШСНУ, за допомогою яких можна приймати такі ефективні рішення на системному рівні. Компонентами таких ІС можуть бути параметричні імітаційні моделі ШСНУ та її частин, результати симуляції,

статистичні моделі відмов, бази знань, машини логічного виведення та інші інформаційні ресурси. У зв'язку зі складністю предметної області ІС підтримки ЖЦ ШСНУ, їх потрібно відносити до класу складних систем. Тоді, відповідно до принципу ізоморфізму теорії систем, їм повинні бути властиві усі закономірності складних систем. Цей факт треба враховувати під час побудови ІС та їхніх абстрактних моделей, які відображають найбільш загальні, як правило якісні, характеристики ІС. Відомі дослідження і концептуальні моделі таких систем слабо акцентують увагу на цьому. На основі міждисциплінарного огляду і аналізу основних загальносистемних закономірностей на прикладах різних природних та штучних складних систем можливою є розробка абстрактної моделі ІС і принципів її функціонування [213].

На основі загальносистемних закономірностей та MAS необхідне розроблення принципів побудови ІС, яка орієнтована на просту реалізацію і використання та створена простою і популярною мовою загального призначення Python. Завдяки цьому ІС буде гнучкою до змін, існуватиме можливість легкого удосконалення компонентів, розробнику будуть доступні усі широкі можливості цієї мови, зросте спільнота користувачів. Усі компоненти ІС повинні бути зручними для створення, модифікації під час удосконалення чи внесення інших змін, мати здатність до еволюції та простої інтеграції в ІС. Для цього вони повинні бути розроблені мовою Python або мати інтерфейс програмування. Існуючі моделі ШСНУ чи її частин не відповідають усім цим вимогам.

Інтеграція бази знань в ІС дозволить систематизувати знання, логічно обґрунтувати проектні рішення на різних етапах ЖЦ, уникнути психологічної інерції, яка проявляється в негнучкому, замкнутому мисленні, небажанні відійти від існуючих уявлень і постулатів. Розглянутий підхід характеризується простотою розробки бази та підтримкою найсучасніших перспективних технологій інженерії знань. Однак для розроблення баз знань необхідно застосовувати не спеціальні мови, а мови загального призначення, зокрема Python, що дозволить полегшити майбутнє розширення функціональності ІС.

На основі вищевикладеного можна сформулювати мету та завдання дисертаційного дослідження, які подані нижче.

Мета і завдання дослідження. Метою роботи є удосконалення методології автоматизованого проектування обладнання ШСНУ на основі системного підходу та її використання для комплексного та ефективного дослідження і забезпечення працездатності ШСНУ.

Досягнення поставленої мети вимагає вирішення наступних **завдань**:

1) розроблення принципів побудови робастних статистичних моделей відмов штангових колон на прикладі відмов штангових колон в НГВУ "Долина нафтогаз";

2) розроблення принципів компонентно-орієнтованого моделювання ШСНУ та розроблення параметричних імітаційних моделей ШСНУ зі здатністю до простої модифікації та інтеграції в інформаційну систему;

3) розроблення множини параметричних геометричних і скінченно-елементних моделей (СЕМ) та прикладних САПР проблемних деталей та вузлів ШСНУ, які дозволяють системне дослідження їхньої працездатності, зокрема різьбових з'єднань (РЗ) ШН та НКТ, з'єднань склопластикових штанг, ШН і НКТ з дефектами та склопластиковими бандажами (БС), протекторів для ШН, клапана свердловинного насоса;

4) дослідження, на основі моделей, працездатності різьбових з'єднань ШН та НКТ в умовах статичного та динамічного навантаження, у тому числі вібраційного, і розроблення принципів удосконалення та оптимізації їхньої конструкції;

5) дослідження, на основі моделей, працездатності елементів колон штанг і НКТ (пресових з'єднань склопластикових штанг, піделеваторного бурта штанги, штанг та НКТ з склопластиковими бандажами, протектора) та розроблення принципів удосконалення та оптимізації їхньої конструкції;

6) розроблення принципів побудови гнучкої бази знань, яка описує різні фактори, що впливають на надійність різьбових з'єднань ШСНУ;

7) розроблення абстрактної моделі та принципів реалізації інформаційної системи для проектування обладнання та підтримки життєвого циклу ШСНУ (далі – ІС), яка базується на множині моделей обладнання і володіє закономірностями складних систем.

РОЗДІЛ 2

СТАТИСТИЧНІ МОДЕЛІ ВІДМОВ КОЛОН НАСОСНИХ ШТАНГ

2.1 Залежності частоти відмов колони від відносної глибини обриву

В розділі описані принципи та методики побудови статистичних моделей відмов колон ШН [290, 291] на основі індуктивних методів самоорганізації моделей. Індуктивний підхід використовує таблицю початкових експериментальних даних і перебір великої кількості моделей цих даних з ціллю пошуку моделі оптимальної складності з застосуванням певного зовнішнього критерію селекції [292]. Основною ціллю є підвищення робастності та зменшення перенавчання моделей. Моделі розробляли на основі статистичних даних про відмови колон в НГВУ “Долина нафтогаз”. Були зібрані статистичні дані про 704 відмови (у тому числі часткові) колон ШН у 1999-2001 роках [1]. Ці дані являють собою таблицю (табл. Б.1), кожен рядок якої відповідає окремій відмові колони, а стовпчики містять наступні дані: назва свердловини, рік і місяць відмови, діаметр плунжера свердловинного насоса *Pump* (мм), довжина колони *H* (м), довжина секцій ШН різного діаметра (*H19*, *H22*, *H25*) (м), глибина обриву колони *H_o* (м), діаметр відмовленої ШН, тип відмови, газовий фактор *Gas*, відсоток води в продукції *Water*, відсутність (*Paraffin=0*) або наявність (*Paraffin=1*) СПУ, відсутність (*Curv=0*) або наявність (*Curv=1*) викривленості свердловини, інтервал кривизни викривленої свердловини (м), продуктивність свердловини *Product* (м³/добу), приведене напруження в верхній частині колони *Stress* (МПа). Реєструвались наступні типи відмов: руйнування по тілу ШН, руйнування по тілу муфти, руйнування по різьбі ніпеля (переважно корозійно-втомні), зрив різьби муфти і ніпеля, відгвинчування, обрив і заміна полірованого штока. Використаємо ці дані для побудови статистичних моделей відмов колон ШН.

Побудуємо залежності частоти відмов колони від відносної глибини обриву [290]. Відносна глибина обриву *h* – це відношення глибини обриву колони *H_o* до довжини колони *H*: $h = H_o / H$. Діапазон значень *h* [0, 1] поділимо на *k* інтервалів.

У кожному інтервалі $\Delta h_i = h_{i+1} - h_i$ порахуємо кількість відмов N_i . Відносною частотою відмов на інтервалі будемо називати відношення $n_i = N_i / \sum N_i$. Сума частот $\sum n_i = 1$. У випадку $\sum N_i \rightarrow \infty$, $\Delta h_i \rightarrow 0$ n_i можна вважати статистичною імовірністю $dF(h)$ відмови в інтервалі Δh_i . Неперервна невід'ємна гладка функція

$f(h) = \lim_{\Delta h \rightarrow 0} \frac{n_i}{\Delta h} = \frac{dF(h)}{dh}$ називається функцією густини імовірності відмов для

випадкової величини h . Для цієї функції $\int_0^1 f(h)dh = 1$, тобто в інтервалі $[0, 1]$

повинно існувати 100% відмов. Функцію $F(h) = \int_0^h f(h)dh$ будемо називати

кумулятивною функцією розподілу ймовірностей відмов для випадкової величини h . Її значення в крайніх точках: $F(0) = 0$, $F(1) = 1$. Імовірність відмови в інтервалі

$[a, b]$ визначається так: $P(a < h < b) = \int_a^b f(h)dh = F(b) - F(a)$, де $F(a)$ – це імовірність

відмови в інтервалі $[0, a]$.

Функцію $f(h)$ шукали шляхом апроксимації значень $n_i/\Delta h$ поліномом. Для пошуку найкращого полінома застосуємо індуктивний метод самоорганізації моделі – метод групового урахування аргументу [144, 292], згідно з яким послідовно породжуються моделі зростаючої складності до тих пір, поки не буде знайдений мінімум певного критерію якості моделі. Опишемо модифікований автором варіант методу. Нехай емпіричні дані $\mathbf{x} = (x_1, x_2, \dots, x_n)$, $\mathbf{y} = (y_1, y_2, \dots, y_n)$. Найкращу модель шукали серед множини поліномів M , яка утворена з усіх можливих комбінації членів повного полінома [144]:

$$f(x, a_0, a_1, \dots, a_m) = \sum_{i=0}^m a_i x^i = a_0 + a_1 x + \dots + a_m x^m \quad (2.1)$$

де a_i – коефіцієнти полінома, x – змінна, i – степінь, m – максимальний степінь полінома.

Коефіцієнти кожного полінома шукали методом найменших квадратів. Точність апроксимації оцінювали за допомогою коефіцієнта детермінації $R^2(\mathbf{y}, \mathbf{f}(\mathbf{x}))$ (квадрату коефіцієнта кореляції Пірсона векторів емпіричних \mathbf{y} і

теоретичних $\mathbf{f}(\mathbf{x}) = (f(x_1), f(x_2), \dots, f(x_n))$ значень). Якщо $R^2 \rightarrow 1$, то це свідчить, про близькість значень \mathbf{y} і $\mathbf{f}(\mathbf{x})$. Це внутрішній критерій відбору моделі, який показує наскільки добре модель описує конкретні емпіричні дані. Проте незначні зміни в емпіричних даних можуть призвести до різкого зниження його значення. В таких випадках слід користувались критерієм, який мало чутливий до таких змін. Розділимо емпіричні дані на три групи: група 0 – усі точки \mathbf{x}, \mathbf{y} , група 1 – непарні точки $\mathbf{x}_1, \mathbf{y}_1$, група 2 – парні точки $\mathbf{x}_2, \mathbf{y}_2$. Виберемо поліном f з множини M . Для кожної групи 0, 1, 2 знайдемо його коефіцієнти методом найменших квадратів та одержимо поліноми f_0, f_1, f_2 . У найпростішому випадку в якості зовнішнього критерію можна взяти $R^2(\mathbf{y}_2, \mathbf{f}_1(\mathbf{x}_2))$. В даному випадку використовували більш стійкий до змін в емпіричних даних узагальнений критерій (2.2) як лінійну комбінацію:

$$S = (S_0 + S_1 + S_2 + S_3)/4, \quad (2.2)$$

де $S_0 = R^2(\mathbf{y}, \mathbf{f}_0(\mathbf{x}))$ – оцінка моделі f_0 за усіма даними (внутрішній критерій);

$S_1 = R^2(\mathbf{y}_2, \mathbf{f}_1(\mathbf{x}_2))$ – оцінка моделі f_1 за даними групи 2 (критерій регулярності 1);

$S_2 = R^2(\mathbf{y}_1, \mathbf{f}_2(\mathbf{x}_1))$ – оцінка моделі f_2 за даними групи 1 (критерій регулярності 2);

$S_3 = R^2(\mathbf{f}_1(\mathbf{x}), \mathbf{f}_2(\mathbf{x}))$ – оцінка подібності моделей f_1 і f_2 за усіма даними (критерій незміщення [292]).

Такий розрахунок повторюється для кожного полінома з множини M . Найкращий поліном володіє максимальним значенням S . Остаточо його коефіцієнти розраховували за усіма точками \mathbf{x} . Для реалізації алгоритму автором розроблена програма мовою Python (лістинг Б.1). Під час вибору моделі додатково враховували вимогу невід'ємності функції $f(h)$ та вимогу $\int_0^1 f(h)dh \approx 1$.

Тобто вибиралась найкраща функція за критерієм S , яка відповідає цим вимогам.

Розглянемо усі відмови колон без урахування відмов полірованого штока. Кількість відмов 569. Гістограма відносних частот для $k=16$ показана на рис. 2.1. Модель:

$$f(h) = 0,58 + 7,54h - 20,46h^2 + 13,89h^3, \quad R^2 = 0,863, \quad S = 0,864$$

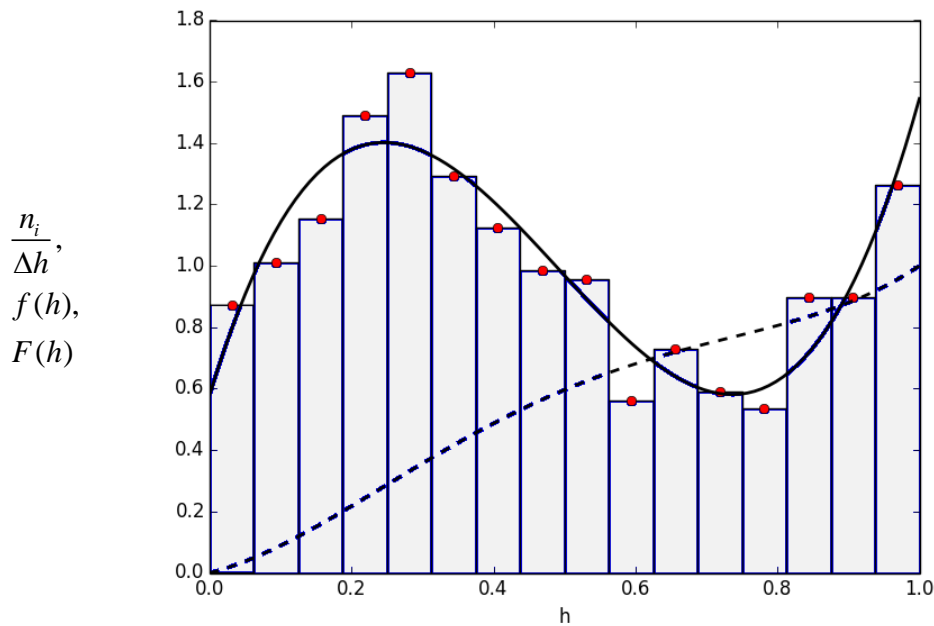


Рисунок 2.1 – Емпірична гістограма для величини h і теоретичні функції $f(h)$ (-) та $F(h)$ (- -)

Крива функції $f(h)$ для усіх відмов має S-подібний вигляд з двома мінімумами ($h=0$, $h=0,7$) і двома максимумами ($h=0,25$, $h=1$) та подібна на криві моделей, отриманих раніше методом найменших квадратів [1]. Проведемо горизонтальну лінію на рівні $f(h) = 1$. Отримаємо три точки її перетину з кривою $f(h)$ з абсцисами 0,07, 0,5, 0,9. Ці точки ділять діапазон h на інтервали: $[0, 0,07]$, $[0,07, 0,5]$, $[0,5, 0,9]$, $[0,9, 1]$. Орієнтовні значення імовірності відмов в цих інтервалах: 0,05, $0,6-0,05=0,55$, $0,9-0,6=0,3$, $1-0,9=0,1$ (табл. 2.2). Другий та четвертий інтервал характеризується підвищеною густиною імовірності відмов, тому під час проектування колони їх потрібно оснащувати деталями з посиленою конструкцією. Фактори, які впливають на частоти відмов на цих інтервалах описані в працях [37, 44, 88, 131, 137, 143]. Їхні максимуми наведено в таблиці 2.1.

Таблиця 2.1 – Максимуми негативних факторів на інтервалах свердловини

Фактор \ Інтервал	1	2	3	4
Навантаження розтягу на колону	+	+		
Навантаження стиску, згину і кручення на колону				+
Гідростатичний тиск (сприяє проникненню середовища в РЗ)				+
Викривленість свердловини (рис. 2.2)		+	+	+
Відсутність захисного шару СПУ на поверхні колони [21]				+

В праці [105] підвищення частоти відмов в середніх і нижніх ділянках колони пояснюють дією значних динамічних навантажень (гідравлічних ударів), які спричинені запізненням руху плунжера у порівнянні з рухом точки підвіски. Накладання негативних факторів в одному інтервалі призводить до різкого збільшення частоти відмов в ньому. Зокрема в праці [134] виявлено, що частота відмов колони зростає зі зменшенням глибини розташування максимального кута викривлення свердловини. В нашому випадку найбільша частота відмов спостерігається у таких викривлених свердловинах (рис. 2.2), які мають інтервал викривлення з початком 1019 м (середнє квадратичне відхилення 394 м) і кінцем 1262 м (середнє квадратичне відхилення 328 м).

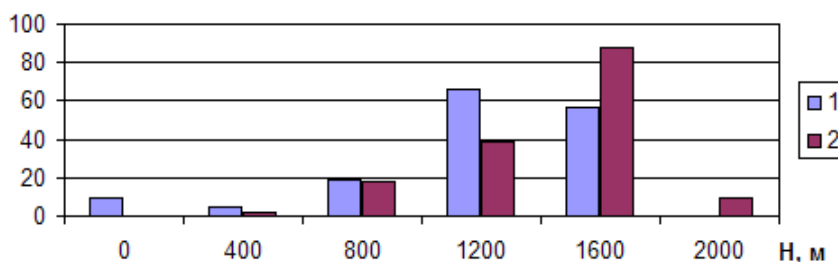


Рисунок 2.2 – Залежність кількості відмов у викривлених свердловинах від початку (1) і кінця (2) інтервалу викривлення

Розглянемо відмови колон з насосами малого діаметра 29 та 32 мм.

Кількість відмов 106. Модель (рис. 2.3а):

$$f(h) = 1,77 - 9,9h^2 + 10,2h^3, \quad R^2 = 0,547, \quad S = 0,592.$$

Розглянемо відмови колон з насосами середнього діаметра 38 та 44 мм. Кількість відмов 194. У даному випадку перевагу було віддано моделі з ненульовим вільним членом (рис. 2.3б):

$$f(h) = 0,33 + 10,74h - 28,09h^2 + 18,67h^3, \quad R^2 = 0,579, \quad S = 0,523.$$

Розглянемо відмови колон з насосами великого діаметра 57, 70 та 95 мм. Кількість відмов 269. Модель (рис. 2.3в):

$$f(h) = 0,49 + 5,64h - 10,2h^2 + 5,49h^4, \quad R^2 = 0,477, \quad S = 0,595.$$

В цілому ці результати відповідають отриманим раніше [1, 44] – обриви колон з насосами великого діаметра частіше відбуваються в їхній нижній частині. У випадку застосування насосів з малим діаметром перший максимум $f(h)$ знаходиться на гирлі. Зі збільшенням діаметра насоса зростає кількість відмов в нижній половині колони, а екстремуми кривої $f(h)$ зміщуються вправо. Це пояснюють [143] збільшенням навантажень стиску, згину і кручення на колону в нижній її частині, які призводять до процесів корозійної втоми, зношування та відгвинчування.

Розглянемо відмови в свердловинах без СПУ (рис. 2.3г). Кількість відмов 339. Модель:

$$f(h) = 0,52 + 8,87h - 22,38h^2 + 14,03h^3, \quad R^2 = 0,797, \quad S = 0,883.$$

Розглянемо відмови в свердловинах з СПУ (рис. 2.3д). Кількість відмов 229. Модель:

$$f(h) = 0,94 + 7,5h^2 - 25,6h^3 + 19,83h^4, \quad R^2 = 0,795, \quad S = 0,505.$$

В свердловинах з СПУ суттєво менша імовірність відмов в верхній частині колони, що можна пояснити зменшенням корозійно-втомних відмов внаслідок наявності захисного шару СПУ на поверхні ШН (табл. 2.1). Експериментальні дослідження [1] підтверджують такі захисні властивості СПУ.

Побудуємо моделі $f(h)$ для відмов різного типу ($k=12$). Розглянемо обриви по тілу ШН. Кількість відмов 115. Модель (рис. 2.3е):

$$f(h) = 1,01 + 2,4h - 18,95h^3 + 17,82h^4, \quad R^2 = 0,646, \quad S = 0,664.$$

Розглянемо обриви по тілу муфти. Кількість відмов 151. Модель (рис. 2.3ж):

$$f(h) = 1 + 4,69h - 15h^2 + 10,65h^3, \quad R^2 = 0,767, \quad S = 0,653.$$

Розглянемо обриви по різьбі ніпеля. Кількість відмов 45. Модель (рис. 2.3з):

$$f(h) = 0,83 + 0,95h^4, \quad R^2 = 0,238, \quad S = 0,457.$$

Імовірно, що такий характер залежності пов'язаний з корозійною втомою різьби ніпеля в умовах підвищених навантажень стиску і згину внизу колони. Розглянемо зриви різьби муфти. Кількість відмов 128. Модель (рис. 2.3і):

$$f(h) = 0,84 + 3,74h - 8,5h^2 + 5,69h^4, \quad R^2 = 0,778, \quad S = 0,633.$$

Розглянемо зриви різьби ніпеля. Кількість відмов 68. Модель (рис. 2.3к):

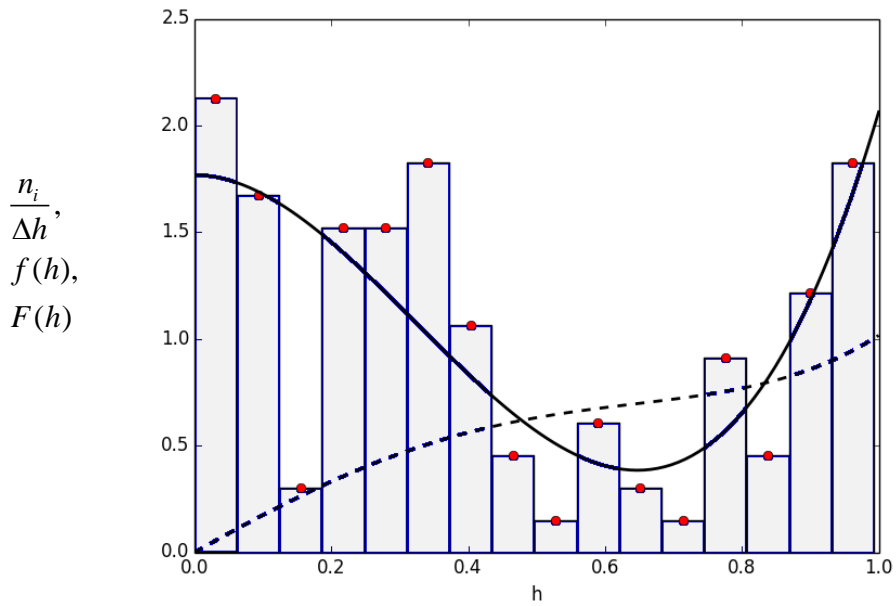
$$f(h) = 13,33h - 29,72h^2 + 13,75h^3 + 4,21h^4, \quad R^2 = 0,508, \quad S = 0,569.$$

Розглянемо відгвинчування. Кількість відмов 62. Модель (рис. 2.3л):

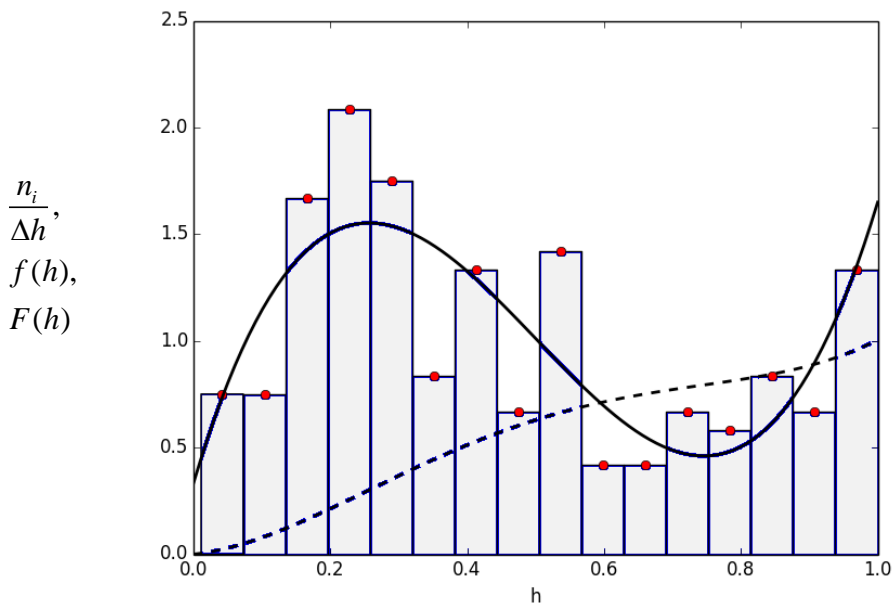
$$f(h) = 7,82h - 38,42h^3 + 33,55h^4, \quad R^2 = 0,851, \quad S = 0,847.$$

Помітно, що частота відгвинчувань мала біля гирла і висока в центральній частині колони і над насосом. Це можна пояснити послабленням затяжки РЗ внаслідок дії навантажень стиску на нього.

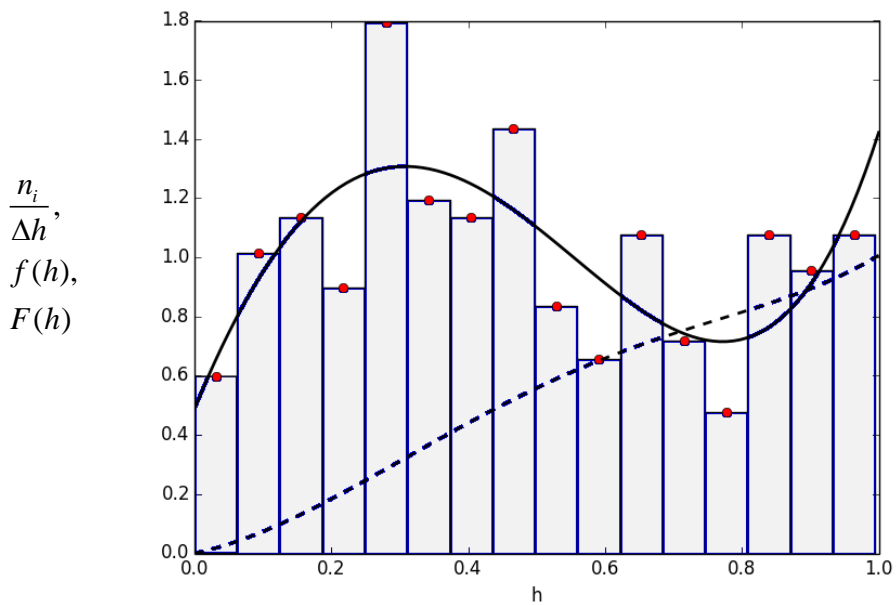
Моделі $f(h)$ можуть бути використані для обґрунтування заміни ШН на глибині h колони. Для полегшення пошуку небезпечних і безпечних інтервалів розроблено таблицю 2.2. У разі підйому колони ШН і після певного наробітку колони така заміна полягає у перестановці ШН з високоаварійних секцій у низькоаварійні та навпаки. У першу чергу це важливо для зменшення втомних відмов ШН.



а

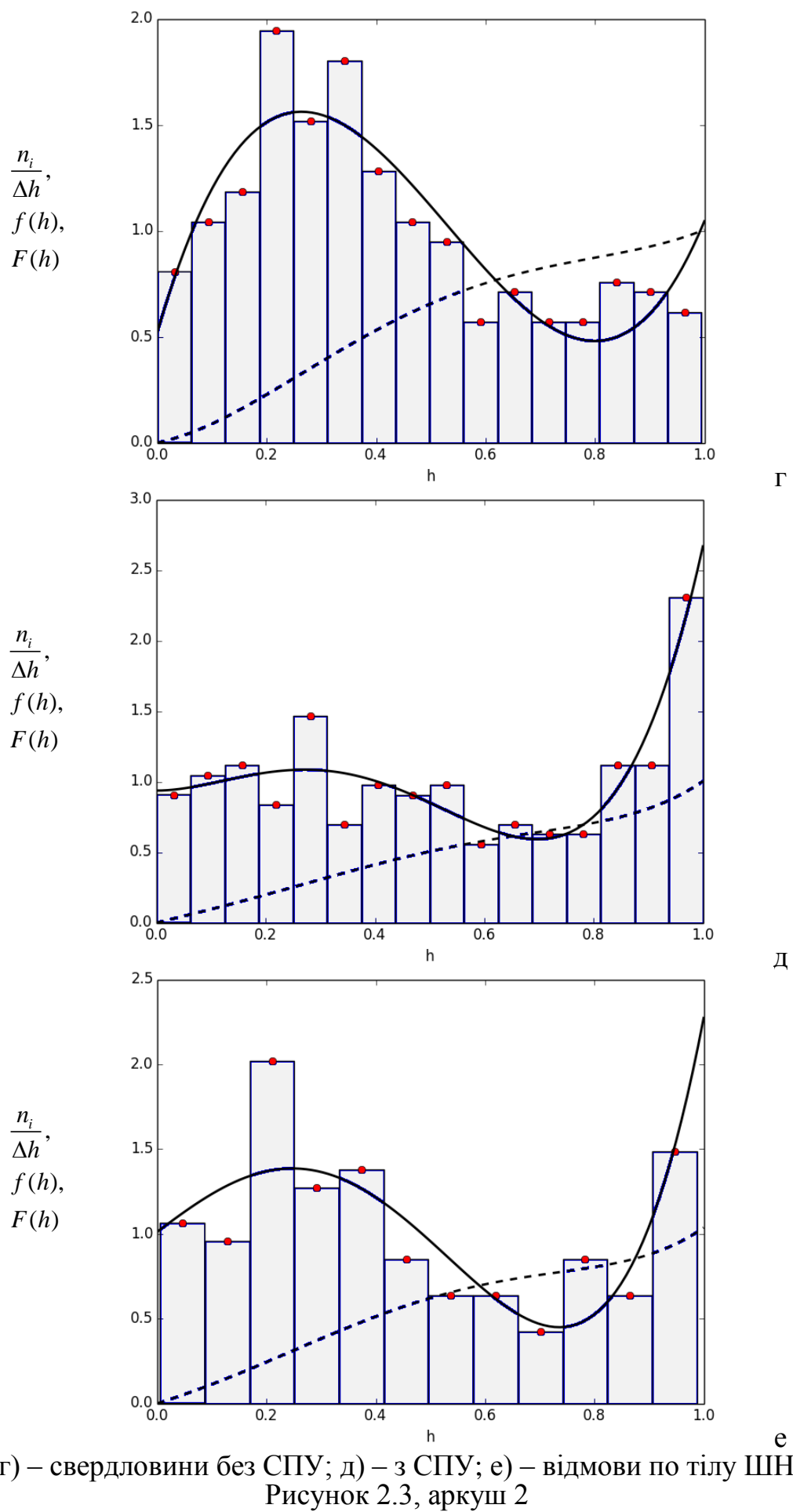


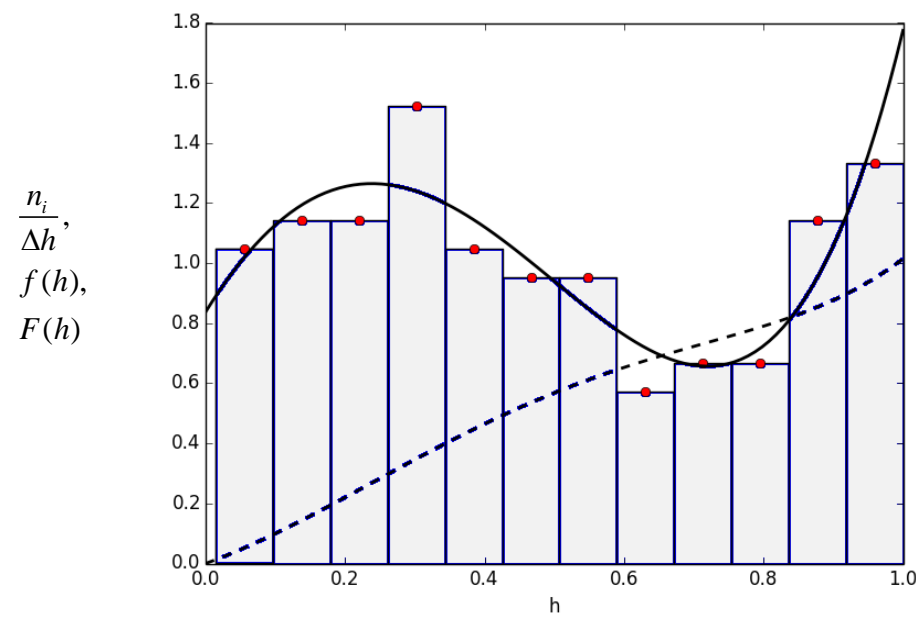
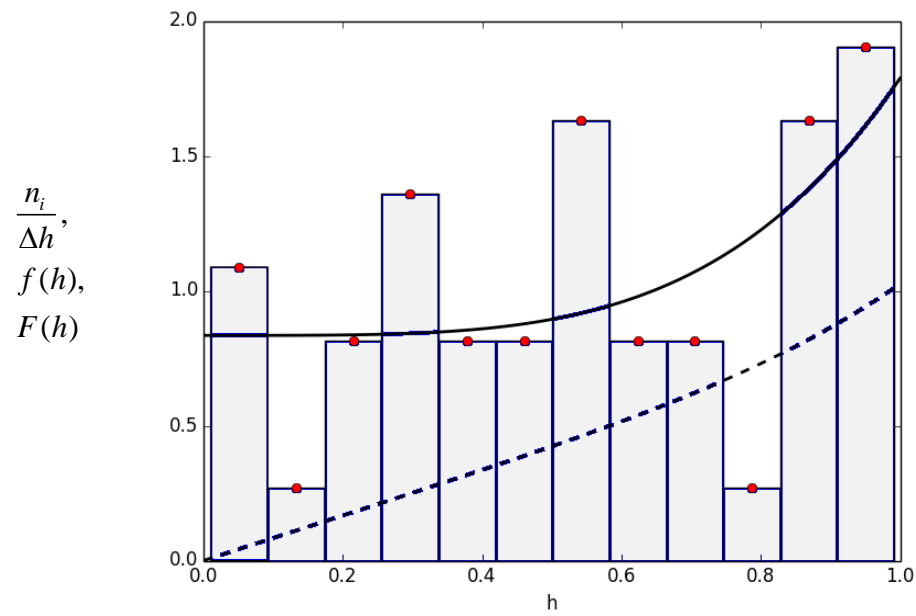
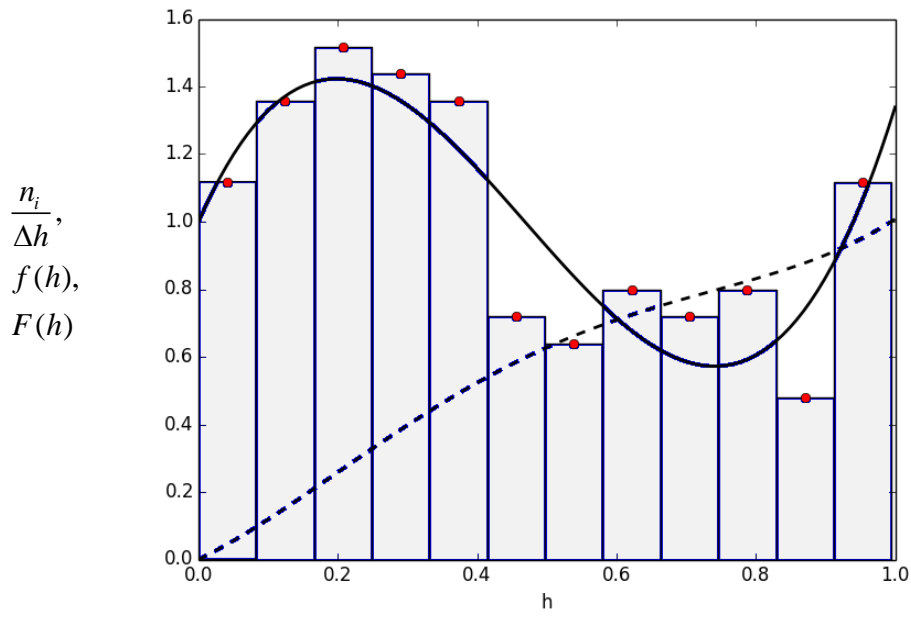
б



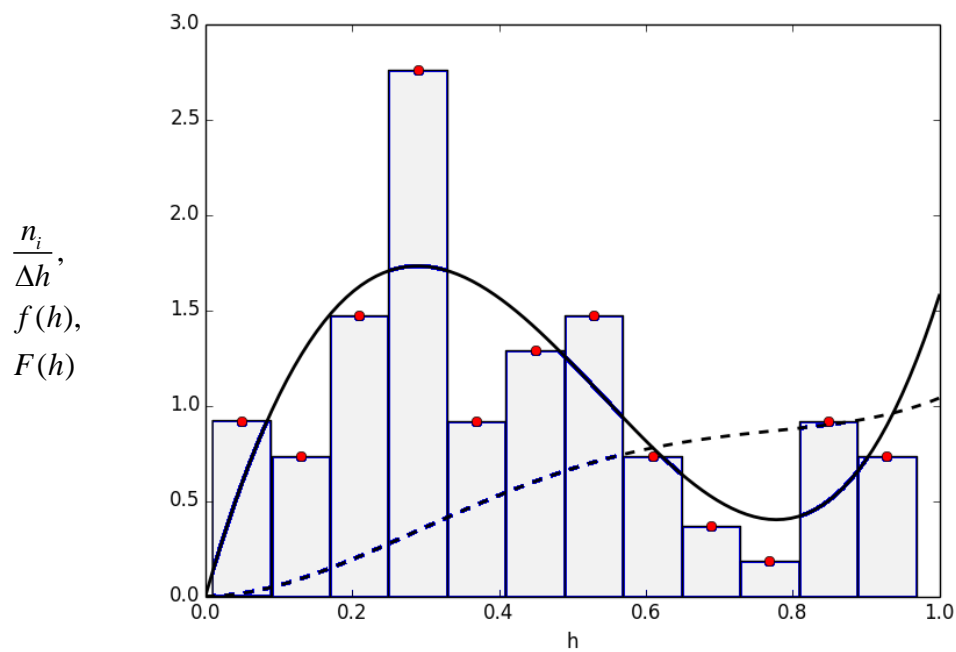
в

а) – насоси малого діаметра; б) – середнього; в) – великого
Рисунок 2.3, аркуш 1 – Емпіричні гістограми для величини h і теоретичні функції $f(h)$ (-) та $F(h)$ (- -)

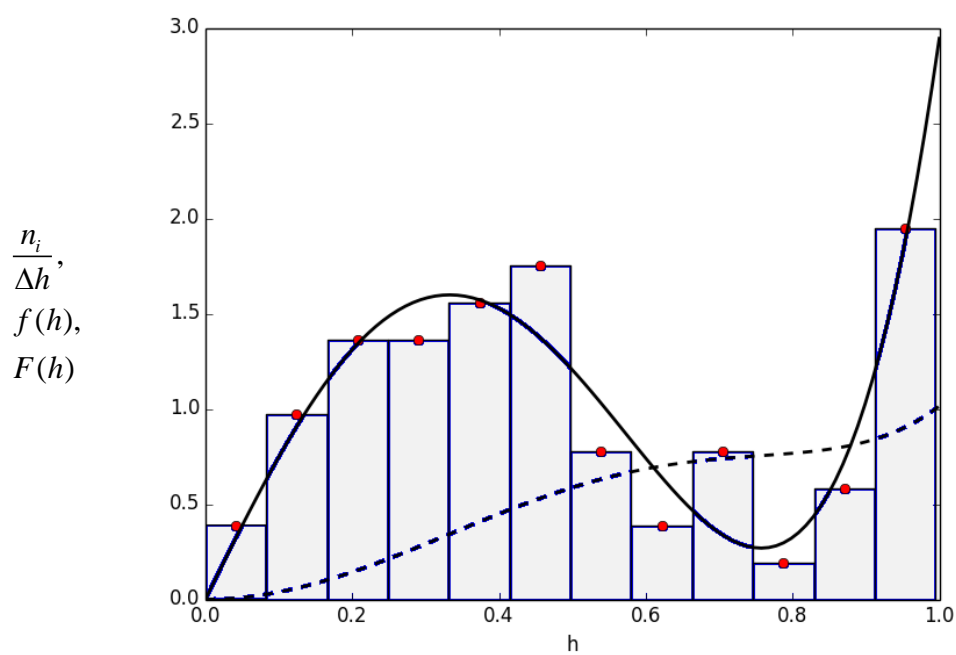




ж) – відмови по тілу муфти; з) – по різьбі ніпеля; і) – зриви різьби муфти
Рисунок 2.3, аркуш 3



к



л

к) – зриви різьби ніпеля; л) – відгвинчування

Рисунок 2.3, аркуш 4

Таблиця 2.2 – Теоретичні імовірності відмов P на інтервалах з підвищеною і пониженою густиною відмов

Клас відмов	Значення P на інтервалах з $f(h)<1$	Значення P на інтервалах з $f(h)>1$
Усі відмови крім відмов штока	$P(0<h<0,07) = 0,05$ $P(0,5<h<0,9) = 0,3$	$P(0,07<h<0,5) = 0,55$ $P(0,9<h<1) = 0,1$
Насоси малого діаметра	$P(0,34<h<0,87) = 0,3$	$P(0<h<0,34) = 0,5$ $P(0,87<h<1) = 0,2$
Насоси середнього діаметра	$P(0<h<0,06) = 0,05$ $P(0,56<h<0,95) = 0,28$	$P(0,06<h<0,56) = 0,57$ $P(0,95<h<1) = 0,1$
Насосаи великого діаметра	$P(0<h<0,11) = 0,1$ $P(0,56<h<0,95) = 0,29$	$P(0,11<h<0,56) = 0,51$ $P(0,95<h<1) = 0,1$
Свердловини без СПУ	$P(0<h<0,06) = 0,06$ $P(0,53<h<1) = 0,3$	$P(0,06<h<0,53) = 0,64$
Свердловини з СПУ	$P(0<h<0,1) = 0,1$ $P(0,4<h<0,84) = 0,35$	$P(0,1<h<0,4) = 0,3$ $P(0,84<h<1) = 0,25$
По тілу ШН	$P(0,5<h<0,9) = 0,25$	$P(0<h<0,5) = 0,6$ $P(0,9<h<1) = 0,15$
По тілу муфти	$P(0,46<h<0,94) = 0,3$	$P(0<h<0,46) = 0,6$ $P(0,94<h<1) = 0,1$
По різьбі ніпеля	$P(0<h<0,65) = 0,55$	$P(0,65<h<1) = 0,45$
Зриви різьби муфти	$P(0<h<0,04) = 0,05$ $P(0,46<h<0,9) = 0,3$	$P(0,04<h<0,46) = 0,5$ $P(0,9<h<1) = 0,15$
Зриви різьби ніпеля	$P(0<h<0,08) = 0,05$ $P(0,58<h<0,96) = 0,2$	$P(0,08<h<0,58) = 0,7$ $P(0,96<h<1) = 0,05$
Відгвинчування	$P(0<h<0,13) = 0,1$ $P(0,54<h<0,91) = 0,2$	$P(0,13<h<0,54) = 0,55$ $P(0,91<h<1) = 0,15$

2.2 Залежності частоти відмов колони від відносної глибини обриву та діаметра плунжера свердловинного насоса

Побудовано поліноміальні регресійні моделі виду $f(h, Pump)$, де $Pump$ – діаметр плунжера свердловинного насоса, f – функція густини імовірності відмов [290]. Моделі отримали за допомогою індуктивного методу самоорганізації моделей шляхом використання програми GMDH Shell 3 [293]. Підготовку даних виконували за допомогою програми мовою Python з використанням бібліотек NumPy, Matplotlib, Pandas. Спочатку для кожного інтервалу значень діаметрів насоса будували гістограму величини h . Кількість інтервалів для h – 14, для $Pump$ – 4 ([29, 32, 38], 44, 57, [70, 95]). Налаштування алгоритму [293]: спосіб вибору спостережень – парні/непарні, спосіб перевірки – перехресна перевірка з виключенням по одному, критерій S вибору моделі – середній квадрат помилки |навчання-перевірка|, основний алгоритм – покроковий змішаний. В більшості випадків повний поліном задавався з наступними обмеженнями: максимальна степінь змінної 3, мінімальна – (-1), максимальна сума степенів в члені – 6, максимальна кількість змінних в члені – 2. Для цього критерію S кращою моделлю вважається модель з меншим його значенням. В деяких випадках замість моделі з меншим значенням критерію вибиралась більш проста модель, якщо значення їхніх критеріїв не суттєво відрізнялись. Точність моделі оцінювалась за коефіцієнтом детермінації R^2 . Модель для усіх відмов крім відмов штока (свердловини з СПУ і без них) (рис. 2.4):

$$f = -0,27109 + 12,5111h + 18,6381h^3 - 28,9086h^2 + 1,64308/(h \cdot Pump) - 14,02h/Pump$$

$$R^2=0,5; S=0,05707.$$

На відміну від попередніх простих моделей ця модель змогла задовільно описати локальний мінімум густини відмов в зоні $h=0,1$ та локальний максимум біля $h=0$ для насосів малого діаметра. Їх можна пояснити такими факторами як максимальні напруження розтягу та максимум товщини захисного шару СПУ на колоні в зоні $h=0..0,1$.

Модель (рис. 2.5) для усіх відмов крім відмов штока (свердловини без СПУ):

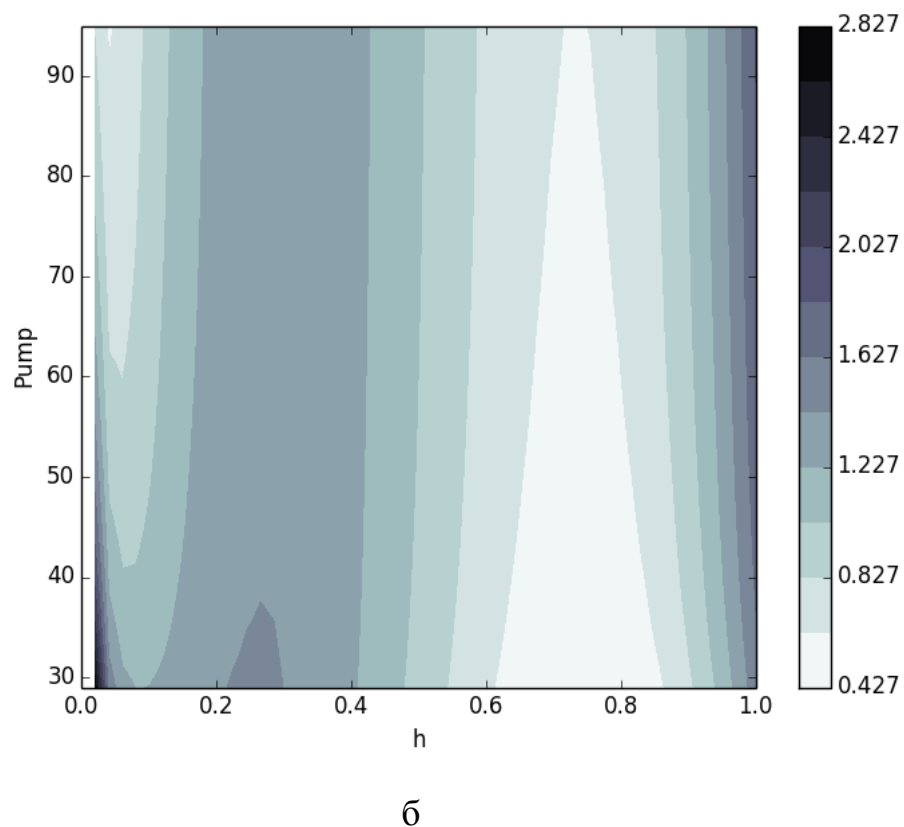
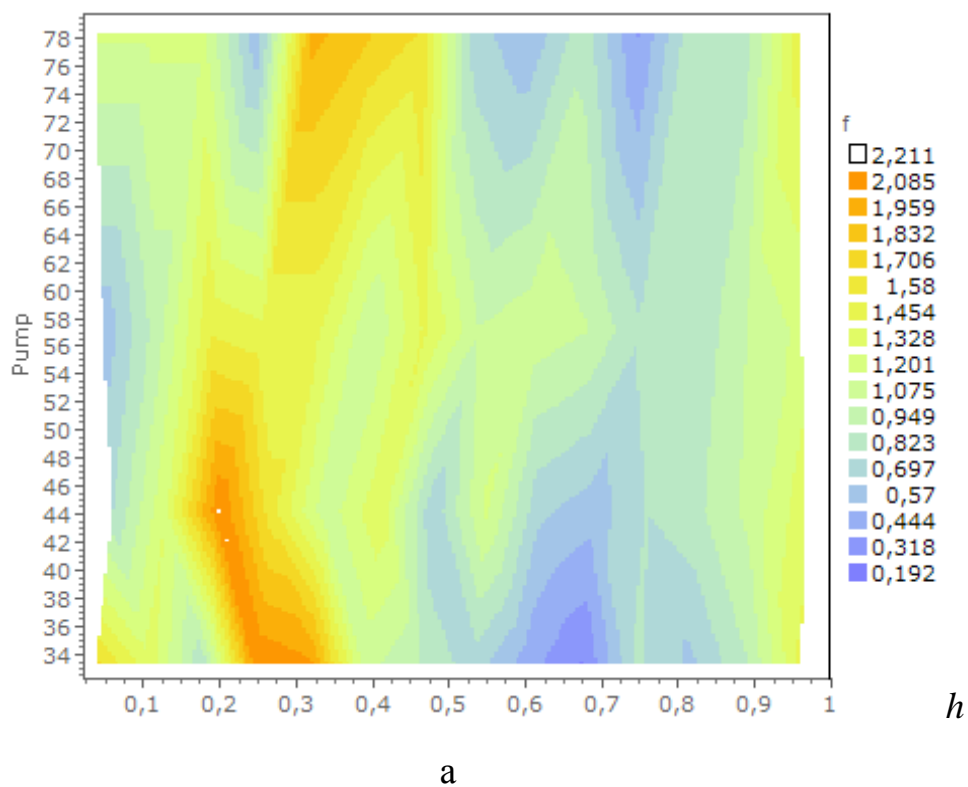
$$f=0,685687 - 1182,73h^2/Pump + 801,619h^3/Pump + \\ + 411,105h/Pump + 0,00482628h \cdot Pump, R^2=0,5, S=0,055957.$$

В цьому випадку максимум відмов в зоні $h=0,25$ та мінімум відмов $h=0,75$ для насосів малого діаметра більш виражені. Це можна пояснити зростанням частоти відмов внизу колони з великими діаметрами насосу внаслідок згину колони [44].

Модель (рис. 2.6) для усіх відмов крім відмов штока (свердловини з СПУ):

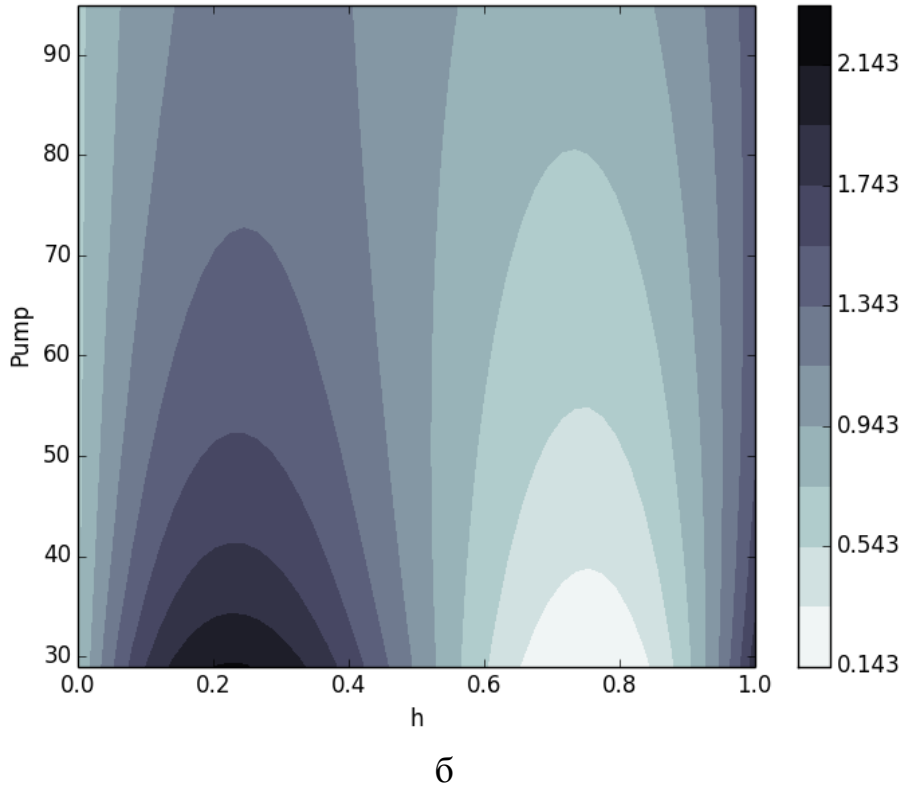
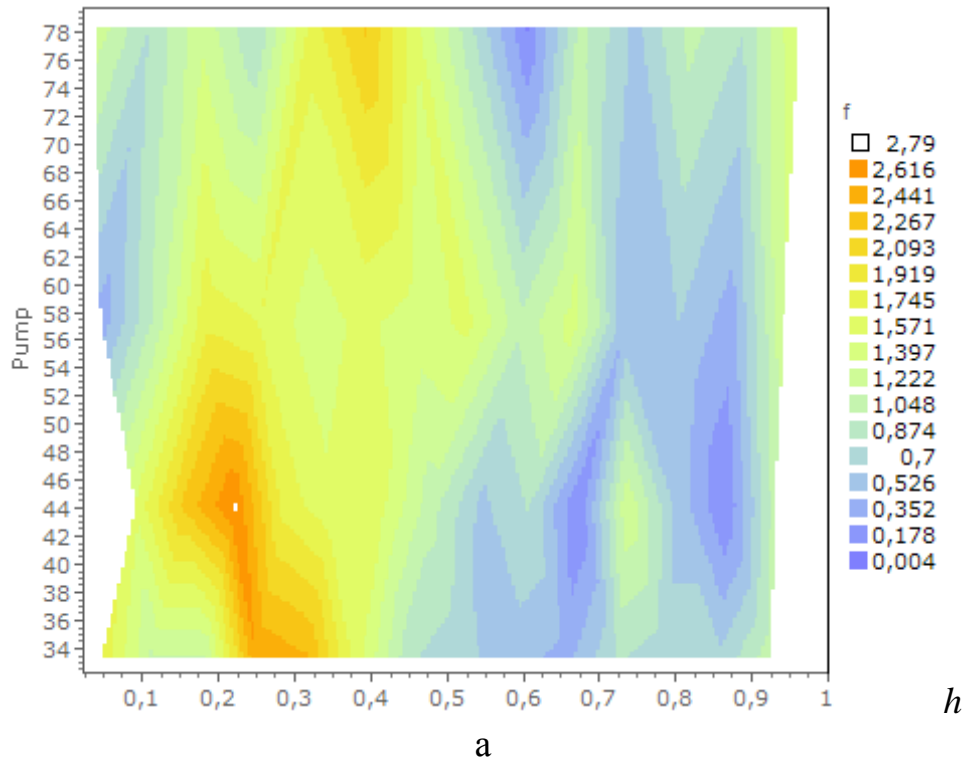
$$f=1,316602 + 1275,17h^4/Pump - 1918,14h^3/Pump + 884,801h^2/Pump - \\ - 162,979h/Pump - 2,0558e-08h^4 \cdot Pump^4, R^2=0,44, S=0,047501.$$

Тут помітне збільшення густини відмов на насосом ($h=1$) особливо для малих діаметрів насосу. Це пояснюється зменшенням корозійно-втомних відмов вверху колони внаслідок захисного шару СПУ на ній.

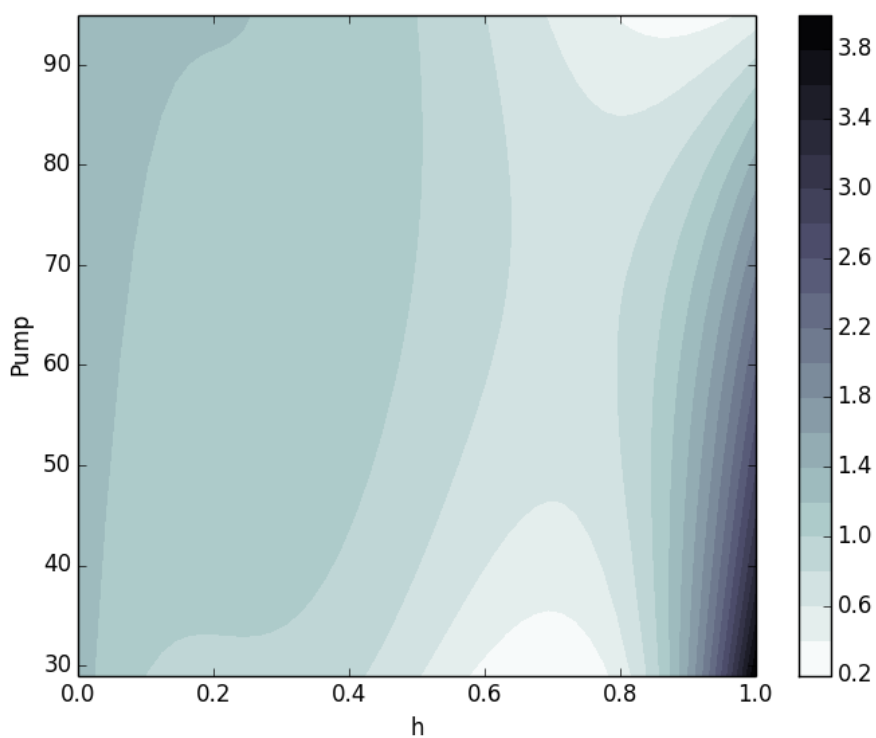
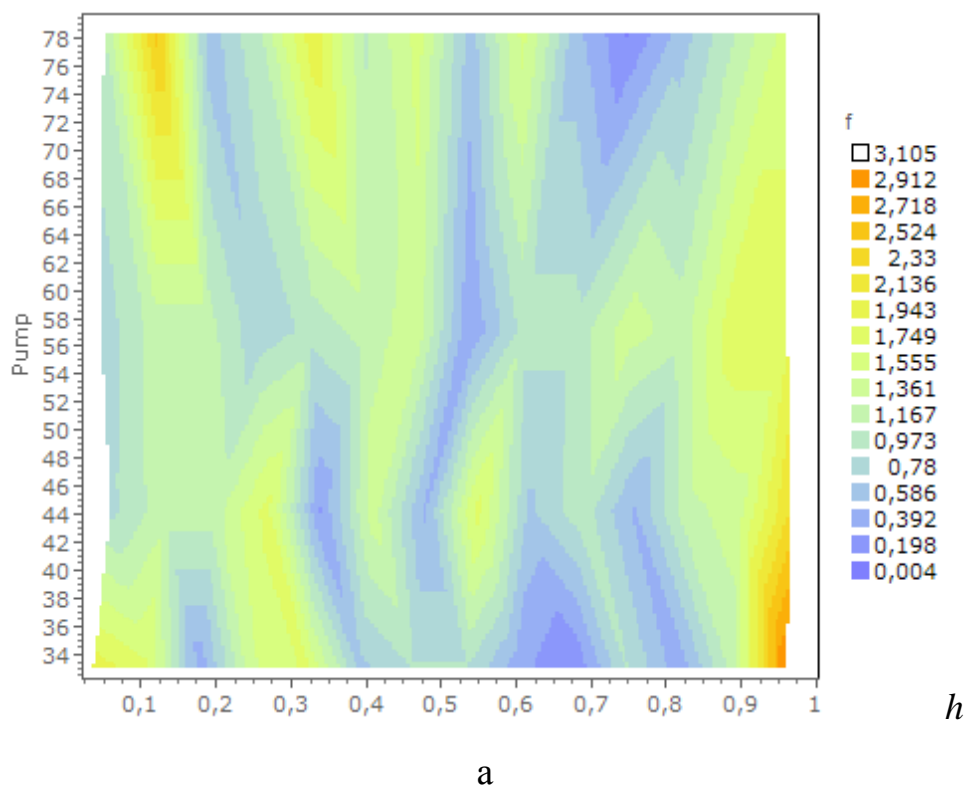


а) – емпіричні дані; б) – теоретичні дані за моделлю

Рисунок 2.4 – Густина імовірності відмов для випадкової величини h та насосів діаметра $Pump$ (усі відмови)



а) – емпіричні дані; б) – теоретичні дані за моделлю
 Рисунок 2.5 – Густина імовірності відмов для випадкової величини h та насосів діаметра $Pump$ (свердловини без СПУ)



а) – емпіричні дані; б) – теоретичні дані за моделлю

Рисунок 2.6 – Густина імовірності відмов для випадкової величини h та насосів діаметра $Pump$ (свердловини з СПУ)

2.3 Багатовимірні поліноміальні моделі частоти відмов

Розраховано кількість відмов Kv (показник аварійності) кожної свердловини за 3 роки. Відмови полірованого штока не враховували. Свердловини з $Kv \leq 4$ віднесено до класу низькоаварійних свердловин ($Kvcat=0$), а свердловини з $Kv > 4$ – до високоаварійних ($Kvcat=1$). Побудовано поліноміальну модель виду $Kvcat=f(X)$, яка дозволить класифікувати свердловини за ознакою $Kvcat$. Множина параметрів моделі $X=\{Pump, Stress, Gas, Curv, Water, H, Product, Paraffin\}$. Кореляція цих змінних з Kv наведена в табл. 2.3. Невідомі значення змінних $Gas, Water, Product$ заміняли відповідними середніми значеннями.

Таблиця 2.3 – Коефіцієнти кореляції параметрів моделі з Kv

<i>Stress</i>	<i>Water</i>	<i>Curv</i>	<i>Pump</i>	<i>Gas</i>	<i>Paraffin</i>	<i>H</i>	<i>Product</i>
0,21	0,2	-0,18	0,18	-0,14	0,073	-0,04	-0,014

Приблизне уявлення про залежності $f(X)$ дають наступні графічні залежності середніх значень $M[x]$ параметра $x \in X$ від Kv (рис. 2.7). На гістограмах для *Stress, Water, Pump, H, Product* показані інтервали $M[x] \pm S[x]$, а на гістограмах для *Curv, Gas, Paraffin* – інтервали $M[x] \pm 0,1S[x]$. Тут $S[x]$ – середнє квадратичне відхилення величини x для заданого Kv . Помітно, що Kv зростає зі збільшенням *Stress, Water, Pump, Paraffin* та зменшується зі збільшенням *Gas, H*.

Моделі $f(X)$ отримали за допомогою індуктивного методу самоорганізації моделей шляхом використання програми GMDH Shell 3. Налаштування алгоритму [293]: спосіб вибору спостережень – парні/непарні, спосіб перевірки – перехресна перевірка з виключенням по одному, критерій – середній квадрат помилки, основний алгоритм – покроковий з доданням. Повний поліном задавався з членами $x_i x_j$ та x_i / x_j . Екзамен проводився над 20% вибіркою, дані якої рівномірно вибирались з генеральної сукупності. Над вихідною змінною $Kvcat$ були виконані перетворення "Декомпозиція категорій" та "Збалансувати класи". Модель наведено нижче. Для визначення класу за моделлю значення $Kvcat$ слід заокруглювати до 0 або 1. Точність моделі наведено в таблиці 2.4.

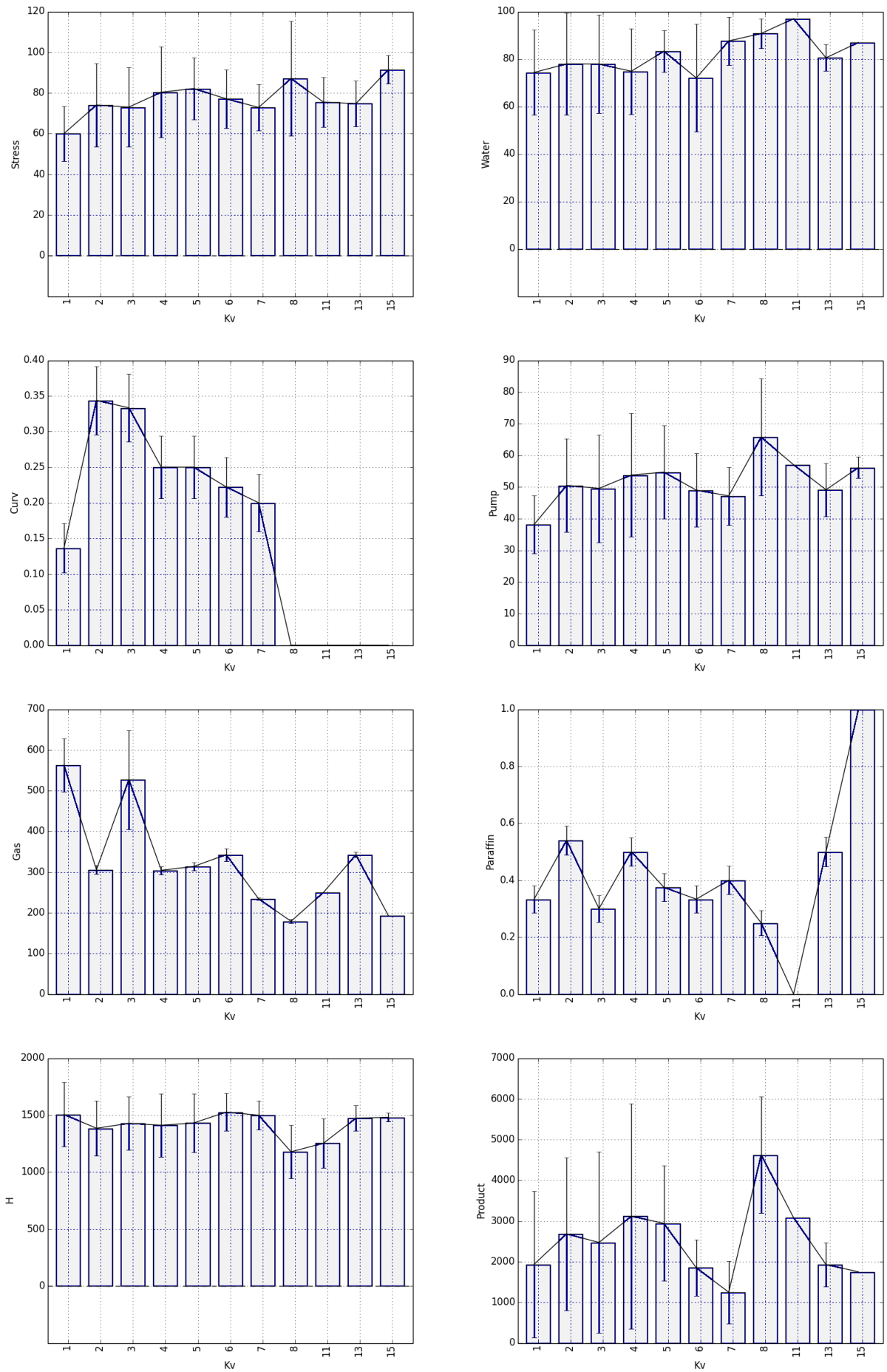


Рисунок 2.7- Залежності середніх значень параметрів від показника аварійності K_v

$$\begin{aligned}
Kvcat = & -0,0620677 - 285,812/(Pump \cdot Stress) - 3,56277e-07 \cdot Gas \cdot Product - \\
& 119,045 \cdot Curv/Gas + 0,0414565 \cdot Water - 17,3462 \cdot Stress/H - 0,736736 \cdot Water/Pump - \\
& 3,28266e-05 \cdot Gas \cdot Water + 0,00723915 \cdot Water \cdot Curv - 90,3911/(Pump \cdot Water) - 4,58273e- \\
& 06 \cdot Water \cdot Product + 8,90024e-05 \cdot Product + 2,75137e-06 \cdot Pump \cdot Product - \\
& 26409,5/(H \cdot Water) + 0,0128187 \cdot Gas/Product - 0,64763 \cdot Water/Product - \\
& 0,000348037 \cdot H \cdot Curv + 0,00293219 \cdot Gas - 0,00130525 \cdot H/Product - \\
& 0,0026065 \cdot Gas \cdot Paraffin + 2284,61 \cdot Paraffin/H - 116,789 \cdot Curv/Stress + 81,4503 \cdot Curv/ \\
& Pump + 0,736313 \cdot Pump/Product - 83,9568 \cdot Paraffin/Gas + 81,23 \cdot Paraffin/Pump - \\
& 80,4992 \cdot Paraffin/Stress + 0,0113072 \cdot Gas/Stress + 0,0116843 \cdot Product/Water - \\
& 1,21023 \cdot Paraffin.
\end{aligned}$$

Таблиця 2.4 – Точність класифікації: навчання (ліворуч) і екзамен (праворуч)

	Класифіковано		Всього	Чутли- вість	Класифіковано		Всього	Чутли- вість
	як 0	як 1			як 0	як 1		
Фактично 0	<i>TN=180</i>	<i>FP=72</i>	252	0,714	<i>TN=41</i>	<i>FP=23</i>	64	0,641
Фактично 1	<i>FN=28</i>	<i>TP=175</i>	203	0,862	<i>FN=7</i>	<i>TP=43</i>	50	0,860
Всього	208	245	455		48	66	114	
Правильність	0,865	0,709			0,854	0,652		
F1-міра	0,783	0,778			0,732	0,741		
Базовий рівень	0,554	0,554	0,554		0,561	0,561	0,561	
Точність	0,780	0,780	0,780		0,737	0,737	0,737	
	Правильно 355 (78%), неправильно 100 (22%), стандартне відхилення 0,459, зважена F1-міра 0,780, κ-міра 0,564				Правильно 84 (74%), неправильно 30 (26%), стандартне відхилення 0,493, зважена F1-міра 0,736, κ-міра 0,484			

В матриці помилок (табл. 2.4) літера *N* означає клас $Kvcat=0$, літера *P* – клас $Kvcat=1$. Літера *T* означає, що прогноз відповідає дійсності, а *F* – не відповідає. В якості показника якості моделі використовують правильність $(TN+TP)/(TN+TP+FN+FP)$, точність $TP/(TP+FP)$, повноту $TP/(TP+FN)$, F1-міру (середнє гармонічне точності та повноти) [145], κ-міру (коефіцієнт каппи Коена). Якщо необхідно знизити кількість прикладів *FN* (свердловин, які помилково віднесені до низькоаварійних), то в якості показника якості моделі слід використовувати повноту. Точність моделі можна підвищити до 0,9 шляхом вибору алгоритму класифікації "Випадковий ліс" в програмі GMDH Shell 3. Однак складність такої моделі не дозволяє описати її у вигляді простого полінома.

Розглянемо відмови полірованого штока. Кількість відмов 135 (123 зноси полірованого штоку, 12 обривів). Свердловини з $Kv=1$ віднесемо до класу

низькоаварійних свердловин ($Kv_{cat}=0$), а свердловини з $Kv>1$ – до високоаварійних ($Kv_{cat}=1$). Розрахуємо середні значення факторів відмов полірованого штока для цих класів (табл. 2.5). Для виявлення відмінності між середніми двох класів використовували t -критерій Стьюдента. В таблиці показані значення p -рівня статистичної значущості t -критерію – імовірності помилки прийняти гіпотезу про нерівність середніх, коли в дійсності середні рівні. Найбільш статистично значущими факторами, які впливають на частоту відмов штока є: довжина секцій ШН діаметром 25 мм $H25$, викривленість свердловини, СПУ, приведені напруження, діаметр плунжера. Збільшення їхніх значень призводить до зростання навантажень на полірований шток, що, разом з його перекосом, призводить до інтенсивного зношування.

Таблиця 2.5 – Середні значення факторів відмов полірованого штока та значення p -рівня t -критерію

Параметр	$Kv=1$	$Kv>1$	p -рівень t -критерію
Кількість відмов	67	68	-
Діаметр плунжера насоса, мм	47	51	0,156
Відсоток відмов в свердловинах з СПУ	33	48	0,064
Глибина свердловини, м	1456	1432	0,574
Довжина секцій ШН діаметром 25 мм, м	273	379	0,033
Газовий фактор	660	700	0,91
Відсоток води в продукції	77	81	0,328
Продуктивність свердловини, м ³ /добу	2272	2645	0,477
Приведене напруження, МПа	72	78	0,068
Відсоток відмов в викривлених свердловинах	16	31	0,049

Побудуємо поліноміальну модель виду $Kv_{cat}=f(X)$, яка дозволить класифікувати свердловини за ознакою Kv_{cat} . Множина параметрів моделі $X=\{Pump, Stress, Curv, H25, Paraffin, Water\}$. Налаштування алгоритму такі як у попередній моделі окрім основного алгоритму (комбінаторний швидкий). Також задана опція "обмежитись 5 найкращими змінними". Точність моделі наведено в таблиці 2.6.

$$K_{vcat} = 5,38653 - 0,0137214 \cdot Pump \cdot Paraffin + 3,66415e-05 \cdot Pump \cdot Stress - 3,38547 \cdot Pump / Stress - 6,61368 / Pump - 1,76909 \cdot Stress / Pump - 0,000508174 \cdot H25 \cdot Curv + 60,8566 \cdot Paraffin / Stress - 0,0585349 \cdot Stress \cdot Curv - 359,116 \cdot Curv / Stress + 9,57126 \cdot Curv$$

Таблиця 2.6 – Точність класифікації: навчання (ліворуч) і екзамен (праворуч)

	Класифіковано		Всього	Чутли- вість	Класифіковано		Всього	Чутли- вість
	як 0	як 1			як 0	як 1		
Фактично 0	<i>TN=43</i>	<i>FP=10</i>	53	0,811	<i>TN=11</i>	<i>FP=3</i>	14	0,786
Фактично 1	<i>FN=16</i>	<i>TP=39</i>	55	0,709	<i>FN=3</i>	<i>TP=10</i>	13	0,769
Всього	59	49	108		14	13	27	
Правильність	0,729	0,796			0,786	0,769		
F1-міра	0,768	0,750			0,786	0,769		
Базовий рівень	0,509	0,509	0,509		0,519	0,519	0,519	
Точність	0,759	0,759	0,759		0,778	0,778	0,778	
	Правильно 82 (76%), неправильно 26 (24%), стандартне відхилення 0,487, зважена F1-міра 0,759, ж-міра 0,519				Правильно 21 (78%), неправильно 6 (22%), стандартне відхилення 0,471, зважена F1-міра 0,778, ж-міра 0,555			

Частоти відмов штока в різні календарні місяці (рис. 2.8) дозволять виявити вплив на них погодних умов. Зокрема помітне постійне зростання кількості відмов після періоду з максимальною кількістю опадів.

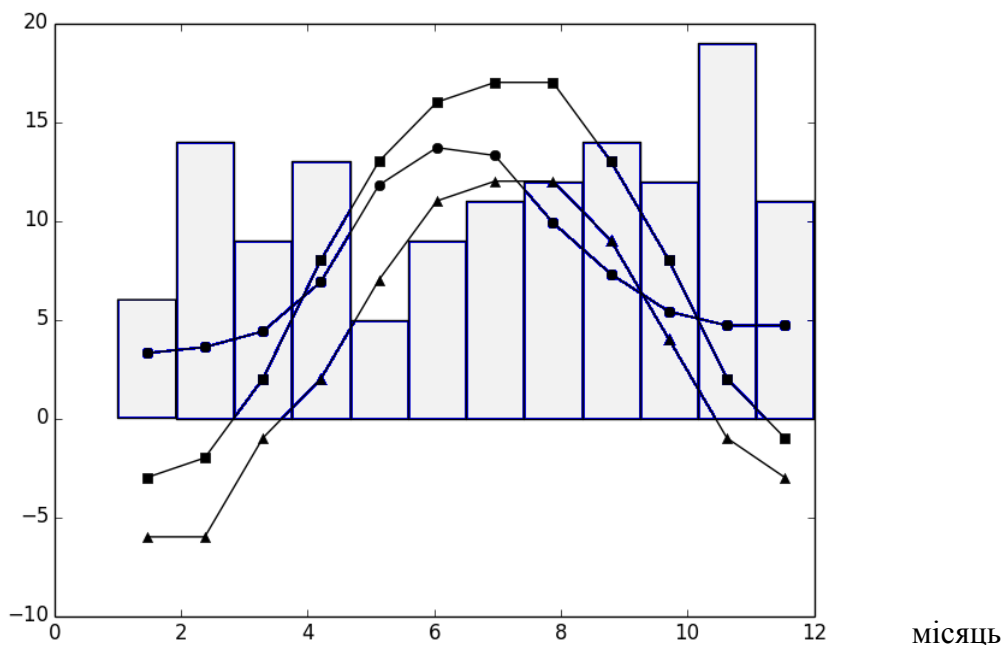


Рисунок 2.8 – Залежності частоти відмов штока (гістограма), середньої місячної температури повітря ■ (°C), точки роси повітря ▲ (°C) і кількості опадів ● (дм) від календарного місяця

Це можна пояснити накопиченням пошкоджень в результаті зношування під час наявності води в ущільненні. Але прямої залежності кількості відмов від кількості опадів виявлено не було. Можливий зв'язок частоти відмов штока з середньою місячною температурою повітря, точкою роси повітря і кількістю опадів потребує додаткових експериментальних досліджень. Тому ці фактори не були включені в модель.

2.4 Моделі частоти відмов на основі ансамблів дерев рішень

Сучасні статистичні методи обробки даних про відмови можуть бути використані для прогнозування частоти відмов колони за різними параметрами свердловини. Метою є обґрунтування ефективності застосування перспективних методів ансамблів дерев рішень, для прогнозування частоти відмов колон ШН [291]. Розглядали метод випадкового лісу та метод градієнтного бустінга (підсилення) дерев регресії, які реалізовані у пакеті Python для машинного навчання scikit-learn версії 0.19.1 [272]. Ці популярні методи основані на використанні множини дерев рішень для створення більш точної моделі. Вони достатньо універсальні, не потребують попереднього масштабування даних і володіють високою правильністю [145]. Моделі розробляли на основі статистичних даних про відмови колон в НГВУ “Долинанаштогаз” [37]. Моделі враховують наступні параметри свердловини: діаметр насоса *Pump*, приведене напруження в верхній частині колони *Stress*, газовий фактор *Gas*, відсутність або наявність викривленості свердловини *Curv*, відсоток води в продукції *Water*, довжина колони *H*, продуктивність свердловини *Product*, відсутність або наявність СПУ *Paraffin*, довжина секцій ШН різного діаметра (*H19*, *H22*, *H25*), місяць відмови *Month*, кількість відмов за три роки *Kv*. Кількість даних – 563, відмови штока не враховували. Код програми наведений у додатку (лістинг Б.2).

Спочатку виконували випадкове перемішування даних та поділ їх на дві частини – для пошуку моделі (80%) та її перевірки (20%). Для оцінки якості регресійних моделей використовували коефіцієнт детермінації R^2 . Найкращі

значення параметрів моделей шукали в певних інтервалах (для RandomForestRegressor $n_estimators=[50, 150]$, $max_depth=[3, 5]$, для GradientBoostingRegressor $n_estimators=[50, 150]$, $learning_rate=[0,01, 0,6]$, $max_depth=[3, 5]$) за допомогою функції для стохастичної оптимізації на основі генетичних алгоритмів *differential_evolution* з пакету SciPy (версія 1.0.0) [170]. Ця функція шукала максимум результату (середнього значення R^2) 7-блокової перехресної перевірки моделі. Результати пошуку для RandomForestRegressor: $n_estimators=133$, $max_depth=5$. Перехресна перевірка моделі на тестових 20% даних показала середнє значення правильності $R^2=0,74$. Результати пошуку для GradientBoostingRegressor: $n_estimators=120$, $learning_rate=0,15$, $max_depth=4$. Середнє значення правильності $R^2=0,89$. Криві перевірки (рис. Б.1) і навчання (рис. Б.2) можуть бути використані для обґрунтування значень $n_estimators$, $learning_rate$ і кількості даних для навчання ($train_size$). Суцільна лінія – це правильність $score$ (R^2) даних для навчання, а штрихова – правильність даних для перевірки. За допомогою функції *feature_importances_* моделі GradientBoostingRegressor обчислили важливість ознак: *Product* – 0,19, *Gas* – 0,14, *H* – 0,12, *H25* – 0,12, *H19* – 0,10, *H22* – 0,09, *Water* – 0,09, *Stress* – 0,08, *Month* – 0,03, *Pump* – 0,02, *Paraffin* – 0,02, *Curv* – 0,01. З рис. 2.9 помітно, що модель прогнозує дещо завищені значення частоти відмов для $Kv < 4$, і дещо занижені для $Kv > 4$.

Розглянута також задача бінарної класифікації (лістинг Б.3). Свердловини з $Kv \leq 4$ віднесено до класу низькоаварійних свердловин ($Kvcat=0$), а свердловини з $Kv > 4$ – до високоаварійних ($Kvcat=1$). Таким чином отримано приблизно збалансовані класи за кількістю відмов. Знайдені значення параметрів найкращої моделі RandomForestClassifier ($n_estimators=121$, $max_depth=5$) та моделі GradientBoostingClassifier ($n_estimators=120$, $learning_rate=0,38$, $max_depth=5$). Криві навчання і перевірки показані на рис. Б.3, Б.4. В таблиці 2.7 наведено результати перехресної перевірки моделей на тестових даних. Жирним виділені матриці помилок розміром 2×2 .

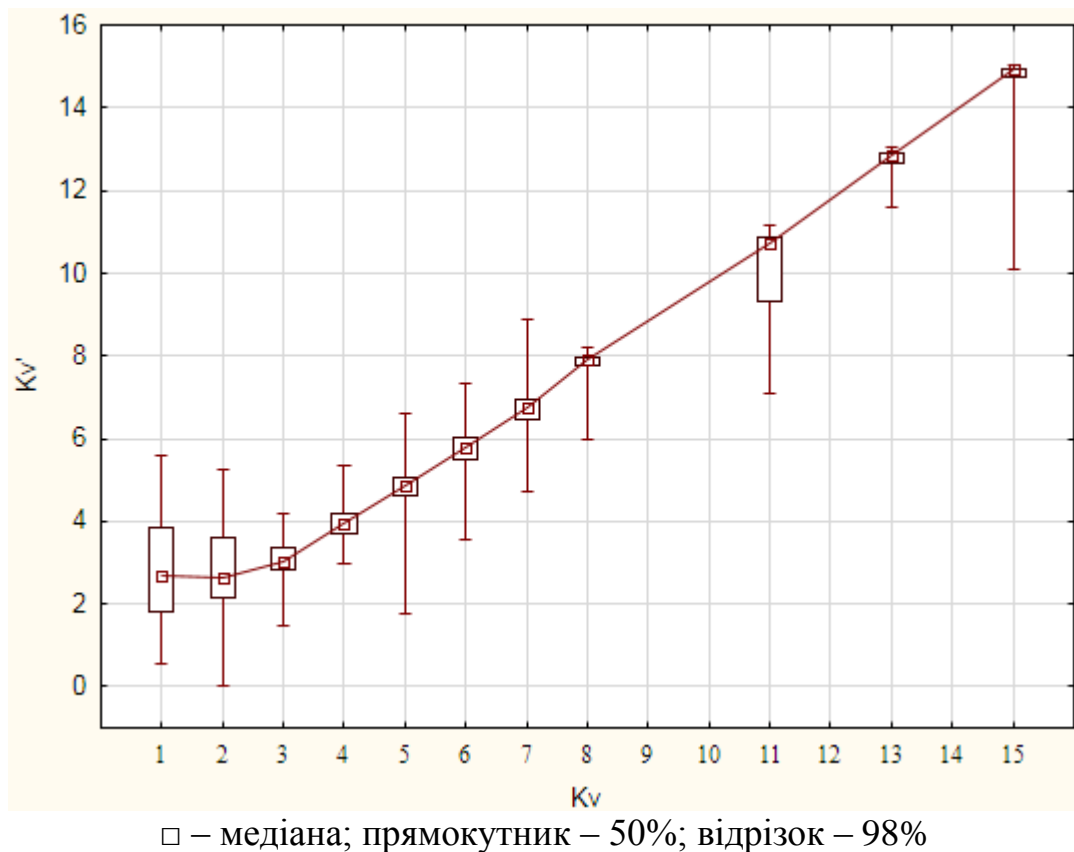


Рисунок 2.9 – Реальні Kv і прогнозовані Kv' значення частоти відмов для моделі GradientBoostingRegressor

Таблиця 2.7 – Середні значення результатів перехресної перевірки моделей RandomForestClassifier і GradientBoostingClassifier на тестових даних

Модель	Фактично	Класифіковано як		Точність	Повнота	F1-міра	Всього
		0	1				
RandomForest	0	TN=51	FP=10	0,89	0,84	0,86	61
	1	FN=6	TP=46	0,82	0,88	0,85	52
GradientBoosting	0	TN=56	FP=4	0,95	0,93	0,94	60
	1	FN=3	TP=50	0,93	0,94	0,93	53

Шляхом перехресної перевірки обчислена середня точність класифікатора (площа під кривою точності-повноти) (рис. Б.5) RandomForestClassifier (0,94) та GradientBoostingClassifier (0,99). Важливість ознак для GradientBoostingClassifier: *Product* – 0,09, *H* – 0,08, *Gas* – 0,07, *Stress* – 0,07, *H25* – 0,06, *H22* – 0,06, *Water* – 0,05, *H19* – 0,04, *Pump* – 0,01, *Month* – 0,01, *Curv* – 0,01, *Paraffin* – 0,01.

Помітно, що моделі GradientBoosting володіють дещо вищою правильністю (0,89 для регресії та 0,94 для класифікації) та їхнє застосування дозволить

ефективно ідентифікувати високоаварійні свердловини ще на етапі проектування [291]. Для використання моделі на етапі експлуатації статистичні дані можуть бути доповнені такими змінними як відносна глибина обриву колони, діаметр обірваної ШН, тип відмови.

2.5 Висновки до розділу

1. Розроблені моделі $f(h)$ та $f(h, Pump)$ можуть бути використані для наближеного визначення густини відмов та імовірності відмов в заданих інтервалах h і для обґрунтування зміцнення чи заміни ШН на цих інтервалах [290]. Ці моделі відповідають максимумам негативних факторів на інтервалах свердловини, підтверджують і доповнюють результати отримані автором та іншими авторами раніше.

2. Моделі $Kvcat=f(X)$, які враховують множину факторів, пов'язаних з параметрами ШСНУ та її відмови, дозволять з високою імовірністю визначити клас аварійності колони ШН та полірованого штока на етапі проектування. Точність класифікації аварійності колон – 0,737. Точність можна підвищити до 0,9 шляхом застосування алгоритму класифікації "Випадковий ліс" на основі дерев рішень. Найбільш статистично значущими факторами, які впливають на частоту відмов штока є: довжина секцій ШН діаметром 25 мм, викривленість свердловини, СПУ, приведенне напруження, діаметр плунжера.

3. Обґрунтовано ефективність застосування методів ансамблів дерев рішень (випадкового лісу та градієнтного бустінга дерев регресії) для прогнозування частоти відмов колон ШН [291]. Моделі GradientBoosting володіють дещо вищою правильністю (0,89 для регресії та 0,94 для класифікації) та їхнє застосування дозволить ефективно ідентифікувати високоаварійні свердловини ще на етапі проектування. Найбільш статистично значущими факторами, які впливають на частоту відмов ШН є: продуктивність, довжина колони, газовий фактор, приведенне напруження.

4. Впровадження методики побудови статистичних моделей на виробництві дозволить швидко будувати складні, точні та робастні статистичні моделі частоти

відмов колон та ефективно ідентифікувати високоаварійні колони та їхні інтервали ще на етапі проектування, виявляти причини відмов та приймати рішення щодо забезпечення їхньої працездатності. Для підвищення точності моделей необхідно уважно ставитись до збору статистичних даних про відмови, не допускати в них пропусків і помилок, збільшувати кількість параметрів моделі, які корелюють з частотою відмов (наприклад, режимами відкачування, динамограмами, марками сталей ШН), а також доповнити дані інформацією про колони без відмов у вказаний період.

Розглянуті статистичні моделі відмов колон ШН, які основані на промислових даних про відмови, та розроблені Python-програми для їхньої побудови є важливими компонентами інформаційної системи для проектування і підтримки життєвого циклу ШСНУ так як дозволять прогнозування працездатності колон на основі великої кількості мало вивчених факторів. Проте ці моделі потребують об'ємної вибірки і можуть бути не точними після принципово нових проектних рішень. Тому вони повинні бути доповнені зручними для побудови імітаційними моделями ШСНУ, які дозволяють проведення дешевих комп'ютерних експериментів із симуляцією її динаміки.

РОЗДІЛ 3

КОМПОНЕНТНО-ОРИЄНТОВАНЕ МОДЕЛЮВАННЯ ШСНУ

3.1 Імітаційна модель ШСНУ на основі абстрактних автоматів

Нижче описані принципи моделювання ШСНУ за допомогою абстрактних автоматів [294, 295]. Секцією колони ШН будемо називати частину колони, яка володіє такими атрибутами як довжина секції, діаметр ШН, матеріал ШН, кут відхилення свердловини внизу секції від вертикалі. Застосування в моделі секцій дозволяє моделювати колону з різними властивостями по довжині. Вузлом будемо називати точку з'єднання секцій колони ШН. В основному для розрахунку компонентів загального навантаження, які діють на колону ШН та плунжер насоса, будемо використовувати працю [97]. Розглянемо суму сил (Н), які діють на колону ШН або секцію (рис. 3.1) без врахування сил, що діють на плунжер насоса:

хід вверх:

$$F_{ум} + F_{дин} + F_{тер} + F_{вн.тер},$$

хід вниз:

$$F_{ум} + F_{дин} - F_{тер} - F_{вн.тер} - F_{г.тер},$$

де $F_{ум}$ – вага секції ШН в рідині (Н):

$$F_{ум} = mg - \rho_{рід} V_{ум} g,$$

де m – маса секції ШН (кг),

g – прискорення вільного падіння ($g=9,81$ м/с²),

$\rho_{рід}$ – густина рідини (кг/м³),

$V_{ум}$ – об'єм секції ШН (м³);

$F_{дин}$ – сила інерції секції ШН (Н):

$$F_{дин} = m \cdot a,$$

де a – прискорення секції ШН (м/с²);

$F_{тер}$ – сила тертя ШН об НКТ в похилій свердловині (Н) без врахування в'язкого тертя ШН $F_{г.тер}$ (див. нижче). Для спрощеної моделі тертя Кулона:

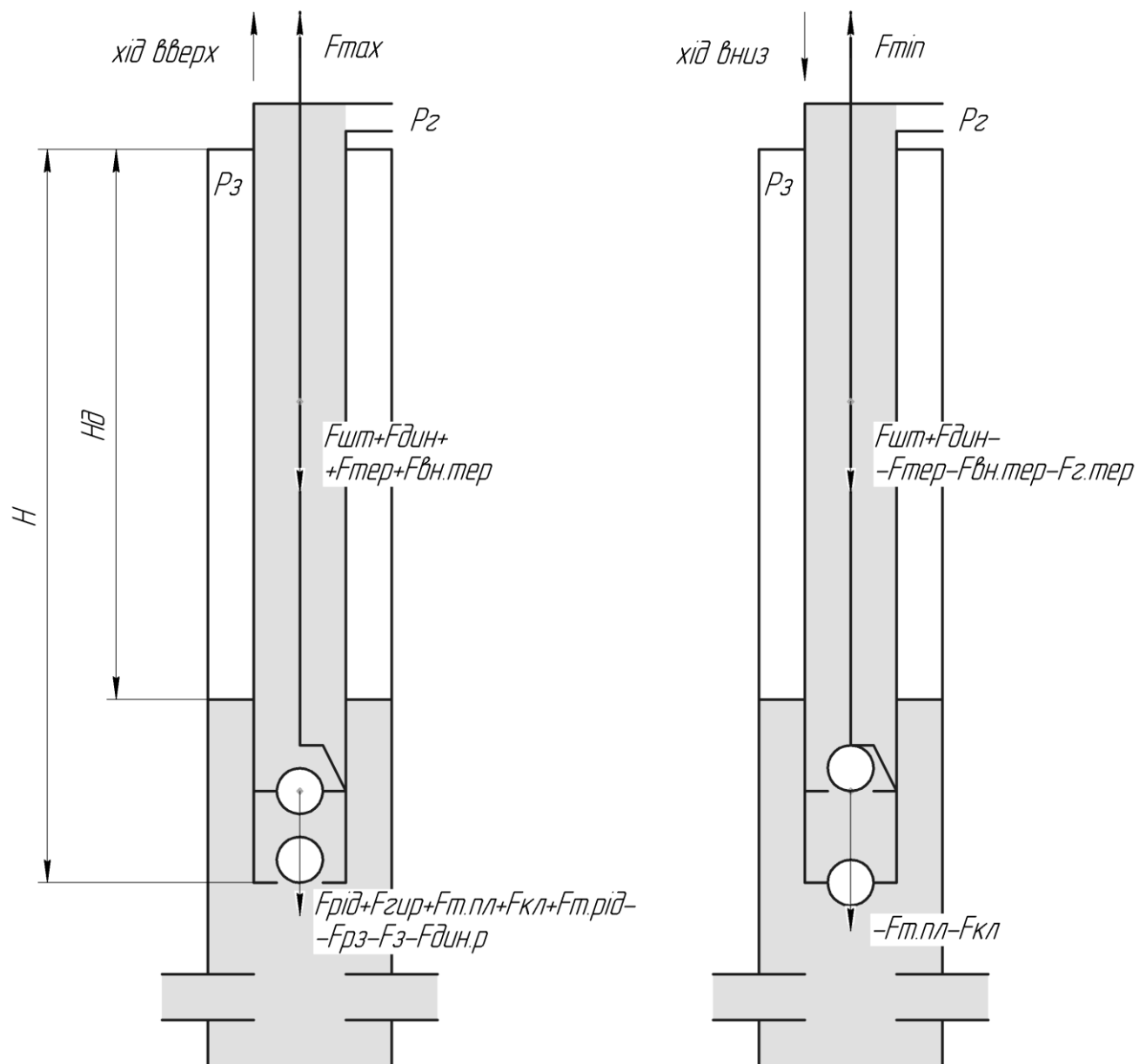


Рисунок 3.1 – Сили, що діють на колону ШН

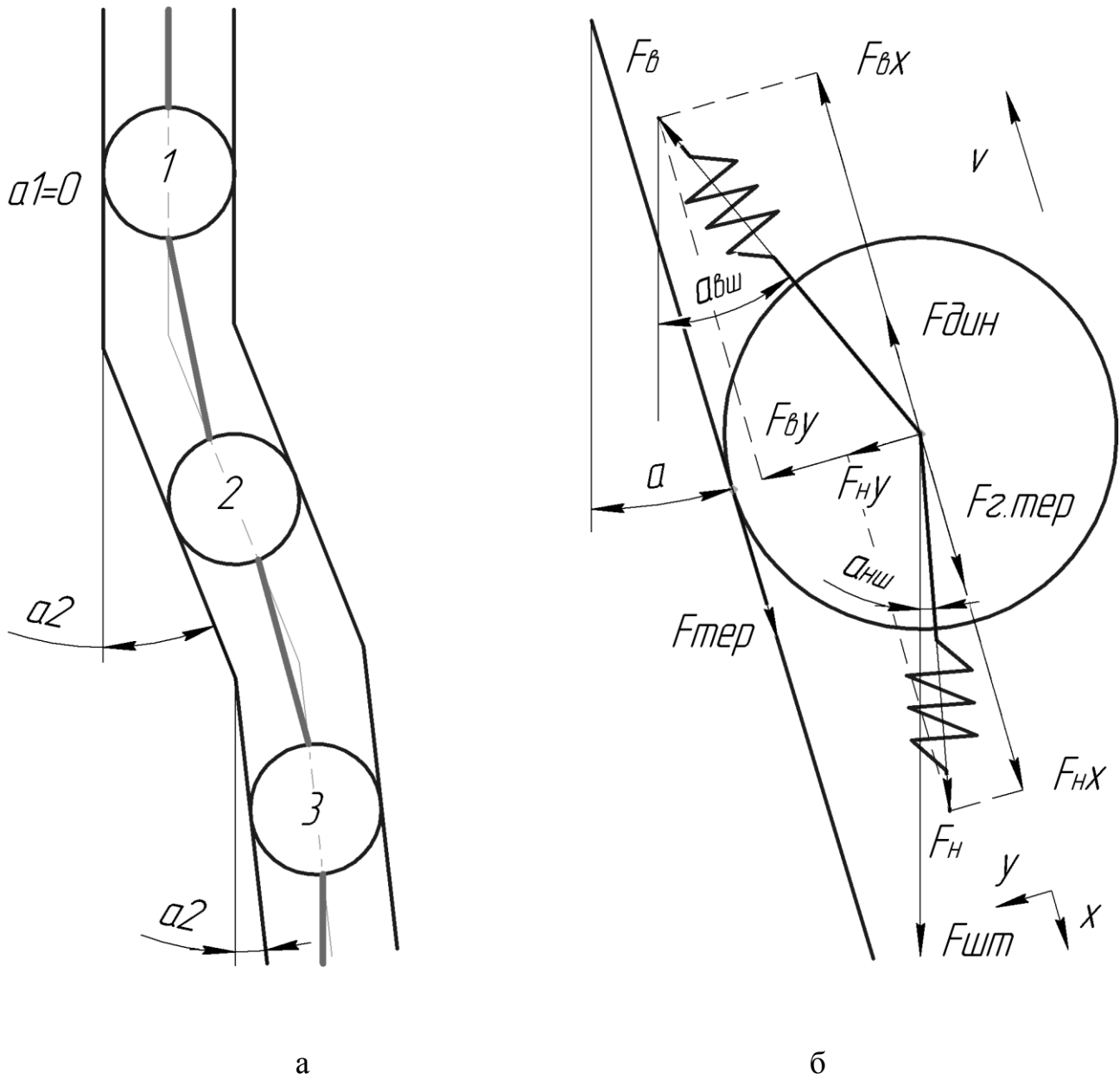


Рисунок 3.2 – Вузли секцій моделі колони ШН (а) та сили, що діють на вузол секції у викривленій свердловині (б)

$$F_{\text{тер}} = -k \cdot |F_{\text{нy}}| \cdot \text{th}\left(\frac{v}{v_0}\right),$$

Для розширеної моделі тертя Штрибека:

$$F_{\text{тер}} = -(k + (k_{\text{max}} - k)) \cdot e^{-\left|\frac{v}{v_s}\right|^n} \cdot |F_{\text{нy}}| \cdot \text{th}\left(\frac{v}{v_0}\right),$$

де k – коефіцієнт тертя ковзання Кулона,

k_{max} – максимальний коефіцієнт тертя ковзання для зони граничного тертя,

v – швидкість секції ШН (м/с),

v_s – коефіцієнт швидкості ковзання Штрибека (приймаємо $v_s = 0,1$ м/с),

n – коефіцієнт кривої Штрибека (приймаємо $n=1$),

$\text{th}\left(\frac{v}{v_0}\right)$ – множник, необхідний для згладжування залежності сили тертя від

швидкості v біля нуля (приймемо $v_0=0,01$ м/с);

$F_{\text{нy}}$ – сума сил, нормальних до стінки НКТ (Н) (рис. 3.2 б):

$$F_{\text{нy}} = F_{\text{умy}} + |F_{\text{ey}} + F_{\text{нy}}|,$$

де $F_{\text{умy}}$ – нормальна сила від ваги ШН (Н):

$$F_{\text{умy}} = F_{\text{ум}} \cdot \sin \alpha,$$

де α – кут відхилення свердловини від вертикалі біля вузла (рад);

F_{ey} – нормальна сила від натягу верхньої секції (Н) (рис. 3.2 б):

$$F_{\text{ey}} = F_e \cdot \sin \alpha_e,$$

де F_e – сила від натягу верхньої секції (Н),

α_e – кут між верхньою секцією і свердловиною (рад):

$$\alpha_e = \alpha_{\text{ви}} - \alpha,$$

де $\alpha_{\text{ви}}$ – кут відхилення верхньої секції від вертикалі (рад);

$F_{\text{нy}}$ – нормальна сила від натягу нижньої секції (Н) (рис. 3.2 б):

$$F_{\text{нy}} = F_n \cdot \sin \alpha_n,$$

де F_n – сила від натягу нижньої секції (Н),

α_n – кут між нижньою секцією і свердловиною (рад):

$$\alpha_n = \alpha - \alpha_{ни},$$

де $\alpha_{ни}$ – кут відхилення нижньої секції від вертикалі (рад).

За умови рівності довжин секцій, $\alpha_{ви}$ і $\alpha_{ни}$ можуть бути просто розраховані як середнє арифметичне кутів відхилення свердловини від вертикалі біля сусідніх вузлів (рис. 3.2 а).

$F_{вн.тер}$ – сила опору внаслідок внутрішнього тертя матеріалу ШН:

$$F_{вн.тер} = c_{екв} v_r,$$

де v_r – відносна швидкість між вузлами секції (м/с),

$c_{екв}$ – еквівалентний коефіцієнт опору матеріалу ШН [296, 297] (Н·с/м):

$$c_{екв} = \frac{\psi(\sigma_a) \cdot j}{2\pi \cdot \omega},$$

де j – жорсткість секції ШН (Н/м),

ω – кутова частота зовнішньої сили (рад/с);

$\psi(\sigma_a)$ – коефіцієнт поглинання матеріалу. Залежить від амплітуди напружень в штанзі σ_a (Па). Визначається експериментально із залежності

$$\psi = 2\delta,$$

де δ – логарифмічний декремент коливань:

$$\delta = \ln\left(\frac{A_0}{A_1}\right),$$

де A_0, A_1 – дві послідовні (через період) амплітуди переміщення під час гармонічних коливань, що затухають.

Для прикладу розглянемо вільні коливання колони ШН, Modelica-модель якої містить компонент «маса» і компонент «пружина-демфпер». Довжина колони – 1000 м, діаметр ШН – 0,019 м, матеріал – сталь з коефіцієнтом поглинання

матеріалу $\psi=0,01$. Маса такої колони буде $m=2228$ кг, жорсткість $j=59541$ Н/м, а еквівалентний коефіцієнт опору матеріалу

$$c_{екв} = \frac{\psi \cdot j}{2\pi \cdot \omega} = \frac{\psi \cdot \sqrt{j \cdot m}}{2\pi} \approx 18 \text{ Н}\cdot\text{с/м},$$

де $\omega = \sqrt{j/m}$ – власна частота (рад/с).

Результати моделювання (або експерименту) дозволяють розрахувати логарифмічний декремент коливань за амплітудою переміщення A_8 через 8 періодів:

$$\delta = \ln\left(\frac{A_0}{A_8}\right)/8 = \ln\left(\frac{0,01}{0,009564}\right)/8 \approx 0,005.$$

Можна прийняти наближені лінійні залежності [297] для сталі 40 $\psi(\sigma_a) = 6 \cdot 10^{-11} \sigma_a$, для склопластику $\psi(\sigma_a) = 4 \cdot 10^{-10} \sigma_a$. Якщо не враховувати зміну амплітуди напружень, то можна прийняти наближено для сталі $\psi=0,01$, для склопластику $\psi=0,08$. Реальні конструкції, які містять з'єднання, володіють значно більшими значеннями коефіцієнту ψ . Так, відповідно праці [298], для металів в пружному діапазоні напружень $\psi < 0,13$, для суцільних металічних конструкцій $\psi = 0,25 \dots 0,5$, для металічних конструкцій зі з'єднаннями $\psi = 0,38 \dots 0,88$. Застосування дефектних ШН і НКТ (у тому числі з парафіновими пробками) змінює демпфування їхніх колон [299, 300].

$F_{г.мер}$ – сила гідродинамічного опору ШН (Н):

$$F_{г.мер} = c \cdot v,$$

де c – коефіцієнт гідродинамічного опору ШН (Н·с/м). Визначається експериментально або шляхом розв'язування гідродинамічної задачі чисельними методами [63] або за допомогою емпіричної формули О.М. Піввердяна [97]:

$$c = \pi^2 \nu_{рід} \rho_{рід} M_{ум} L_{ум},$$

де $\nu_{рід}$ – кінематична в'язкість рідини ($\text{м}^2/\text{с}$),

$\rho_{рід}$ – густина рідини ($\text{кг}/\text{м}^3$),

M_{um} – коефіцієнт

$$M_{um} = \frac{1}{\ln(d_{mp} / d_{um}) \frac{(d_{mp} / d_{um})^2 + 1}{(d_{mp} / d_{um})^2 - 1}},$$

де d_{mp} – внутрішній діаметр НКТ,

d_{um} – діаметр тіла ШН,

L_{um} – довжина ШН (м).

Сума сил, що діють на плунжер насоса (рис. 3.1):

хід вверх:

$$F_{pid} + F_{zur} + F_{m.nl} + F_{кл} + F_{m.pid} - F_{pz} - F_z - F_{дин.p},$$

хід вниз:

$$-F_{m.nl} - F_{кл},$$

де F_{pid} – вага рідини (Н):

$$F_{pid} = V_{pid} \rho_{pid} g,$$

де V_{pid} – об'єм рідини над плунжером насосу (м³);

F_{zur} – сила від тиску на гирлі (Н):

$$F_{zur} = P_{zur} \cdot S_{nl},$$

де P_{zur} – тиск на гирлі (Па),

S_{nl} – площа плунжера насоса (м²);

$F_{m.nl}$ – сила тертя плунжера (Н). За емпіричною формулою В. І. Сердюка [97]:

$$F_{m.nl} = 1,84 \cdot d_{nl} / \delta_{nl} - 137,$$

d_{nl} – діаметр плунжера (м),

δ_{nl} – зазор між плунжером і циліндром (орієнтовно 0,0001 м);

$F_{кл}$ – сила гідродинамічного опору клапана (Н):

$$F_{кл} = dP_{кл} \cdot S_{кл},$$

де $dP_{кл}$ – перепад тисків на клапані (Па):

$$dP_{кл} = \zeta \cdot \frac{\rho_{pid} \cdot v_{кл}^2}{2},$$

де $v_{кл}$ – швидкість потоку в сідлі (м/с):

$$v_{кл} = \frac{Q_{кл}}{\pi \cdot d_{кл}^2 / 4}$$

де $Q_{кл}$ – об'ємна витрата рідини через клапан (м³/с),

$d_{кл}$ – діаметр отвору сідла клапана (м);

ζ – коефіцієнт місцевого опору клапана, який пов'язаний з коефіцієнтом витрат μ наступною залежністю:

$$\zeta = 1 / \mu^2.$$

Коефіцієнт витрат клапана μ може бути знайдений за числом Рейнольдса Re потоку рідини в сідлі. За даними [136] для клапанів з однією кулькою та $d_{кл}=20...25$ мм можна отримати наступну емпіричну залежність:

$$\mu = \begin{cases} 0,0846 Re^{0,2872} & , Re \leq 225; \\ 0,4 & , 225 < Re \leq 30000; \\ 0,0085 Re^{0,3764} & , 30000 < Re \leq 300000; \\ 1,0 & , Re > 300000; \end{cases}$$

де

$$Re = \frac{d_{кл} \cdot v_{кл}}{\nu_{pid}}.$$

$F_{m.pid}$ – сила тертя рідини об НКТ (Н):

$$F_{m.pid} = dP_{m.pid} \cdot S_{nl},$$

де $dP_{m.pid}$ – втрати тиску від тертя рідини в НКТ (Па):

$$dP_{m.pid} = f \frac{L_{mp}}{d_{mp}} \cdot \frac{\rho_{pid} \cdot v_{mp}^2}{2},$$

де L_{mp} – довжина НКТ (м),

v_{mp} – середня швидкість рідини в НКТ під час ходу плунжера ввєрх (м/с).
Якщо прийняти, що вона приблизно рївна швидкостї ШН v , то можна їгнорувати
силу гїдродинамїчного опору ШН пїд час ходу ввєрх $F_{z.тер2} = 0$. В їншому випадку

$$F_{z.тер2} = c \cdot (v - v_{mp}).$$

$$v_{mp} = Q / S_{mp},$$

де Q – теоретична подача насоса (м³/с),

S_{mp} – площа поперечного сїчення кїльцевого отвору мїж НКТ ї ШН (м²);

f – коефїцієнт, який залежить вїд числа Рейнольдса:

$$f = \begin{cases} f_L & , \text{Re} \leq 2000; \\ f_L + (f_T - f_L) \cdot (\text{Re} - 2000) / 2000 & , 2000 < \text{Re} < 4000; \\ f_T & , \text{Re} \geq 4000; \end{cases}$$

де $f_L = 64/\text{Re}$, $f_T = 0,316/\text{Re}^{1/4}$,

Re – число Рейнольдса:

$$\text{Re} = \frac{d_{mp} \cdot v}{\nu_{рїд}};$$

F_{pz} – вага рїдини в затрубному просторї (Н):

$$F_{pz} = V_{pz} \rho_{рїд} g,$$

де V_{pz} – об'єм рїдини в затрубному просторї (м³):

$$V_{pz} = S_{nl} (H_{шт} - H_{дин}),$$

де $H_{дин}$ – глибина динамїчного рївня (м),

$H_{шт}$ – висота колони ШН або свердловини (м).

F_3 – сила вїд тиску в затрубному просторї (Н):

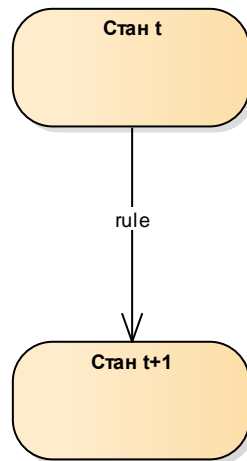
$$F_3 = P_3 \cdot S_{nl},$$

де P_3 – тиск в затрубному просторї (Па).

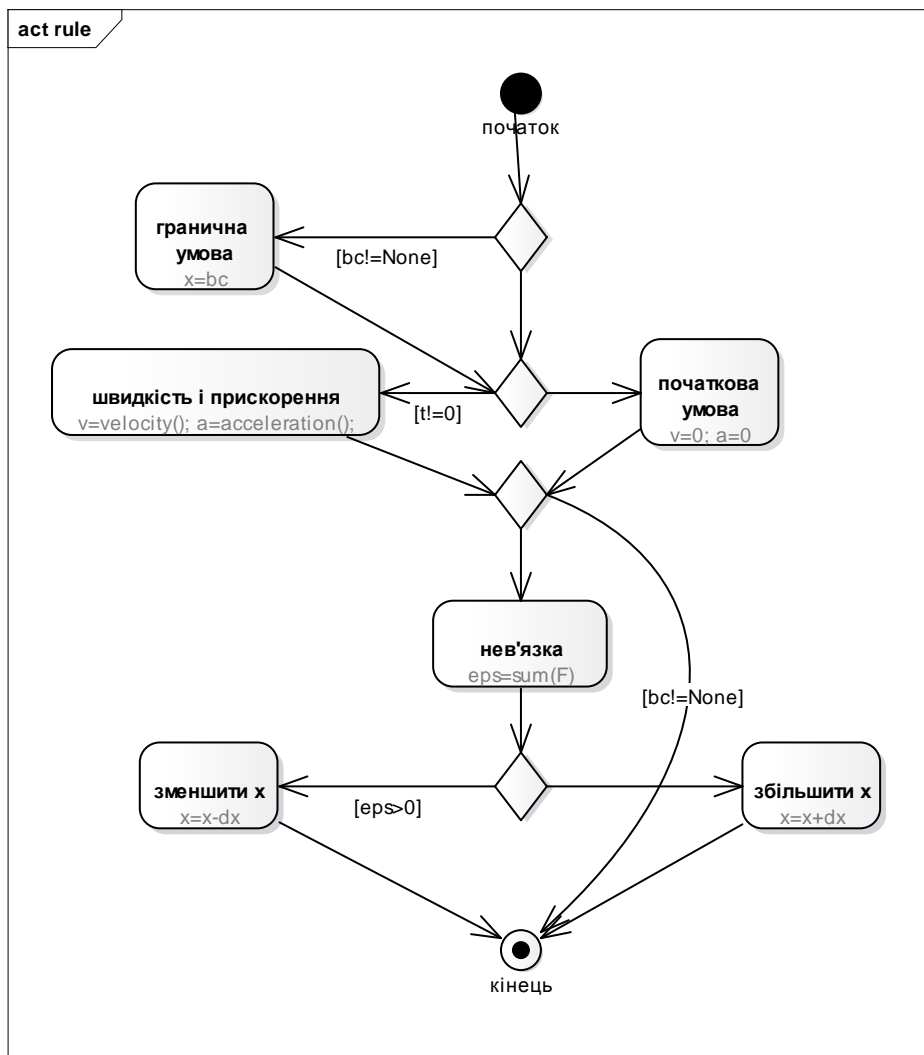
$F_{дин,p}$ – сила їнерцїї рїдини пїд час ходу ввєрх (Н):

$$F_{дин,p} = m_{рїд} \cdot a.$$

Створено модель колони ШН за допомогою системи автоматів, які з'єднані пружно-демпферними зв'язками (рис. 3.2 а) [294, 295]. Окремий автомат являє собою уявний вузол колони (рис. 3.2 б), у якому зосереджена маса секції, її вага та інші сили. Наведено код класу `Automaton` (лістинг В.2) мовою Python, який описує абстрактний автомат для моделювання руху вузла колони ШН. Автомат володіє атрибутами та функцією переходу (правилом поведінки), яка призначена для зміни стану автомата. Атрибути-дані автомата: координата x , початкова координата x_0 , попередня координата `prevx`, гранична умова `bc` (`bc=0` – нерухомий), зовнішня сила f , вага секції знизу `fs`, верхній сусід `left`, нижній сусід `right`, початкова відстань до нижнього сусіда `delta`, коефіцієнт жорсткості секції знизу k , коефіцієнт опору матеріалу секції знизу c , маса секції знизу m , прискорення a , швидкість v , попередня швидкість `prevv`, нев'язка `eps`, крок наближення координати dx , площа поперечного перетину секції знизу `area`, об'єкт `domain` класу `CA_model`, секція знизу `rSection`, кут відхилення свердловини від вертикалі біля автомата `alfa`; атрибути-функції, які обчислюють: швидкість, прискорення, сили пружності секцій зверху і знизу, сили демпфування секцій зверху і знизу, силу інерції секції, силу інерції рідини (діє на плунжер), суму нормальних сил, силу тертя секції, силу гідродинамічного опору секції, суму сил на плунжері насоса. Окремому стану відповідає множина значень атрибутів автомата. Функція-конструктор `__init__` викликається під час створення автомата і описує його атрибути зі значеннями за замовчуванням – координату, швидкість, прискорення, масу, вагу та інші. Коди реалізацій інших функцій класу показано в лістингу В.2. В додатках наведено коди програм повністю – модуль для підготовки параметрів моделей (лістинг В.1) та модель ШСНУ на основі абстрактних автоматів (лістинг В.2). Кожен автомат володіє інформацією про сусідні автомати зверху і знизу. Для обчислення швидкості та прискорення автомат запам'ятовує значення координати (`prevx`) і швидкості (`prevv`) для попереднього стану. На рис. 3а показана UML-діаграма станів автомата – орієнтований граф, в якому вершини відповідають станам, а дуга – переходу між двома станами. Стан автомата може змінитися під час зміни значення змінної часу t шляхом виклику функції `rule`, алгоритм якої наведено на рис. 3.3б.



а



б

а) – діаграма станів автомата; б) – діаграма діяльності, яка описує перехід rule

Рисунок 3.3 – UML-діаграми автомата

Опишемо функцію `rule`. Якщо задана гранична умова `bc` для цього автомата, то координаті `x` присвоюється її значення. В початковий момент часу ($t=0$), змінним швидкості та прискорення присвоюється 0. В інші моменти часу їхні значення розраховується за допомогою функцій `velocity` та `acceleration` класу `Automaton`. Після цього, якщо задана гранична умова `bc`, функція завершує своє виконання. В іншому випадку розраховуються компоненти сил, які діють на вузол. Для цього використовуються вищенаведені залежності, які реалізовані у відповідних функціях класу `Automaton`. Відповідно принципу д'Аламбера сума усіх сил у вузлі разом з силою інерції повинна бути рівною нулю. Алгоритм функції намагається знайти такий урівноважений стан. Якщо сума сил `eps` більша 0, то `x` зменшується на `dx`. Якщо вона менша 0, то `x` збільшується на `dx`.

Система автоматів (колона ШН) описується класом `CA_model` (лістинг В.2). Його функція-конструктор `__init__` створює систему автоматів. Тут створюється список автоматів, за об'єктом класу `PU` задаються значення їхніх атрибутів, створюється список значень часу та списки граничних умов, зокрема, який описує закон переміщення полірованого штока. Клас `PU` містить атрибути і функції, які визнають параметри ШСНУ (лістинг В.1). Функція `run` знаходить урівноважений стан системи автоматів в момент часу `t`. Алгоритм основної її частини такий:

ПОКИ сума нев'язок більша похибки розрахунку:

```

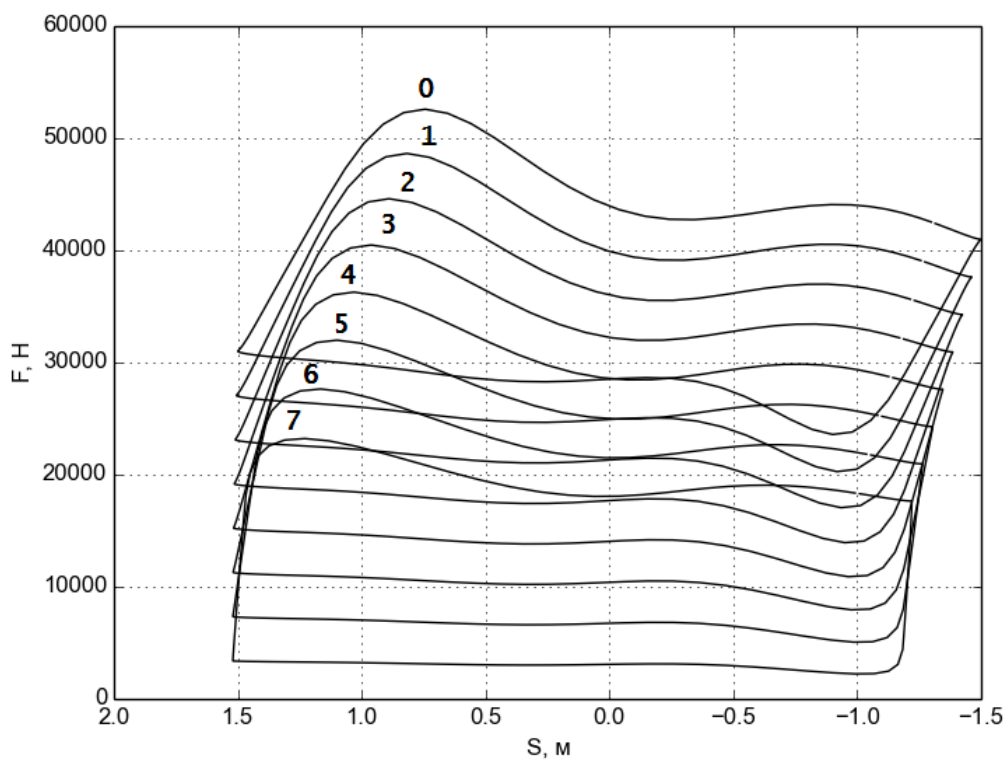
    для КОЖНОГО автомата:
        вказати крок наближення
        виконати функцію rule
    обчислити суму нев'язок

```

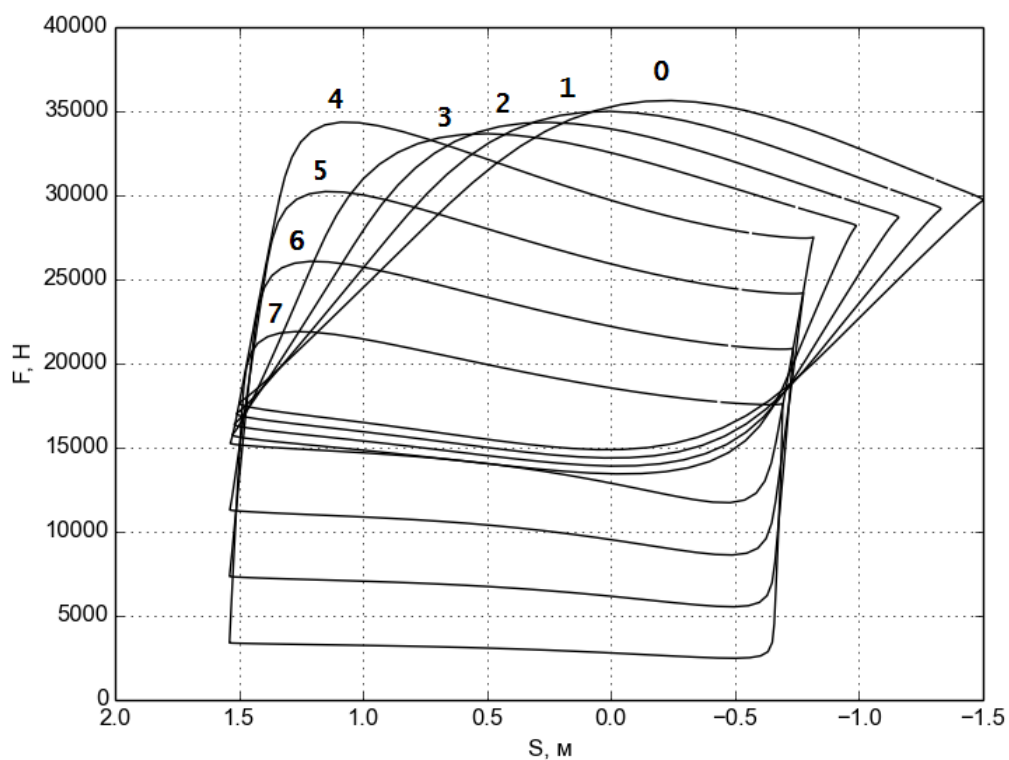
Функція `runDynamic` знаходить урівноважені стани системи автоматів для різних моментів часу. Тут для кожного значення у списку часу задаються граничні умови `bc` і зовнішні сили `f` для усіх автоматів, викликається функція `run`

та зберігаються поточні значення координати x і швидкості v у змінних $prevx$ та $prevv$ для кожного автомата. Додатково в класі `CA_model` розроблено функції, які формують списки часу та граничних умов, зберігають історію результатів, виводять довідкову інформацію, рисують графічні залежності.

Виконано симуляцію ШСНУ за допомогою розробленої програми. Для прикладу розглянуто колону довжиною 1500 м, що складається з суцільних ШН діаметром 0,019 м. Довжина ходу точки підвіски 3 м; кількість подвійних ходів за хвилину 6,5; внутрішній діаметр НКТ 0,1 м; діаметр плунжера 0,038 м; діаметр отвору сідла клапана 0,025 м; густина рідини 1000 кг/м³; кінематична в'язкість $2 \cdot 10^{-6}$ м²/с; глибина занурення насоса під динамічний рівень 100 м. Тиск на гирлі в НКТ 100 кПа, тиск в затрубному просторі 500 кПа. Вважаємо, що свердловина не має викривлень. Колону поділено на 8 секцій однакової довжини. Секції позначено індексами від 0 (верхня) до 7 (нижня). Тоді кількість вузлів (автоматів) буде рівна 9. Розглянуто колону зі сталевих ШН та колону, яка містить 50% склопластикових ШН в її верхній частині. Результати моделювання показано на рис. 3.4. Тут індексами 0..7 позначено динамограми, отримані в верхній частині відповідних секцій. Видно, що форма динамограм з індексом 0 відповідає формі практичних динамограм, отриманих на реальних ШСНУ. Порівняно динамограми з індексом 0 на рис. 3.4а і 3.4б. Для колони з склопластиковими ШН помітне зменшення навантажень в верхній її частині внаслідок зменшення її ваги та помітний сильний нахил лівої частини динамограми, який пояснюється малим коефіцієнтом пружності матеріалу склопластикових ШН. Малий коефіцієнт пружності ШН спричинює суттєве зменшення довжини ходу плунжера, про що свідчить результат порівняння плунжерних динамограм з індексом 7. Результати моделювання з вищими частотами подвійних ходів показали збільшення довжини ходу плунжера, але і зросла амплітуда напружень. Звідси виникає необхідність оптимізації конструкції колони з склопластиковими ШН, довжини ходу полірованого штоку та частоти подвійних ходів.



а



б

а) – сталеві ШН (100%); б) – склопластикові (50%) і сталеві (50%) ШН

Рисунок 3.4 – Динамограми для колони ШН

3.2 Імітаційна модель ШСНУ мовою Modelica

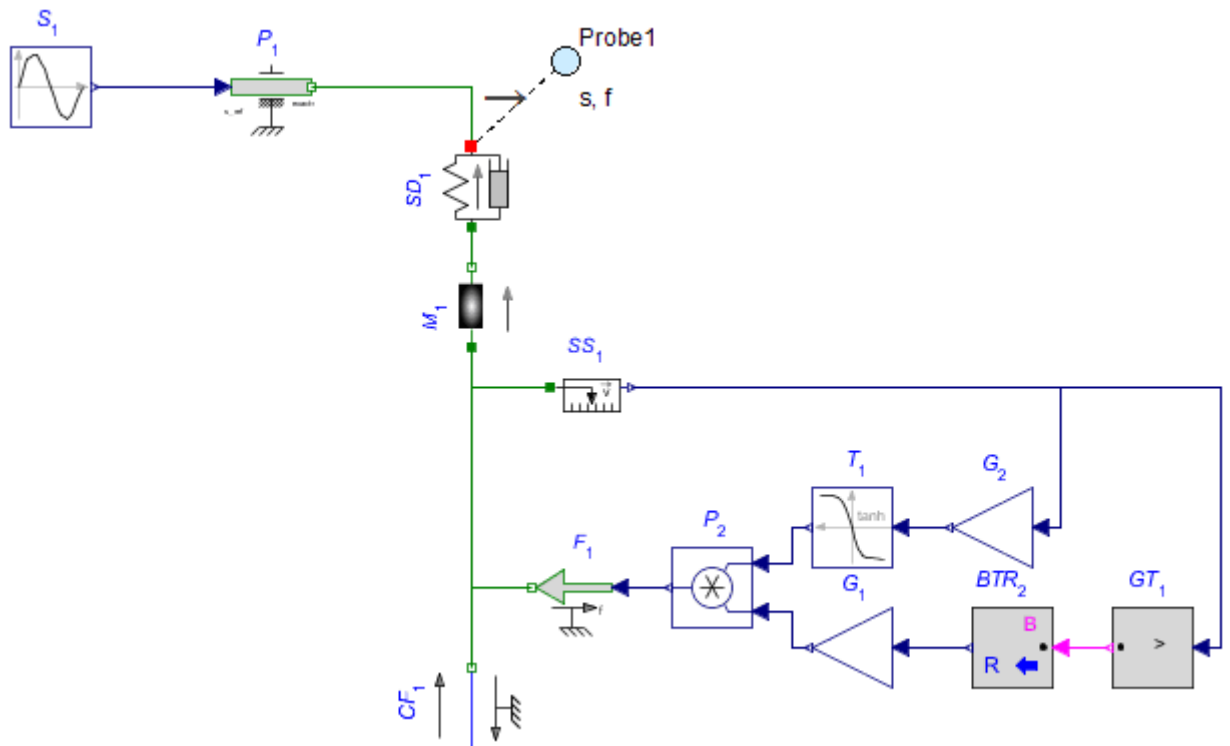
3.2.1 Принципи побудови моделі ШСНУ мовою Modelica

Нижче описано принципи побудови моделі ШСНУ мовою Modelica для Maplesim [24, 26, 301]. Під час побудови моделі мовою Modelica секцією колони ШН будемо називати частину колони, яка володіє такими атрибутами, як довжина секції, діаметр ШН, матеріал ШН, кут відхилення свердловини внизу секції від вертикалі. Посекційне моделювання дає змогу отримати колону з різними властивостями по довжині. Вузлом будемо називати точку з'єднання секцій колони ШН. В основному для розрахунку компонентів загального навантаження, які діють на колону ШН та плунжер насоса, використано працю [97].

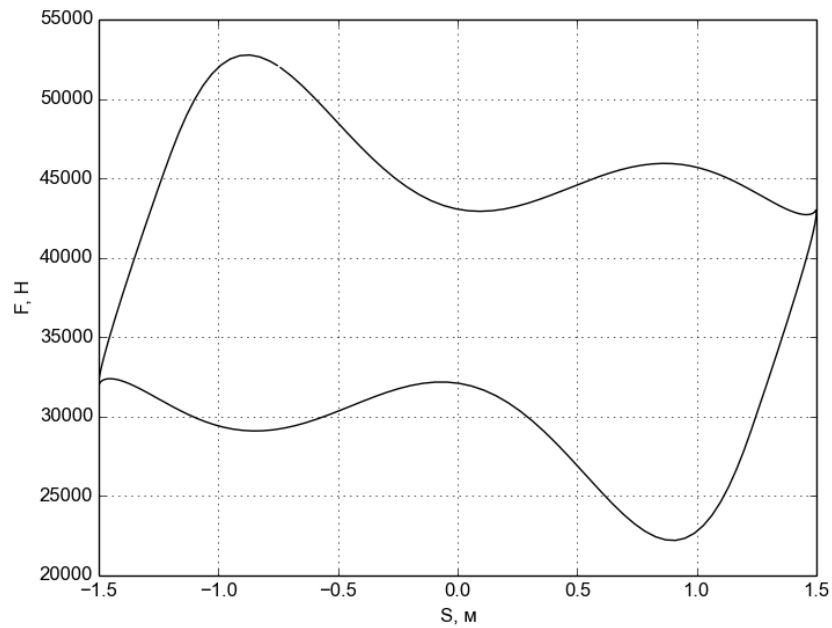
Для прикладу розроблено модель ШСНУ мовою Modelica (середовище MapleSim), яка має наступні параметри. Колона довжиною 1500 м складається зі сталевих суцільних ШН діаметром 0,019 м. Довжина ходу точки підвіски – 3 м, кількість подвійних ходів за хвилину – 6,5, внутрішній діаметр НКТ – 0,1 м, діаметр плунжера – 0,038 м, діаметр отвору сідла клапана – 0,025 м, густина рідини – 1000 кг/м^3 , кінематична в'язкість – $2 \cdot 10^{-6} \text{ м}^2/\text{с}$, глибина занурення насоса під динамічний рівень – 100 м. Тиск на гирлі в НКТ та тиск в затрубному просторі не враховуємо. Спочатку розроблено спрощену модель цієї ШСНУ (рис. 3.5а).

Нижче перелічено компоненти цієї моделі, їхні постійні параметри та змінні. Змінна t – це час (с). Символи $_u(t)$, та $_u2(t)$ в кінці назв змінних використовуються для вхідних сигналів компонента, а символи $_y(t)$ – для вихідних сигналів.

SI – генератор синусоїдального сигналу. Для спрощення моделі ШСНУ допустимо замінити модель механізму ВК простим генератором синусоїдального переміщення. Повна модель механізму ВК наведена в праці [301]. Параметри цього компонента:



а



б

Рисунок 3.5 – Спрощена модель ШСНУ (а) та її динамограма (б)

$SI_amplitude$ – амплітуда. Рівна половині ходу полірованого штока (м);

SI_freqHz – частота (Гц). Рівна кількості подвійних ходів полірованого штока за секунду;

SI_phase – фаза (рад).

PI – 1-D поступальне переміщення за заданим вхідним сигналом. Використовується для моделювання переміщення полірованого штока. В даному випадку значення його параметра $exact=true$.

SDI – лінійні 1-D пружина і демпфер. Змінна $SDI_s_rel(t)$ – відносна відстань між фланцями компонента, $SDI_v_rel(t)$ – відносна швидкість між фланцями компонента. Використовується для моделювання пружно-демпферних властивостей колони ШН. Параметри:

SDI_c – коефіцієнт пружності (Н/м),

SDI_d – коефіцієнт демпфування (опору) (Н·с/м). Якщо враховувати тільки демпфування матеріалу колони ШН, то еквівалентний коефіцієнт опору матеріалу ШН [296, 297] (Н·с/м):

$$c_{екв} = \frac{\psi \cdot j}{2\pi \cdot \omega},$$

де j – жорсткість секції ШН (Н/м),

ω – кутова частота зовнішньої сили (рад/с);

ψ – коефіцієнт поглинання матеріалу, який визначається експериментально.

MI – маса з поступальним переміщенням. Використовується для моделювання інерції колони ШН. Параметри:

MI_m – значення маси (кг).

$CF1$ – поступальна постійна сила. Використовується для моделювання ваги колони ШН. Параметри:

$CF1_f_constant$ – значення сили (Н).

SSI – сенсор абсолютної швидкості v . Змінна $SS_flange_s(t)$ – переміщення в точці вимірювання. В даному випадку потрібен для моделювання ваги рідини під час ходу вверх (коли $v>0$).

$GT1$ – генерує сигнал $GT1_y(t)$ рівний логічному $True$, якщо вхідний сигнал $GT1_u(t)$ більше значення порогу (в даному випадку 0).

$BTR2$ – конвертує логічний сигнал в дійсний.

$G1$ – добуток вхідного сигналу і константи. Параметри:

$G1_k$ – значення константи. Рівне вазі рідини (Н).

$G2$ – добуток вхідного сигналу і константи. Використовується разом з компонентом $T1$ для згладжування сигналу біля точки $v=0$. Без компонентів $G2$ та $T1$ розв'язок задачі не можливо отримати. Параметри:

$G2_k$ – значення константи. Прийmemo $G2_k=100$.

$T1$ – гіперболічний тангенс вхідного сигналу. Вихідний сигнал $T1_y(t)$ буде рівний $v \cdot G2_k$.

$P2$ – добуток двох вхідних сигналів – $P2_u(t)$ та $P2_u2(t)$.

$F1$ – поступальна сила за вхідним сигналом.

Детальніший опис цих компонентів наведено в керівництві користувача MapleSim [165]. Цій моделі відповідає наступна система диференціально-алгебраїчних рівнянь, яка автоматично згенерована MapleSim:

$$\frac{P2_u2(t)}{G1_k} = \begin{cases} 1 & GT1_y(t) = true \\ 0 & otherwise \end{cases}$$

$$SD1_s_rel(t) + SS_flange_s(t) - \left(\begin{cases} 0 & t < 0 \\ S1_amplitude \sin(2 \pi t S1_freqHz + S1_phase) & otherwise \end{cases} \right) = 0$$

$$GT1_y(t) = (0 < GT1_u(t))$$

$$P2_y(t) = T1_y(t) P2_u2(t)$$

$$T1_y(t) = \tanh(G2_k GT1_u(t))$$

$$\frac{d}{dt} GT1_u(t) = \frac{SD1_c SD1_s_rel(t) + SD1_d SD1_v_rel(t) + P2_y(t) + CF1_f_constant}{M1_m}$$

$$\frac{d}{dt} SD1_s_rel(t) = SD1_v_rel(t)$$

$$\frac{d}{dt} SS_flange_s(t) = GT1_u(t)$$

В цьому випадку $CF1_f_constant = -29205$ Н, $G1_k = -16688$ Н, $M1_m = 3402$ кг, $S1_amplitude = 1,5$ м, $S1_freqHz = 0,108$ Гц, $SD1_c = 39694$ Н/м, $SD1_d = 4641$ Н·с/м. Оскільки тут $SD1$ є єдиним компонентом, який використовується для

моделювання демпфування колони, то коефіцієнт ψ враховує усі види демпфування. Шляхом порівняння графіків вільних коливань (верхня та нижня частина динамограми) моделі та реальної колони знайдено його орієнтовне значення $\psi=0,5$. Результати моделювання показані на рис. 3.5б.

В спрощеній моделі не враховано окремо такі фактори, як сили зовнішнього тертя (ШН та плунжера насоса), сили гідродинамічного опору (ШН, клапанів та рідини), сила від тиску на гирлі, вага рідини в затрубному просторі, сила від тиску в затрубному просторі, сила інерції рідини під час ходу угору. В даному випадку модель колони ШН не відповідає моделі пружного стрижня з рівномірно розподіленою масою. Неможливо також моделювати ступінчасті колони і нерівномірно розподілені сили опору. Нижче розглянуто шляхи удосконалення цієї моделі.

Побудовано гідравлічну частину моделі та досліджено її роботу на спрощеній моделі ШСНУ (рис. 3.6а). Колону ШН тут слід вважати абсолютно жорстким стрижнем без маси і ваги. Нижче перелічено компоненти цієї моделі та їхні параметри.

НС1 – конвертує поступальну механічну енергію в гідравлічну енергію і навпаки. Використовується для моделювання штангового насоса. Параметри:

НС1_A – площа поперечного перетину плунжера (м^2).

F1 – зафіксований фланець з відсутнім поступальним переміщенням;

AP1, AP2 – гідравлічні вузли з заданим тиском. Моделюють відповідно вхід насоса та викидну лінію на гирлі.

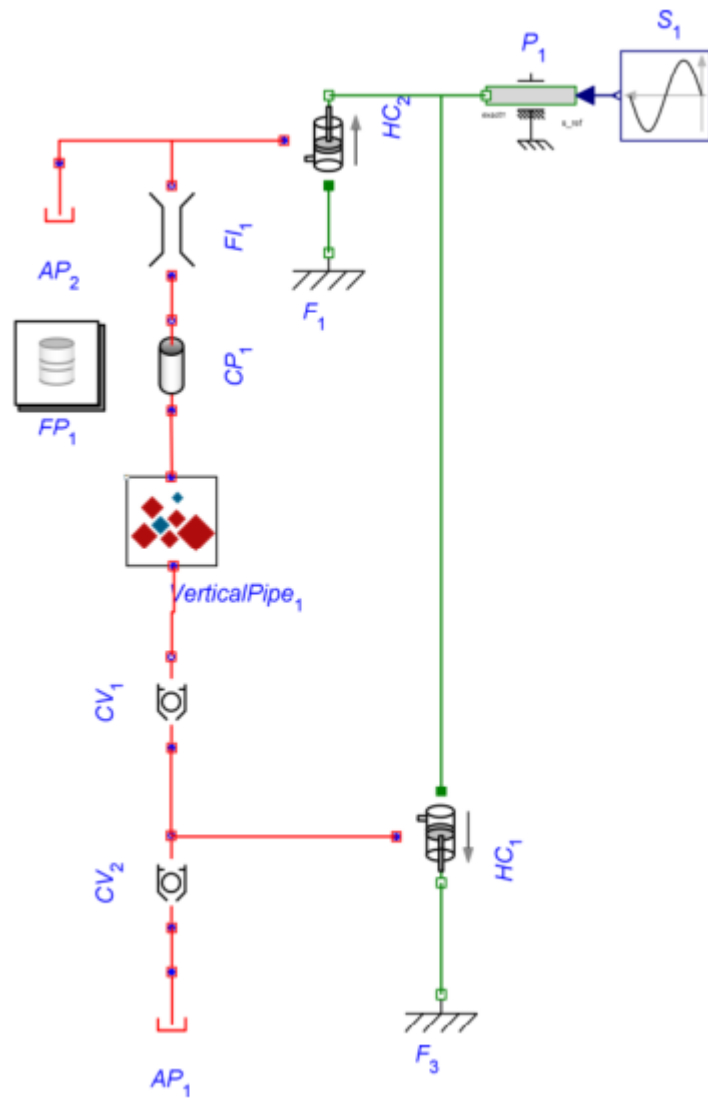
AP1_P – тиск на прийомі насоса (Па),

AP2_P – тиск у викидній лінії на гирлі (Па).

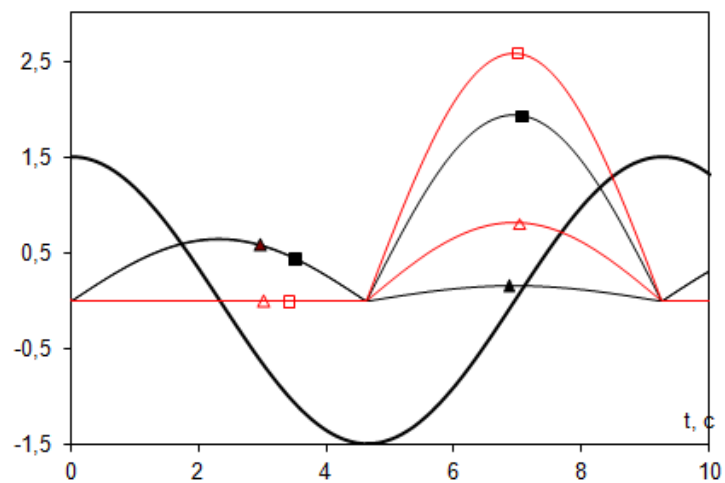
CV1, CV2 – зворотні клапани. Моделюють клапани свердловинного насоса.

Параметри:

CV1_Ropen, CV2_Ropen – коефіцієнти місцевого опору відкритих клапанів ($\text{Па}\cdot\text{с}/\text{м}^3$).



а



б

— – переміщення полірованого штока (м);

▲, ■ – подача на гирлі; Δ, □ – подача насоса ($0,001 \cdot \text{м}^3/\text{с}$)

Рисунок 3.6 – Спрощена модель ШСНУ з гідравлічною частиною (а) та її залежності (б) від часу для насосів діаметром 32 мм (▲, Δ) та 57 мм (■, □)

Точніші моделі клапанів та насоса можна побудувати за допомогою компонентів додаткової платної бібліотеки Modelon's Hydraulics Library [302]. Точні значення коефіцієнтів місцевого опору клапанів можна визначити експериментально або обчислити за допомогою МСЕ [110].

VerticalPipe1 – компонент для моделювання різниці гідростатичних тисків $dP(t)$ в вертикальній НКТ, яка спричинена вагою рідини. Цей компонент відсутній в стандартній бібліотеці компонентів MaplseSim. Для його створення можна скористатись засобом MaplseSim "Custom Component". Потрібно ввести назву нового компоненту, рівняння, які описують поведінку цього компоненту

$$eq := [dP(t) = rho * g * z, Pin(t) - Pout(t) = dP(t), Qin(t) = Qout(t)],$$

параметри зі значеннями за замовчуванням

$$params := [z = 0, rho = 1000, g = 9.81],$$

та початкові умови

$$initialconditions := [].$$

Тут параметри rho – густина рідини, g – прискорення вільного падіння, z – висота труби; $Pin(t)$, $Pout(t)$ – змінні тиску на вході та виході; $Qin(t)$, $Qout(t)$ – змінні об'ємної витрати на вході та виході.

Далі потрібно створити два гідравлічні порти – вхідний зі змінними $Pin(t)$, $Qin(t)$ та вихідний зі змінними $Pout(t)$, $Qout(t)$. Після цього MaplseSim згенерує код нового компоненту мовою Modelica.

Замість компонента *VerticalPipe1* можна застосувати стандартний компонент для створення постійного тиску *FP* (Fixed Pressure Source). Якщо *FP* використовується тільки для моделювання гідростатичного тиску рідини в затрубному просторі, то цей компонент повинен бути розміщений внизу компонента *VerticalPipe1* і мати від'ємний тиск.

CP1 – компонент для моделювання втрат тиску від тертя рідини в трубі з круглим отвором. Основні параметри:

CP1_D – діаметр отвору труби (м),

CPI_L – довжина труби (м),

$CPI_epsilon$ – висота мікронерівностей поверхні труби (м),

CPI_ReL , CPI_ReT – максимальні значення числа Рейнольдса для ламінарного та турбулентного режимів.

FII – компонент для моделювання ефекту інерції рідини – змін тиску, спричинених змінами швидкості рідини. Параметри:

FII_A – площа поперечного перетину труби (m^2),

FII_L – довжина труби (м).

FPI – компонент з такими параметрами рідини:

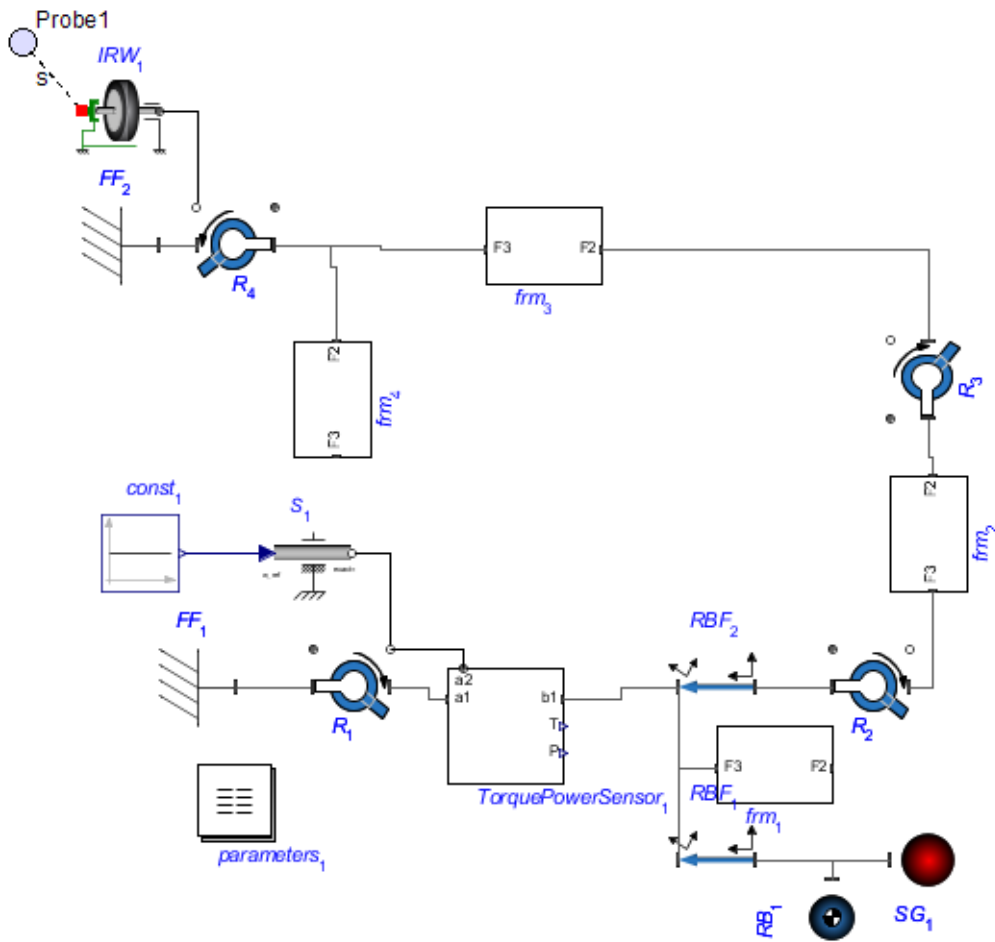
$rhoFluid$ – густина (kg/m^3),

$nuFluid$ – кінематична в'язкість (m^2/s).

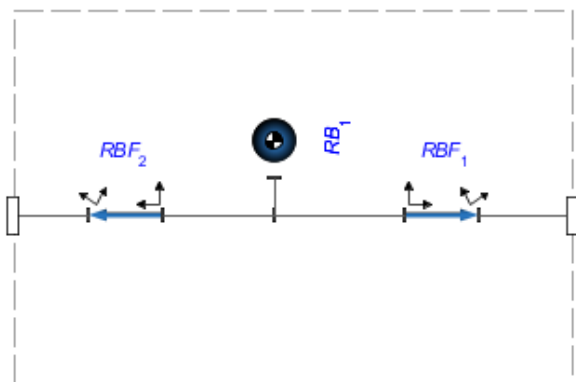
Ці параметри використовуються в компонентах *VerticalPipe1*, *CPI*, *FII*.

Відомо [143], що під час ходу вгору полірований шток вивільняє певний об'єм рідини з НКТ, а під час ходу вниз додає цей же об'єм. Завдяки цьому об'єму подача на гирлі під час ходу вгору буде меншою ніж подача насоса, а під час ходу вниз – більшою. Для моделювання цього явища в модель можна додати ще один компонент – гідравлічний циліндр *HC2*, параметер *HC2_A* якого повинен бути рівний площі поперечного перетину штока. Розглянемо модель з діаметром полірованого штока 28,6 мм. В цьому випадку $HC2_A = 0,00064 m^2$; $AP1_P = 0$ Па; $AP2_P = 0$ Па; $CV1_Ropen = CV2_Ropen = 176,36$ Па·с/ m^3 ; $FII_A = 0,00785 m^2$; $rhoFluid = 1000$ кг/ m^3 ; $nuFluid = 0,2e-5$ m^2/s . Результати моделювання цієї моделі відображені на рис 3.6б. Помітно, що у разі застосування насосів малого діаметра подача на гирлі під час ходу вниз буде більшою ніж під час ходу вгору. Компонент *HC2* слід додавати в модель, якщо цілтю моделювання є графік подачі на гирлі та моделюються насоси малого і середнього діаметра.

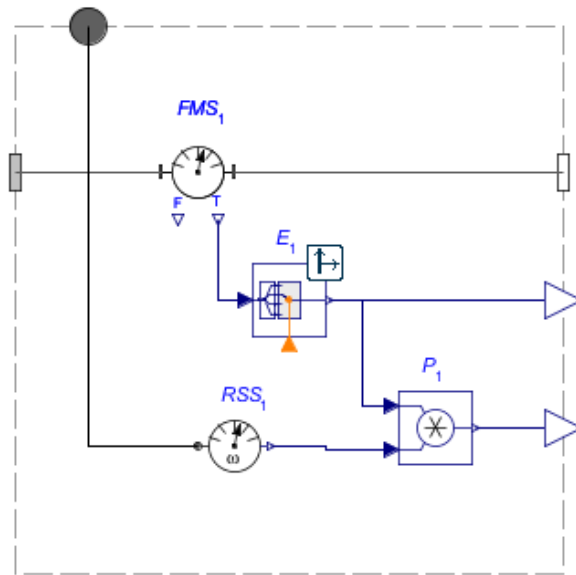
Побудовано модель механізму ВК (рис. 3.7) [301]. Компоненти MapleSim Multibody [165] дозволяють побудову моделей плоских і просторових механізмів.



a



б



в

б) – жорстка ланка механізму з моментом інерції та зосередженою в центрі масою;

в) – сенсор крутного моменту та потужності на вихідному валу редуктора

Рисунок 3.7 – Модель ВК (а) та її частини (б, в)

Введемо глобальну прямокутну систему координат. Її вісь Z співпадає з віссю вихідного вала редуктора, вісь X горизонтальна, вісь Y вертикальна. Перелічимо компоненти цієї моделі та їхні параметри.

$const_1$ – генератор постійного сигналу. Параметр k відповідає кутовій швидкості кривошипів.

S_1 – кутова швидкість фланця відповідно до вхідного сигналу. Параметр $exact=true$.

FF_1, FF_2 – стаціонарні фрейми з відсутнім переміщенням і орієнтацією відносно глобальної системи координат. Використовуються для моделювання опори кривошипів та опори балансира відповідно. Основні параметри:

\bar{r}_{xyz} – вектор зміщень відносно глобальної системи координат,

$\bar{\theta}_{\zeta\eta\xi}$ – вектор поворотів відносно глобальної системи координат.

R_1, R_2, R_3, R_4 – з'єднання, які дозволяють одну обертову ступень вільності навколо вибраної осі. Вісь задається параметром \hat{e}_1 . В даному випадку кожне це з'єднання дозволяє поворот навколо осі, яка паралельна осі Z , тобто $\hat{e}_1 = \langle 0,0,1 \rangle$.

$frm_1, frm_2, frm_3, frm_4$ – компоненти для моделювання жорстких ланок механізму ВК з моментом інерції та зосередженою в центрі масою (двох кривошипів без противаг, двох шатунів, заднього плеча балансира з траверсою, переднього плеча балансира з головкою відповідно).

Кожен такий компонент складається з двох компонентів RBF (фрейм жорсткого тіла) та одного RB (центр мас жорсткого тіла). Наприклад, розглянемо модель кривошипів frm_1 (рис. 3.7). Параметри \bar{r}_{xyz} компонентів RBF задають довжину кривошипа та розташування центра мас. Для RBF_2 $\bar{r}_{xyz} = [-L/2, 0, 0]$, а для RBF_1 $\bar{r}_{xyz} = [L/2, 0, 0]$, де L – довжина кривошипа (м). Для компоненту RB_1 слід ввести значення параметра m (маса кривошипів, кг) та $[I]$ (матриця тензора інерції в точці центра мас, $\text{кг}\cdot\text{м}^2$):

$$[I] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_z \end{bmatrix},$$

де I_z – момент інерції двох кривошипів відносно осі, яка проходить через центр мас і паралельна осі Z .

RBF_1 – фрейм жорсткого тіла, довжина якого рівна радіусу центра мас противаг.

RB_1 – центр мас жорсткого тіла (усіх противаг). Основний параметр – сумарна маса противаг m (кг). Масу противаг вважаємо зосередженою в точці, тому моментом інерції противаг відносно осі, яка паралельна осі Z і проходить через центр мас противаг, нехтуємо $I_z=0$.

RBF_2 – фрейм жорсткого тіла, довжина якого рівна відстані від осі обертання кривошипів до осі з'єднання кривошипів з шатунами.

IRW_1 – ідеальне колесо без тертя чи інерції. Використовується для перетворення обертового руху балансира в поступальний рух точки підвіски ШН. Параметр *radius* (м) відповідає довжині переднього плеча балансира з головкою.

TorquePowerSensor₁ – сенсор крутного моменту та потужності на вихідному валу редуктора ШСНУ (рис. 3.7).

FMS₁ – сенсор сили та моменту.

E_1 – компонент для виділення одного сигналу з вектора сигналів. Використовується для виділення сигналу крутного моменту навколо осі Z . Параметр n_{in} – кількість вхідних сигналів. В даному випадку є три сигнали крутних моментів навколо осей X, Y, Z . Щоб виділити крутний момент навколо осі Z необхідно задати початкову умову $value_0=3$.

RSS₁ – сенсор кутової швидкості (рад/с).

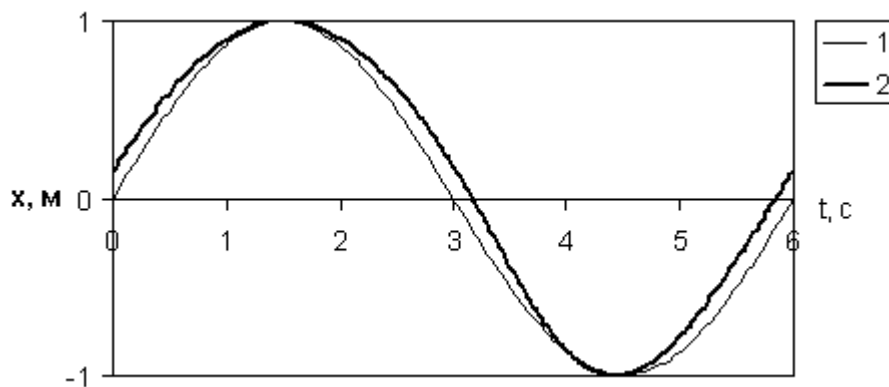
P_1 – добуток сигналів.

parameters₁ – компонент, який містить значення змінних параметрів моделі. Використовується для спрощення зміни значень параметрів за допомогою програми, що використовує інтерфейс програмування MapleSim.

SG_1 – сферична геометрія. Використовується тільки для візуалізації руху противаг в 3D Playback.

Результат моделювання ВК СКД8-3-4000 за умови $S=2$ м, $\omega=1,047$ рад/с показаний на рис. 3.8. Жирною лінією показана залежність переміщення точки

підвіски від часу (Probe1). Відомо, що для ВК з від'ємним дизаксіалом (типу СКД) та обертанням кривошипів за годинниковою стрілкою (гірло знаходиться зліва від спостерігача) хід вгору відбувається дещо довше [303]. Але з рис. видно, що реальна залежність мало відрізняється від функції $x=(S/2)*\sin(\omega*t)$. Тому для спрощення моделі ШСНУ допустимо замінити модель механізму ВК простим генератором синусоїдального переміщення, як це було зроблено вище.



1 – функція $x=(S/2)\cdot\sin(\omega\cdot t)$; 2 – реальна залежність для СКД8-3-4000

Рисунок 3.8 – Порівняння залежностей переміщення точки підвіски від часу

Розширена модель ШСНУ (рис. 3.9) містить компоненти, які є у спрощеній моделі та у моделі гідравлічної частини, а також такі додаткові компоненти:

TF_1 – тертя в контактї між тілами, що ковзають. Цей компонент дозволяє застосовувати розширену модель тертя (Кулона, Штрибека та в'язкого тертя) [165]. Застосовується для моделювання тертя плунжера насоса об стінки циліндра.

Параметри:

f_s – максимальна сила тертя, Н;

f_c – сила тертя Кулона, Н;

d – коефіцієнт в'язкого тертя, Н·с/м. Силу в'язкого тертя не враховуємо ($d=0$);

v_s – коефіцієнт швидкості ковзання Штрибека (приймаємо $v_s=0,1$ м/с);

n – коефіцієнт кривої Штрибека (приймаємо $n=1$);

v_0 – параметр функції згладжування $th(\frac{v}{v_0})$ (приймемо $v_0=0,01$ м/с).

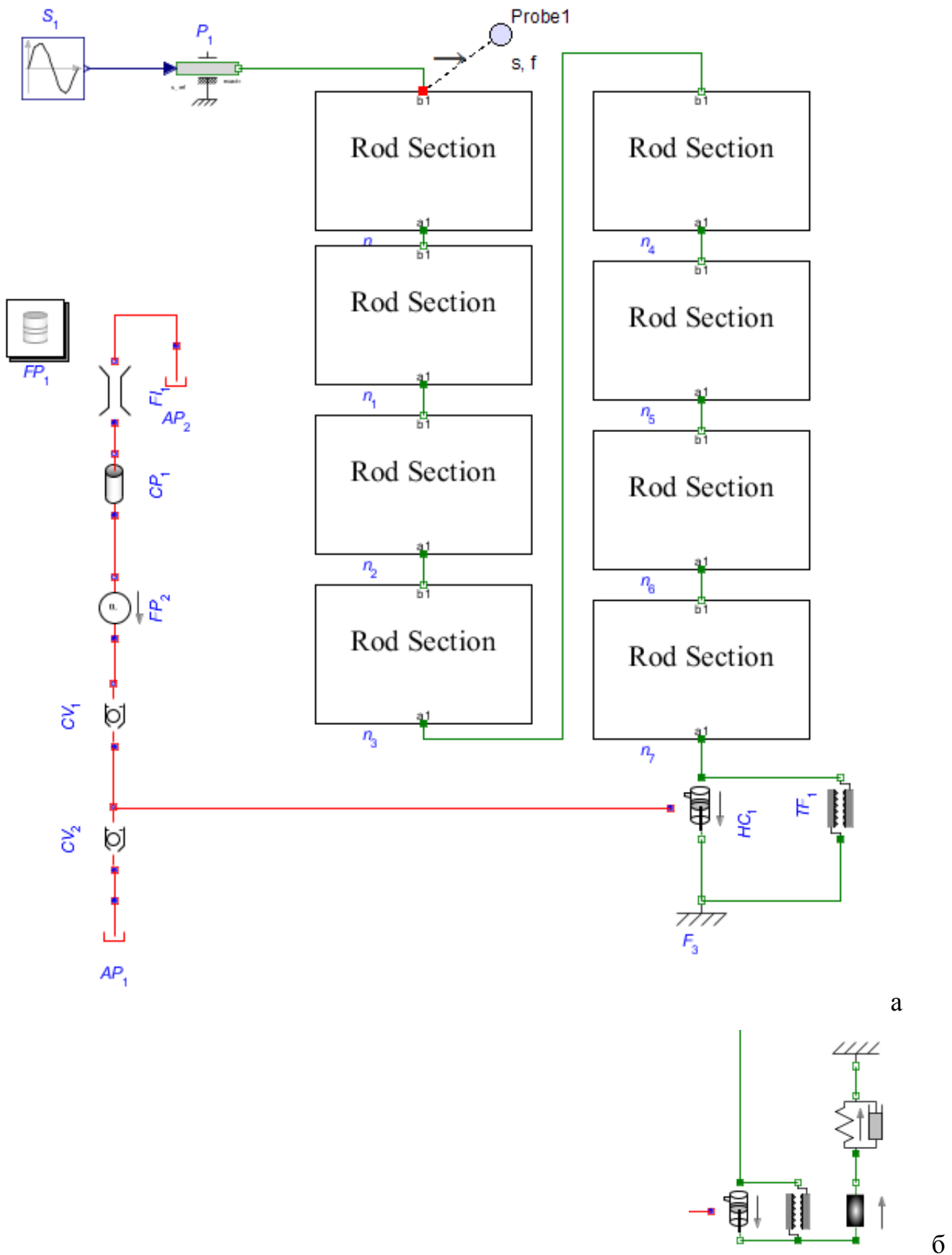


Рисунок 3.9 – Розширена модель ШСНУ (а) та компоненти для моделювання пружно-демпферної НКТ (б)

n_0, n_1, n_2 – секції колони ШН. Для кожної секції (рис. 3.10) моделюються пружні та демпферні властивості, маса, вага в рідині, сили тертя об стінки в НКТ та сила гідродинамічного опору під час ходу вниз. Компоненти M_1, SD_1, CF_1 мають ті ж функції, що і у спрощеній моделі, але стосуються одної секції. Далі описано інші компоненти секції:

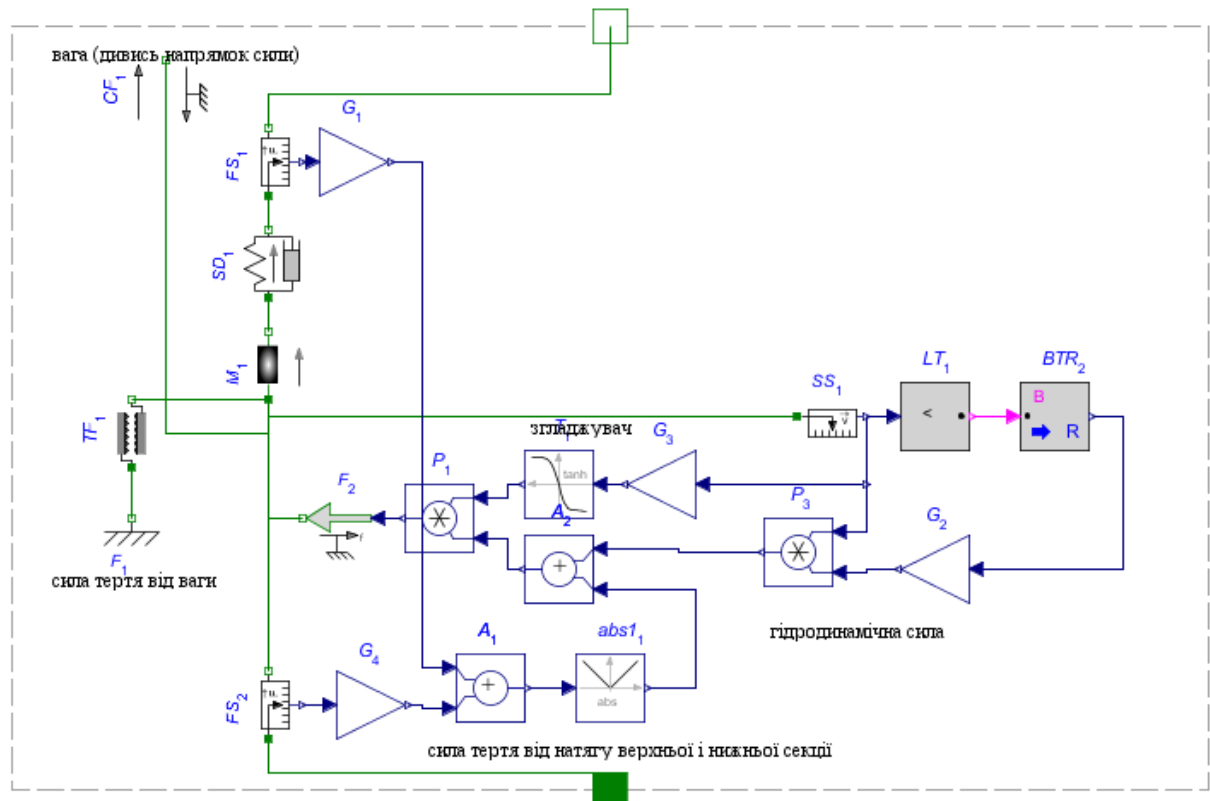


Рисунок 3.10 – Модель секції колони ШН

F_1 та TF_1 використовуються для моделювання сили тертя секції об НКТ, яка виникає від ваги секції в похилій свердловині. Їх можна також використовувати для моделювання сили гідродинамічного тертя ШН, знак якої завжди протилежний швидкості руху ШН. У цьому випадку значення коефіцієнта в'язкого тертя компонента TF_1 повинно бути узгоджено зі значенням множника k компонента G_2 . Порівняння моделей з практичними динамограмами показало, що значення цих коефіцієнтів повинно бути орієнтовно в 10 раз більшим значень, розрахованих за формулою Пірвердяна [97]. Зазвичай гідродинамічне тертя ШН найбільше під час руху їх вниз. Тоді слід використовувати тільки компонент G_2 .

FS_1, FS_2 – сенсори сили натягу верхньої та нижньої секції відповідно.

G_1 виводить добуток сили натягу верхньої секції та константи $k \cdot \sin \alpha_e$, де k – коефіцієнт тертя, α_e – кут між свердловиною і верхньою секцією (рад) (рис. 3.2б).

G_4 виводить добуток сили натягу нижньої секції та константи $k \cdot \sin \alpha_n$, де α_n – кут між свердловиною та нижньою секцією (рад) (рис. 3.2б).

A_1 виводить суму сил тертя від натягу верхньої та нижньої секцій, а $abs1_1$ виводить абсолютне значення вхідного сигналу. Таким чином, на виході $abs1_1$ буде сумарна сила тертя від натягу верхньої та нижньої секцій.

SS_1 – сенсор швидкості нижнього вузла секції.

LT_1 виводить логічну істину (*true*), якщо значення вхідного сигналу менше значення параметра *threshold*. Тут *threshold*=0.

BTR_2 конвертує булевий сигнал в дійсний. В даному випадку – *true* в 1, а *false* в 0.

G_2 виводить добуток вхідного сигналу та значення параметра k , який рівний коефіцієнту гідродинамічного опору секції.

P_3 виводить добуток швидкості та коефіцієнта гідродинамічного опору секції. Результатом є сила гідродинамічного опору, яка діє під час ходу ШН вниз ($v < 0$).

A_2 виводить суму сили гідродинамічного опору та сили тертя від натягу.

Компоненти G_3 і T_1 необхідні для згладжування залежності сили від швидкості біля нуля. G_3 виводить добуток вхідного сигналу v та значення параметра k , який рівний $1/v_0$. T_1 виводить гіперболічний тангенс вхідного сигналу v/v_0 .

P_1 виводить добуток сигналу сумарної сили та сигналу функції згладжування. Вихідний сигнал поступає на вхід компонента F_2 .

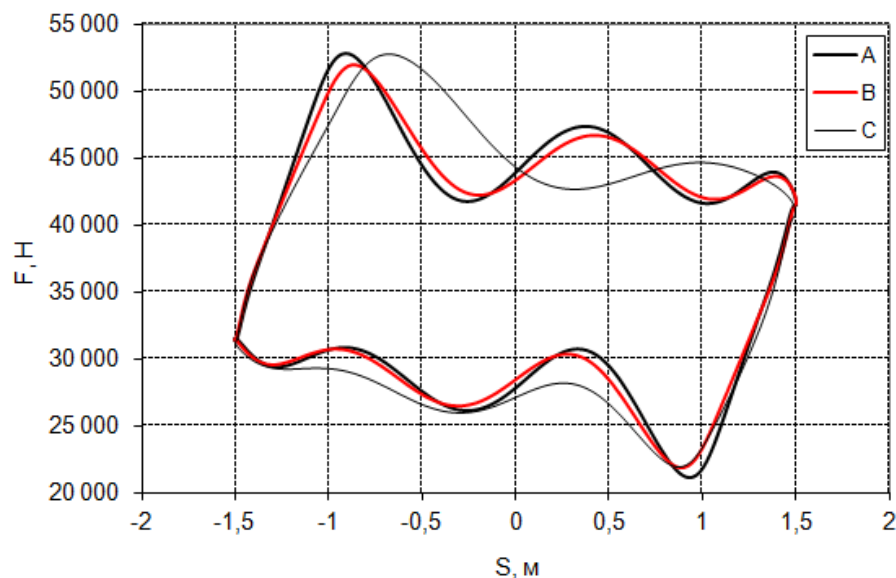
F_2 створює поступальну силу відповідно до вхідного сигналу. Таким чином, F_2 призначений для створення сили гідродинамічного опору під час руху ШН вниз та сили тертя, яка виникає внаслідок натягу верхньої та нижньої ШН в похилій свердловині.

В окремих випадках доцільним є застосування компоненту *Hardstop* (поступальний механізм, що обмежує рух на заданій довжині), розташованого

відразу після компоненту P_1 , для моделювання розбіжності руху полірованого штока і балансира, яке можливе, для прикладу, під час відкачування в'язких рідин, СПУ, надмірному зтягненні гирлового ущільнення [301].

Для симуляції руху НКТ, нижня частина якої не закріплена, потрібно між компонентами F_3 "Нерухомий фланець" і HC_1 "Гідравлічний циліндр" додати поступальні одновимірні компоненти "Пружина-демпфер", що моделює пружно-демпферні властивості НКТ, та "Маса", що моделює її інерцію (рис. 3.9б).

Розглянемо ШСНУ з наведеними вище параметрами. З рис. 3.11 помітно, що застосування НКТ великого діаметра (з великою жорсткістю) не суттєво впливає на форму динамограми – дещо збільшується кут її бічних сторін і дещо зменшуються екстремальні навантаження. Але застосування НКТ малого діаметра значно змінює форму динамограми – значно збільшується кут нахилу і зростає демпфування, спричинене гідродинамічним опором. Таким чином, якщо нижня частина НКТ не закріплена і застосовуються НКТ з малим діаметром, то потрібно застосовувати модель з пружно-демпферною НКТ.



A – без симуляції руху НКТ; B, C – з симуляцією руху НКТ: діаметром 114 мм і товщиною стінки 7 мм (B), діаметром 60 мм і товщиною стінки 5 мм (C)

Рисунок 3.11 – Динамограми ШСНУ

Розроблена модель є параметричною і гнучкою, тобто можна легко змінювати значення окремих параметрів і розширювати можливості моделі шляхом додання потрібних компонентів. MapleSim має можливість компіляції моделі у файл з машинним кодом для пришвидшення обчислень. Це дозволяє організувати високопродуктивні ітераційні обчислення для пошуку оптимальних значень параметрів. Код моделі без анотацій мовою Modelica показано в лістингу Г.1.

Для полегшення зміни значень параметрів моделі використовували мову програмування Python та інтерфейс прикладного програмування MapleSim API, який дозволяє іншим програмам взаємодіяти з моделями MapleSim. Розроблені автором компоненти САПР ШСНУ (лістинги В.1, Г.1-3), яка базується на системах Maple і MapleSim, показані на рис. 3.12. Клас PU описує всю ШСНУ (лістинг В.1). Цей клас містить об'єкти класів SuckerRodString, Well, Pump та PumpingUnit, які описують відповідно колону ШН, колону НКТ, насос та ВК. Клас SuckerRodString (лістинг В.1) містить атрибут `items` – список об'єктів класу SuckerRod, який описує секцію однотипних ШН. Клас Maplesim_model (лістинг Г.2) описує модель ШСНУ у MapleSim. Він використовує об'єкти класів PU та MapleInterface4Maplesim і потребує файлу моделі MapleSim. За допомогою класу Maplesim_model (див. функцію `optimize`) можна виконувати оптимізацію конструкції ШСНУ шляхом ітерації, яка включає зміну параметрів моделі, розрахунок та читання результатів. Клас MapleInterface4Maplesim описує інтерфейс Python-MapleSim (лістинг Г.3). Він дозволяє змінювати параметри моделі MapleSim та читати результати моделювання. Він успадковує клас MapleInterface, який дозволяє Python згенерувати код програми Maple і виконати його шляхом створення окремого процесу `smarle.exe`. Maple і MapleSim взаємодіють за допомогою MapleSim API. Maple зберігає результати моделі MapleSim у файлі `result.csv`. Клас MapleInterface4Maplesim містить функцію, яка читає дані з цього файлу. Клас Maplesim_model успадковують класи Maplesim_model2 та Maplesim_model3, які відповідно описують спрощену модель ШСНУ та модель ВК у MapleSim.

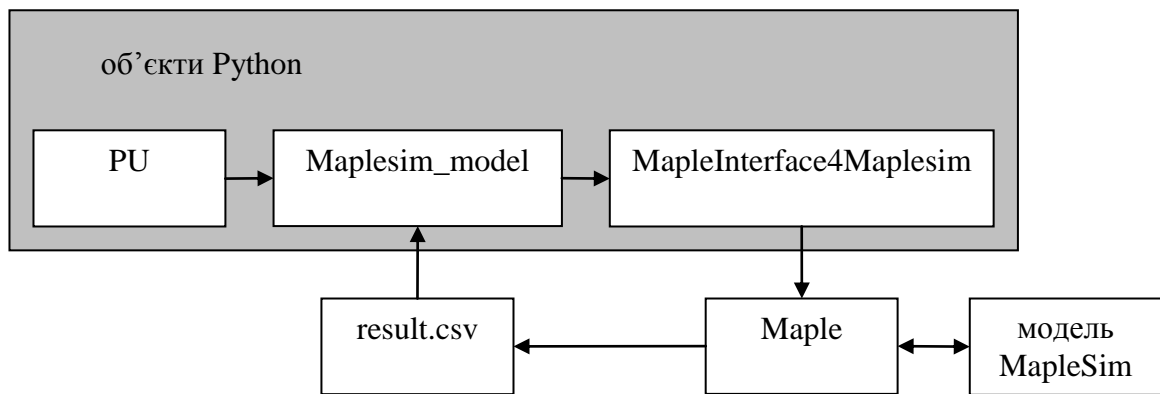
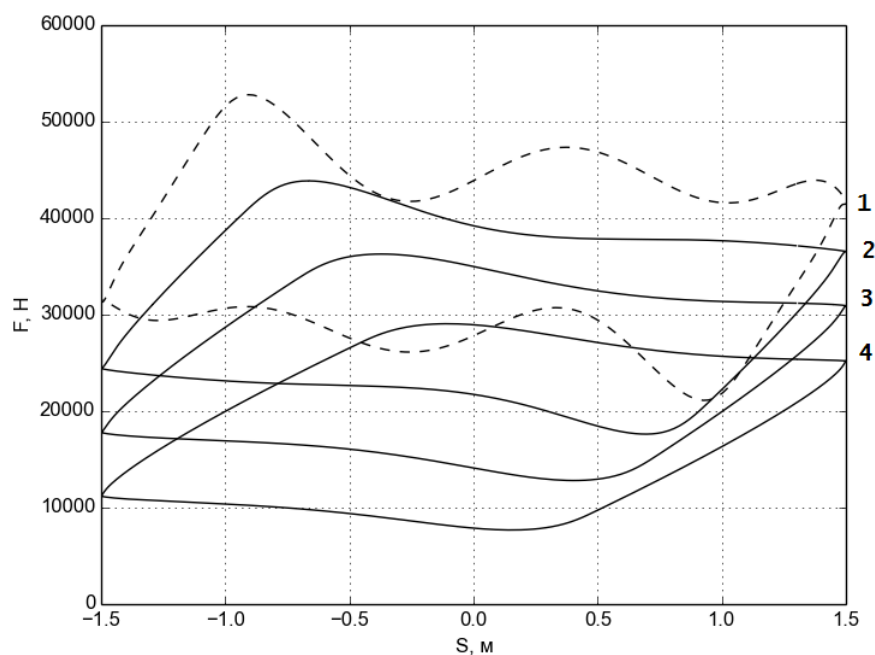


Рисунок 3.12 – Схема взаємодії компонентів САПР ШСНУ

3.2.2 Застосування і верифікація моделей

Виконано симуляцію ШСНУ за наведеними вище вихідними даними. Розглянуто колони, у яких секції складаються з ШН різного типу. Вибрано суцільні сталеві та склопластикові ШН діаметром 19 мм. На основі порівняння графіків вільних коливань моделі та реальної колони [98, 304] для секції сталевих ШН прийнято $\psi=0,2$, для склопластикових $\psi=1,6$. На рис. 3.13 показані результати – динамограми в верхній частині колони.



1 – 100% сталеві; 2 – 25% склопластикові, 75% – сталеві;
3 – 50% склопластикові, 50% – сталеві; 4 – 75% склопластикові, 25% – сталеві

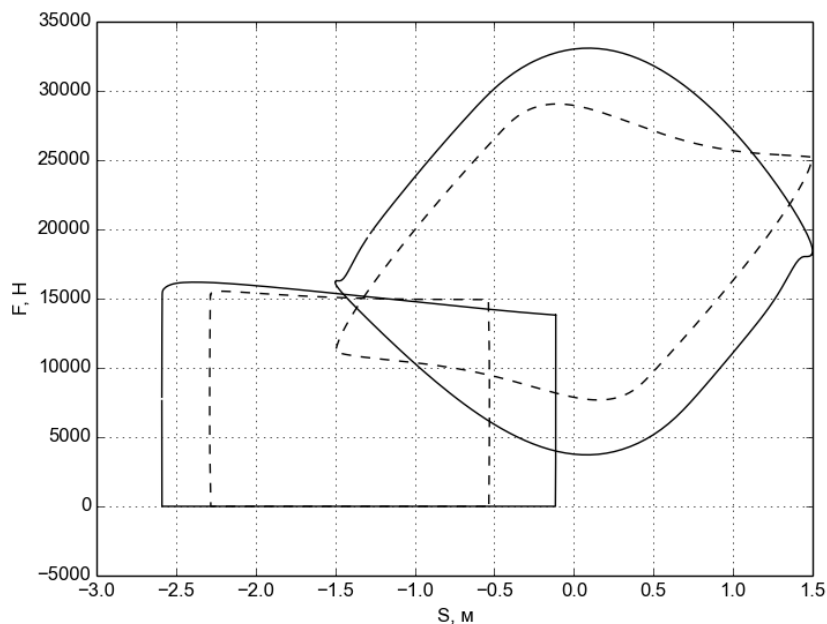
Рисунок 3.13 – Динамограми для колон з суцільними ШН 19мм

Отримані моделюванням динамограми 2, 3, 4 візуально подібні на практичні динамограми для колон з склопластиковими ШН [304] малим ходом плунжера та значним демпфуванням. Мале переміщення плунжера спричинене малою жорсткістю склопластикової частини колони та її значною деформацією. Це свідчить про необхідність оптимізації конструювання таких колон та режимів роботи ВК. Наприклад можна збільшувати частоту подвійних ходів в напрямку частоти власних коливань колони [23]. Це призводить до зростання довжини ходу плунжера (рис. 3.14). Водночас зростає амплітуда зусиль і зменшується мінімальне зусилля циклу. Тому процес оптимізації частоти повинен містити обмеження для будь-якої частини колони:

$$\sigma_{np} < [\sigma], F_{\min} > 0,$$

де σ_{np} – приведені напруження, $[\sigma]$ – допустиме приведені напруження,

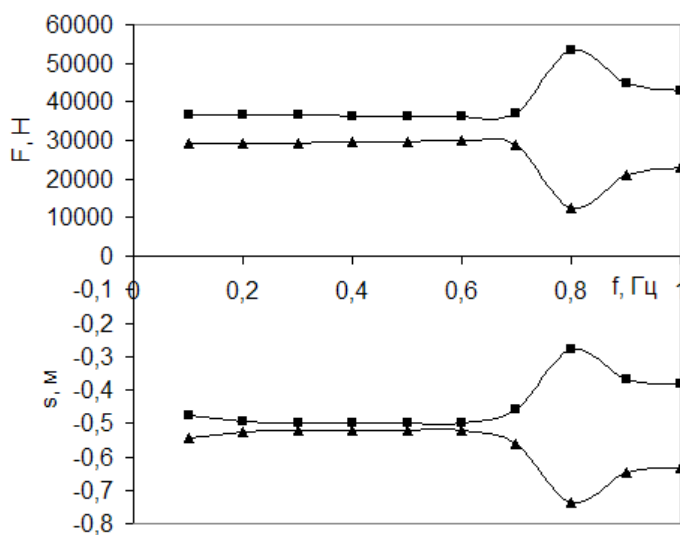
F_{\min} – мінімальне навантаження на ШН.



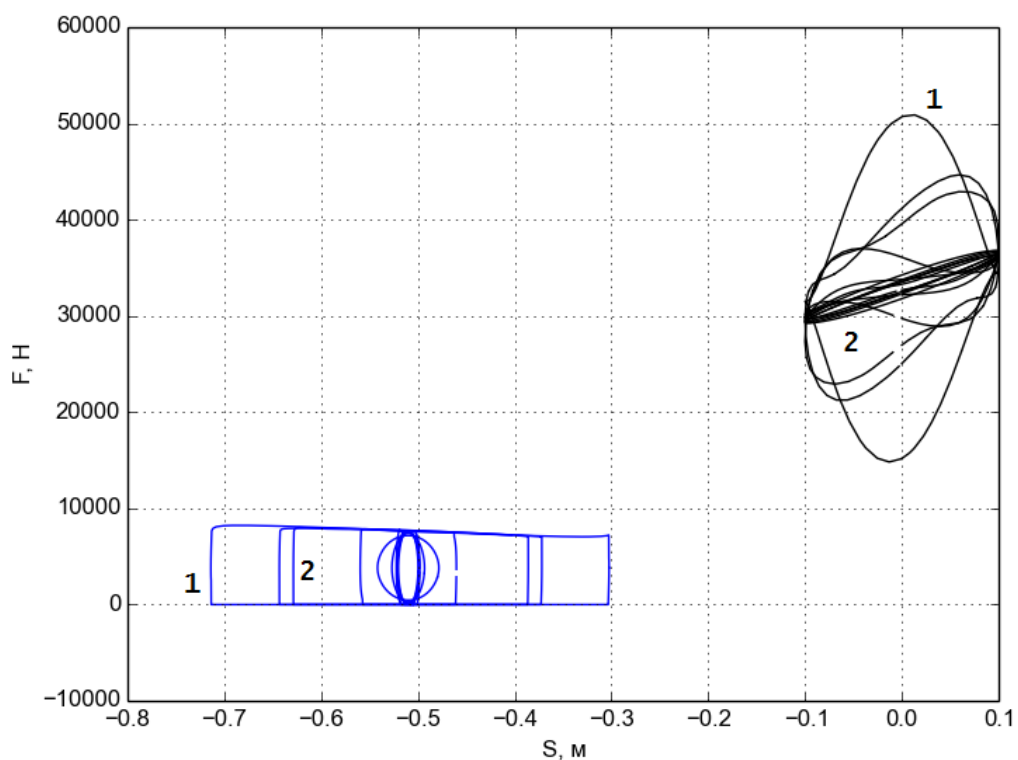
зліва внизу – плунжерні динамограми; справа вверху – гирлові

Рисунок 3.14 – Динамограми для колони зі склопластиковими ШН №4 для частоти подвійних ходів 0,108 Гц (- -) та 0,217 Гц (—)

Експлуатація з високими частотами вимагає зменшення діаметра плунжера насоса та довжини ходу штока. Розглянемо модель ШСНУ з колоною сталевих ШН 19 мм довжиною 1500 м та діаметром плунжера насоса 27 мм. Довжина ходу штока 0,2 м. Результати моделювання (рис. 3.15) показують суттєве збільшення амплітуд в околі 0,8 Гц.



а



б

а) – залежність сили в точці підвісу F та переміщення плунжера s від частоти подвійних ходів f (мінімальні (▲) та максимальні (■) значення);

б) – плунжерні (зліва внизу) та гирлові (справа вверху) динамограми

Рисунок 3.15 – Симуляція колони зі сталевими ШН для діапазону частот

подвійних ходів від 0,1 до 1 Гц

Цифрою 1 на рис. 3.15 позначено динамограми для частоти 0,8 Гц, цифрою 2 – динамограми для інших частот. Експлуатація частотою 0,8 Гц дозволяє підвищити довжину ходу плунжера майже до 0,5 м. Водночас приведені напруження в ШН не перевищує допустимого. Для класичних балансирних верстатів не досяжні така мала довжина ходу та висока частота. Але відповідні приводи можна створити на основі лінійних приводів з рейковою передачею [305], які можуть бути дуже компактними. Такі режими прийнятні тільки для колон зі сталевими ШН. На практиці експлуатація ШСНУ з резонансною частотою може бути небезпечною. Неконтрольоване збільшення амплітуди напружень може зруйнувати колону. Тому такі режими вимагають застосування адаптивної системи керування, яка слідкує за навантаженнями і, у разі досягнення ними граничних значень, зменшує частоту обертання двигуна приводу. Нижче показано результати симуляції моделі такої системи (рис. 3.16). Частота ходів полірованого штока зі значенням 0,8 Гц тривалістю 13 півперіодів змінюється на частоту 0,7 Гц тривалістю 5 півперіодів у разі досягнення граничного значення приведенного напруження на полірованому штоці.

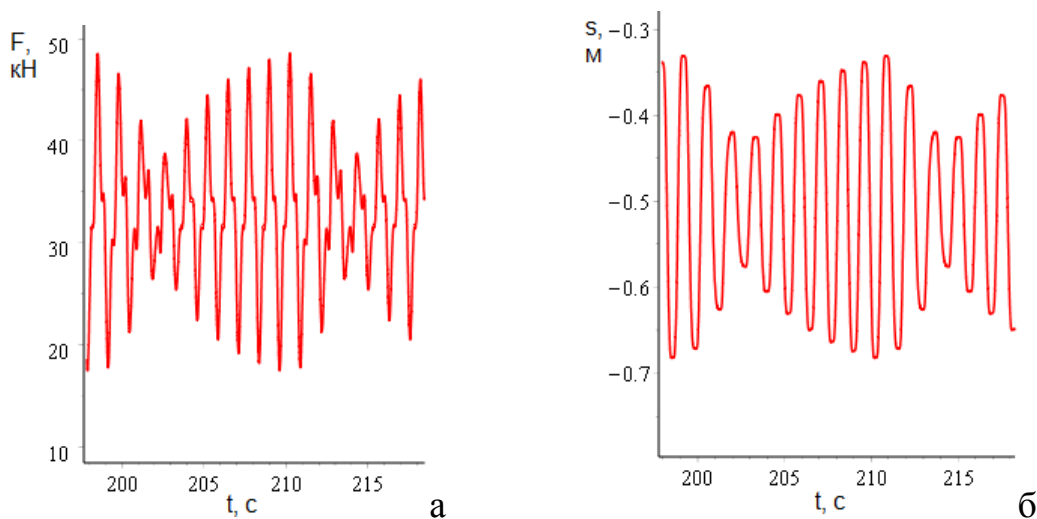
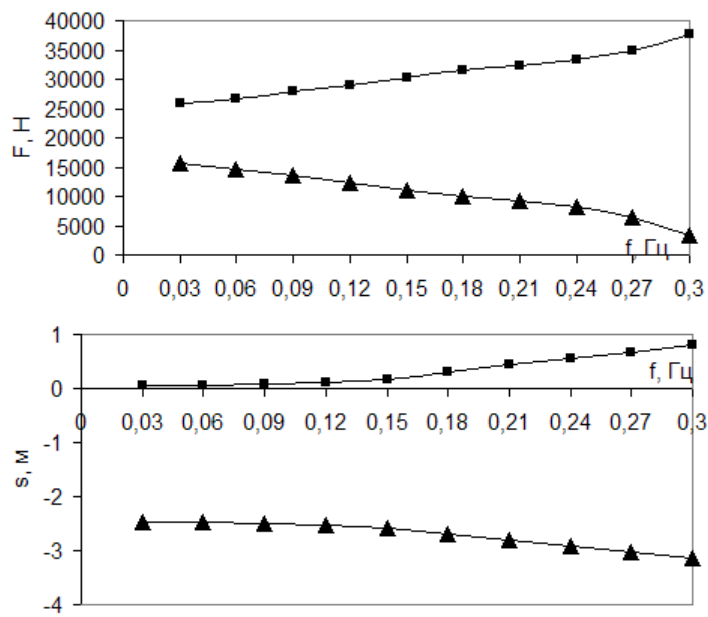
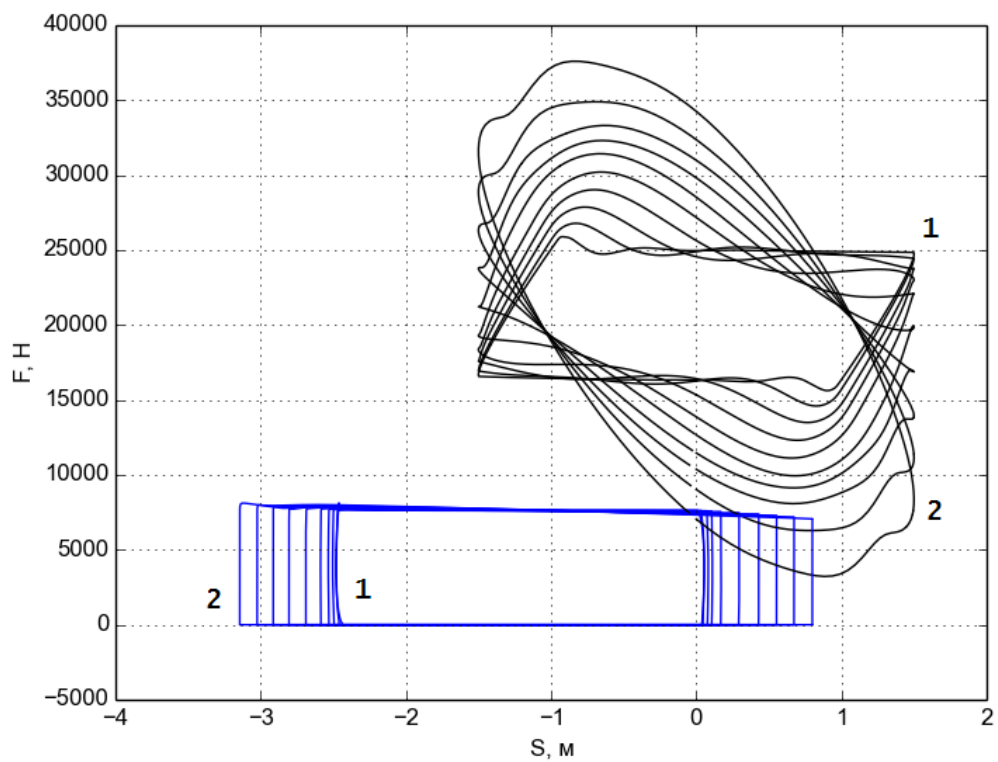


Рисунок 3.16 – Залежність сили в точці підвісу F (а) та переміщення плунжера s (б) від часу t в ШСНУ, яка обладнана системою адаптивного керування частотою

Розглянемо модель ШСНУ з колоною склопластикових (50%) та сталевих (50%) ШН 19 мм довжиною 1500 м та діаметром плунжера насоса 27 мм. Довжина ходу штока 3 м. Результати моделювання (рис. 3.17) показують можливість появи зусиль стиску вверху колони для частот більших 0,3 Гц.



а



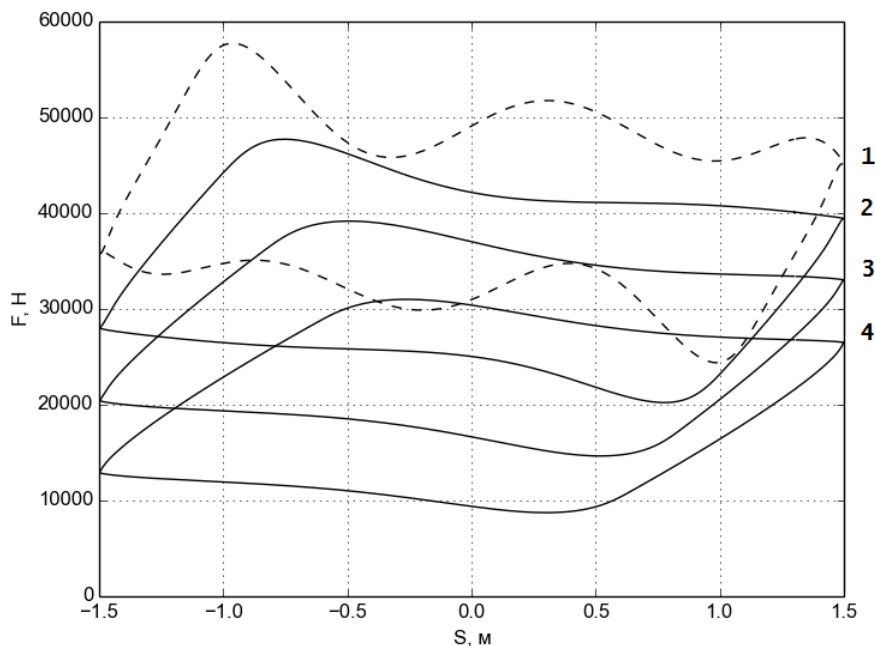
б

- а) – залежність сили в точці підвісу F та переміщення плунжера s від частоти подвійних ходів f (мінімальні (\blacktriangle) та максимальні (\blacksquare) значення);
 б) – плунжерні (зліва внизу) та гирлові (справа вверху) динамограми

Рисунок 3.17 – Симуляція колони з склопластиковими (50%) ШН для діапазону частот подвійних ходів від 0,03 Гц до 0,3 Гц

Такі зусилля недопустимі для склопластикових ШН, тому під час вибору частот слід обмежитись діапазоном 0...0,3 Гц. Високі частоти (до 0,3 Гц) вимагають застосування спеціального приводу, наприклад лінійного. Цифрою 1 на рис. позначено динамограми для частоти 0,03 Гц, цифрою 2 – динамограми для частоти 0,3 Гц.

Розглянемо колони з порожнистими ШН, наповненими рідиною. Результати моделювання показані на рис. 3.18. Для зменшення ваги порожнисті ШН можуть бути наповнені матеріалом з низькою густиною. Але динамограми для колон з порожнистими ШН з отвором 8,5 мм, які наповнені повітрям, майже не відрізняються від динамограм на рис. 3.18, що пояснюється дуже незначним зменшенням ваги і маси ШН. Більш доцільним є наповнення порожнистих ШН спеціальною рідиною, витік якої легко діагностувати на гирлі, наприклад за допомогою хімічних індикаторів. Такий витік свідчить про пошкодження тіла ШН і є сигналом для зупинки ШСНУ.



1 – 100% сталеві; 2 – 25% склопластикові, 75% сталеві; 3 – 50% склопластикові, 50% сталеві; 4 – 75% склопластикові, 25% сталеві

Рисунок 3.18 – Динамограми для колон з порожнистими ШН 22/8,5 мм, наповненими рідиною

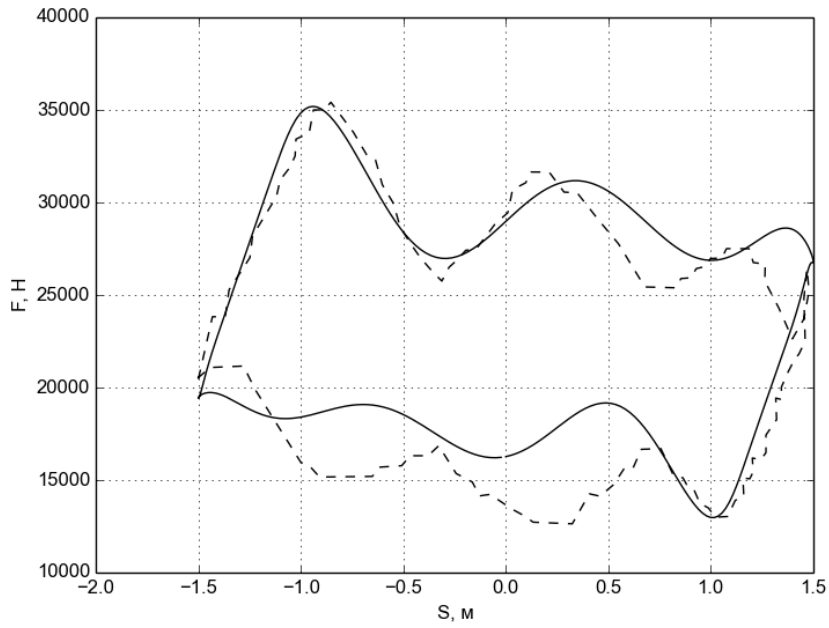
Перевірку адекватності моделі виконано шляхом порівняння її динамограм з практичними динамограмами [24]. Були вибрані динамограми, отримані на ШСНУ нафтовидобувних компаній "Долинанaftогаз" та "Чорноморнаftогаз" (у 2014 році) і які характерні для нормальної роботи ШСНУ (табл. 3.1).

Таблиця 3.1 – Параметри ШСНУ, вибраних для порівняння з моделлю

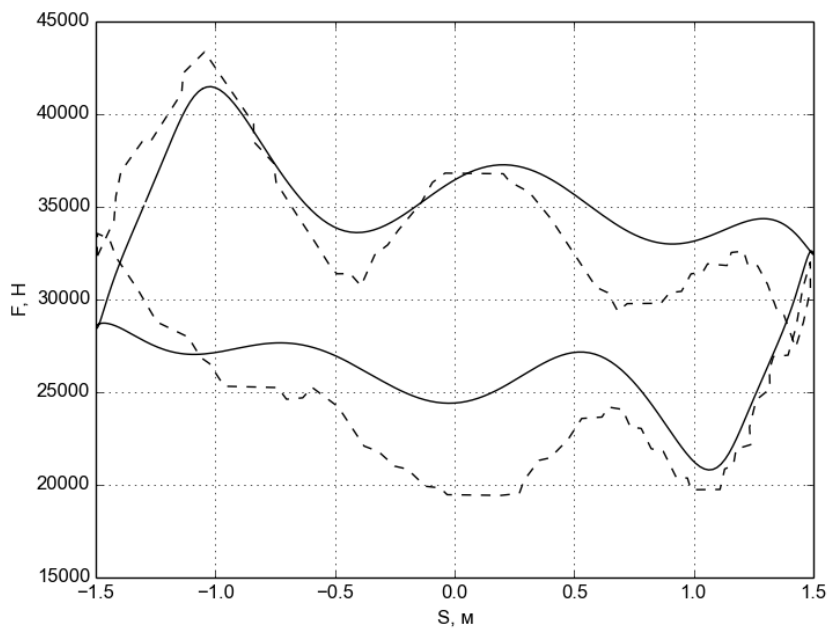
Номер свердловини	Глибина опускання насоса, м	Відсотки p ШН діаметром d (мм) в колоні, $d(p\%)$	Діаметр плунжера насоса, мм	Частота подвійних ходів, 1/хв	Довжина ходу, м	Глибина динамічного рівня, м	Густина продукції, кг/м ³
1	1359	16 (100%)	44	6,6	3	700	1000
2	1626	19 (50%), 16 (50%)	38	6,4	3	735	1000
3	1563	22 (100%)	50	6,5	3	1500	880
4	1279	22 (50%), 19 (50%)	57	6,6	3	1179	1000
5	1878	19 (100%)	32	6,6	3	1878	900
6*	1510	22 (46%), 19 (54%)	43	6,4	2,1	1410	860
7**	178	19 (50%), 16 (50%)	32	6,7	1,2	32	920

Примітки: * – за даними [98]; ** – свердловина "Чорноморнаftогаз"

Загалом відповідність теоретичних і практичних динамограм прийнятна (рис. 3.19) у порівнянні з найпростішими теоретичними діаграмами у вигляді паралелограмів [98]. Невідповідність форм верхньої та нижньої кривих теоретичних і практичних динамограм може бути пов'язана з невідповідністю сил демпфування в моделях і реальних колонах. Але точні значення коефіцієнтів демпфування для реальної колони отримати важко. Спотворення ділянки практичної діаграми №7 під час ходу вверх спричинене не видовженням колони ШН, яке в даному випадку не може бути таким великим. Його можна пояснити пропусками в напірному клапані або НКТ, як про це свідчать типові динамограми [306].

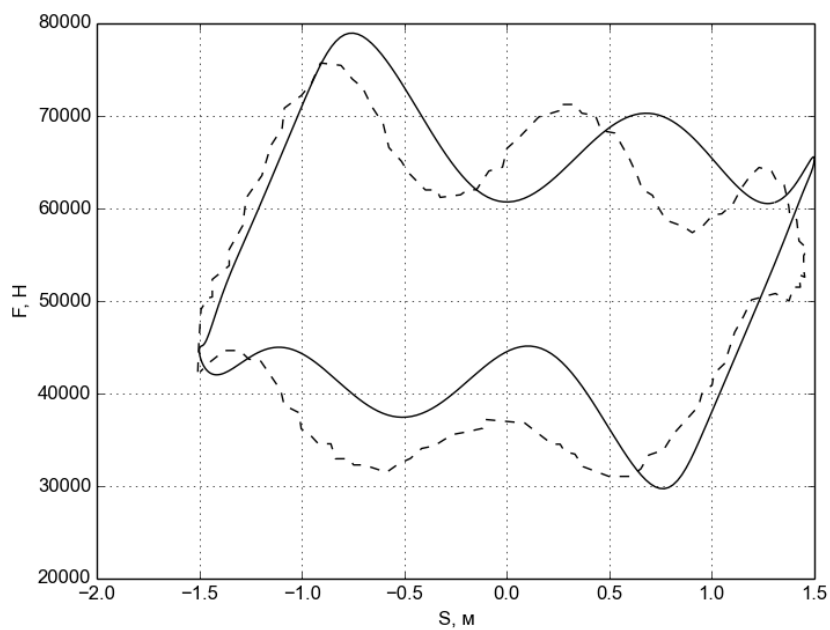


Свердловина 1

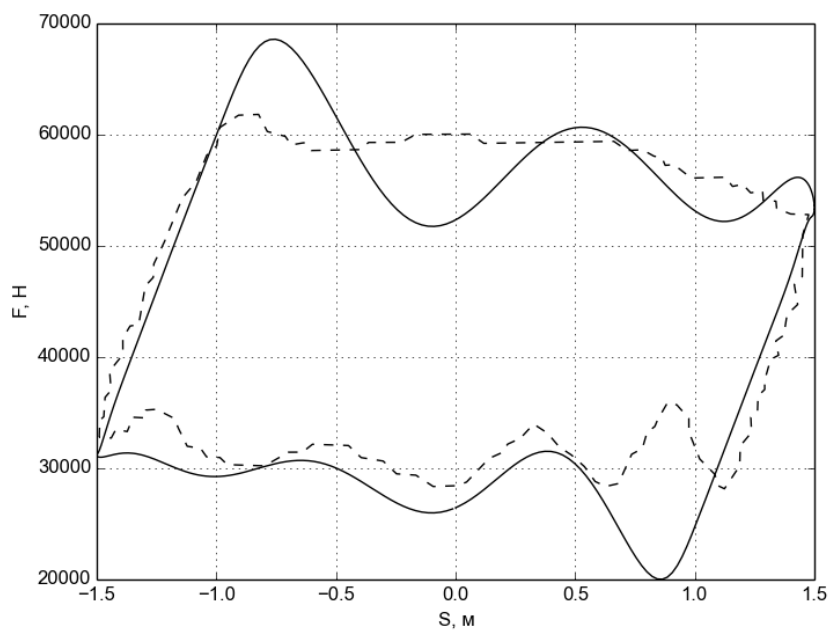


Свердловина 2

Рисунок 3.19, аркуш 1 – Порівняння результатів симуляції (—)
та практичних динамограм (- -)

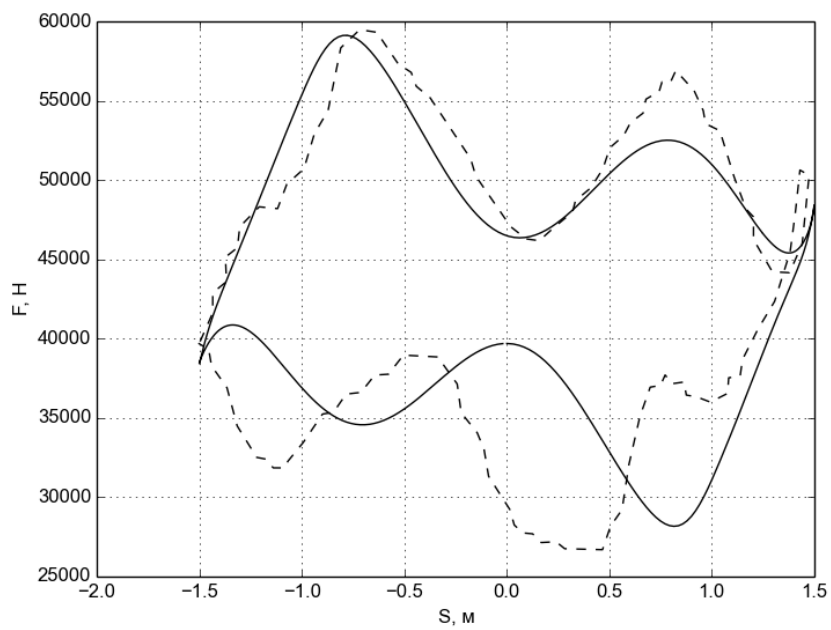


Свердловина 3

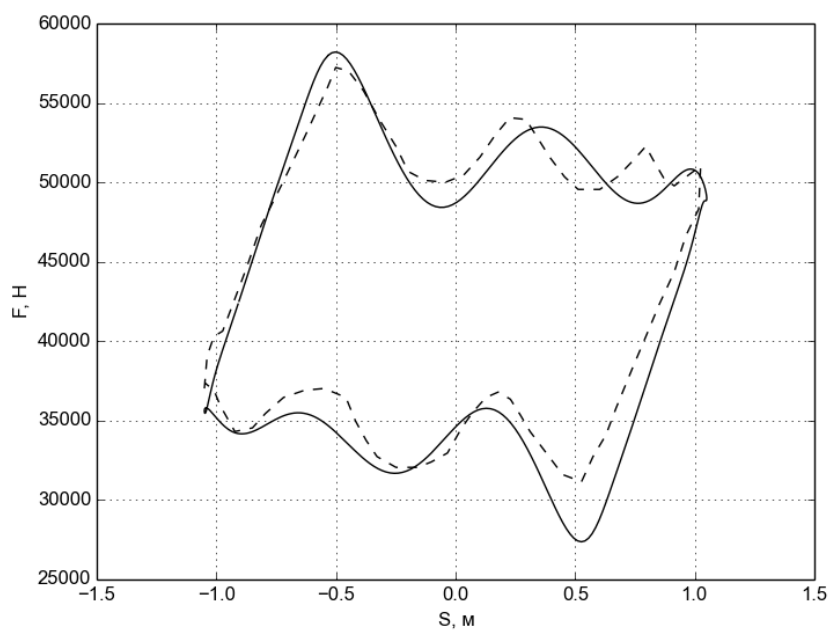


Свердловина 4

Рисунок 3.19, аркуш 2

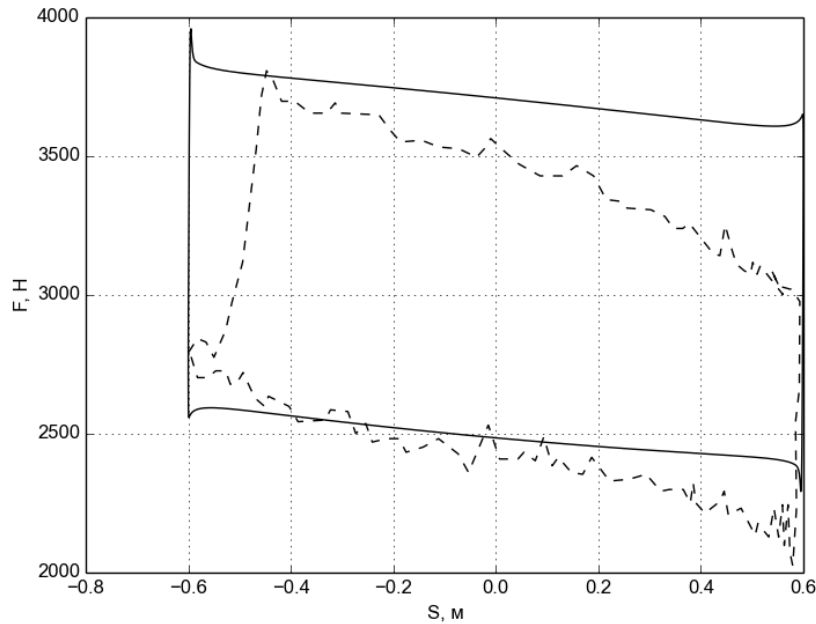


Свердловина 5



Свердловина 6

Рисунок 3.19, аркуш 3



Свердловина 7

Рисунок 3.19, аркуш 4

3.2.3 Обчислення циклічних навантажень, що діють на штанги і НКТ в умовах парафіноутворення

З метою визначення циклічних навантажень, що діють на ШН та НКТ в умовах СПУ в верхніх частинах колон, виконано симуляцію роботи ШСНУ за допомогою описаної вище моделі мовою Modelica з пружною моделлю НКТ. Колона ШН довжиною 1500 м містить ШН діаметром 19 мм, плунжер насоса має діаметр 38 мм, довжина ходу точки підвіски – 3 м, частота ходів – $6,5 \text{ хв}^{-1}$. Під час нормальної роботи ШСНУ і відсутності СПУ, циклічні навантаження розтягу на колону ШН рівні 20-50 кН, на колону НКТ – 260-290 кН (рис. Д.1). Це відповідає циклічним напруженням розтягу σ_p в тілі ШН 53-132 МПа та в тілі НКТ 110-123 МПа. Колона ШН сприймає максимальне навантаження під час початку ходу плунжера верх і мінімальне – під час початку ходу вниз. Колона НКТ, навпаки, сприймає мінімальне навантаження під час початку ходу верх і максимальне – під час початку ходу вниз. Під час ходу плунжера вгору на НКТ не діє осьове навантаження розтягу від ваги рідини, яка сприймається плунжером, але водночас

на внутрішню бічну стінку НКТ передається гідростатичний тиск стовпа рідини [82]. Модель підтверджує ці факти. Різке закриття клапана під час ходу вверх спричиняє зростання тиску внизу НКТ і коливання колони. Цикл зміни тиску P в НКТ рівний 13-14 МПа.

У випадку СПУ, які утворюються в верхній частині колони, навантаження на ШН і НКТ зростає. Для моделювання СПУ зменшували внутрішній діаметр НКТ в верхніх 500 м колони до 20 мм і 10 мм. У разі зменшення отвору НКТ до 20 мм осьові навантаження на колони ШН та НКТ майже не змінюються, але зростає значення тиску P в НКТ (під час ходу вверх) до 18 МПа (рис. Д.1) та збільшується демпфування. В цей момент на колону НКТ діє мінімальне осьове навантаження розтягу. Цикл зміни тиску в НКТ рівний 13-18 МПа. У разі зменшення отвору НКТ до 10 мм осьові навантаження на колони значно зростають. Цикл навантажень на ШН рівний 20-85 кН, а на НКТ – 220-290 кН. Це відповідає циклічним напруженням σ_p 53-224 МПа в штанзі та 93-123 МПа в НКТ. Таким чином значення σ_p штанзі перевищують допустимі циклічні напруження за діаграмою Гудмена (53-170 МПа). Значення σ_p НКТ знаходяться в межах допустимих. Але цикл зміни тиску P в НКТ значно зростає і рівний 13-50 МПа. Якщо у верхній частині НКТ напруження від її ваги рівні 120 МПа, а гідростатичний тиск близько 0 МПа (внизу 13 МПа), то екстремальні навантаження на НКТ будуть наступними (табл. 3.2).

Таблиця 3.2 – Екстремальні навантаження на НКТ, отримані за результатами симуляції ШСНУ (МПа)

Частина колони	Навантаження	Крок 1 (хід вверх)	Крок 2 (хід вниз)
верхня	σ_p	93	123
	P	37	0
нижня	σ_p	-27	3
	P	50	13

Вказані в таблиці кроки навантаження можна використовувати для скінченно-елементної симуляції елементів НКТ (РЗ, склопластикових бандажів) та обчислення коефіцієнта запасу втомної міцності.

3.3 Автоматизована симуляція гідродинаміки клапана свердловинного штангового насоса для різної висоти підйому кульки

Для Abaqus/CFD, що реалізує MCE для обчислювальної гідродинаміки, розроблено параметричну СЕМ кульового клапана [110]. Геометричні параметри тривимірної моделі клапана V11-125 показані на рис. 3.20. Тут параметр h визначає висоту підйому кульки і є змінним.

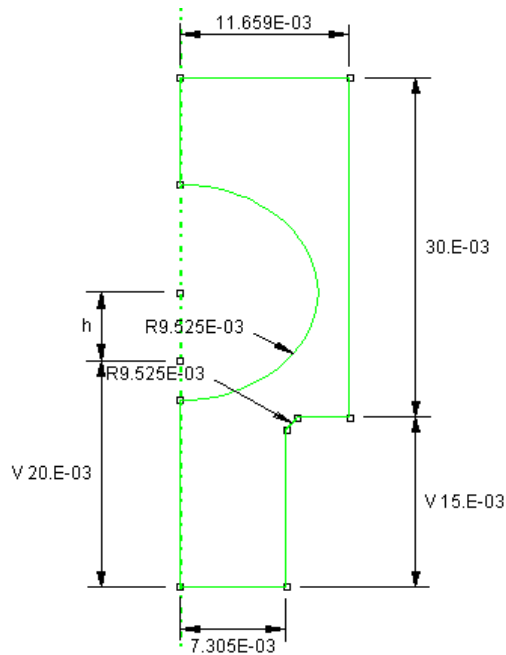


Рисунок 3.20 – Геометричні параметри моделі клапана

Властивості рідини: густина 1000 кг/м^3 , динамічна в'язкість $0,001 \text{ Па}\cdot\text{с}$. Вибрано RNG k- ϵ модель турбулентності. Початкові умови: швидкість у будь-якій точці рівна 0, задано також початкові значення кінетичної енергії турбулентності та дисипації турбулентності. Граничні умови: тиск на виході (верхня площина) клапана 0 Па , швидкість рідини на вході (нижня площина) 1 м/с . Під час моделювання зворотних перетоків вхід і вихід міняються місцями. Коефіцієнт витрати розраховували за формулою $\mu = V / (2\Delta P / \rho)^{0,5}$, де V – середня швидкість потоку в заданому перетині, ΔP – перепад тисків на вході та виході (рис. 3.21а,б), ρ – густина. Додатково розроблено макрос мовою Python, який використовує Abaqus/CAE API для автоматизації перебудови параметричної моделі та отримання результатів (додаток Е). Нижче наведено його алгоритм для автоматизованого отримання залежності $\Delta P(h)$.

```
FOR EACH h IN (0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007):
    встановити значення параметра 'h' та перебудувати геометрію
    генерувати сітку елементів
    виконати задачу і чекати завершення
    відкрити результати, отримати значення і зберегти їх у базі даних
```

Тут використовуються розроблені автором функції `set_values` та `readODB_set2` (лістинг Е.3). Перша присвоює значення геометричним параметрам (`par`) деталі (`part`). Для цього ця функція використовує об'єкт ескізу деталі з потрібним геометричним параметром. Друга функція повертає список результатів заданої величини (`var`) із заданої множині вузлів (`set`) з останнього фрейму кроку (`step`). Для цього вона використовує об'єкт бази даних результатів `odb`. Детальнішу інформацію про об'єкти та функції інтерфейсу програмування Abaqus/CAE можна знайти в документації програми.

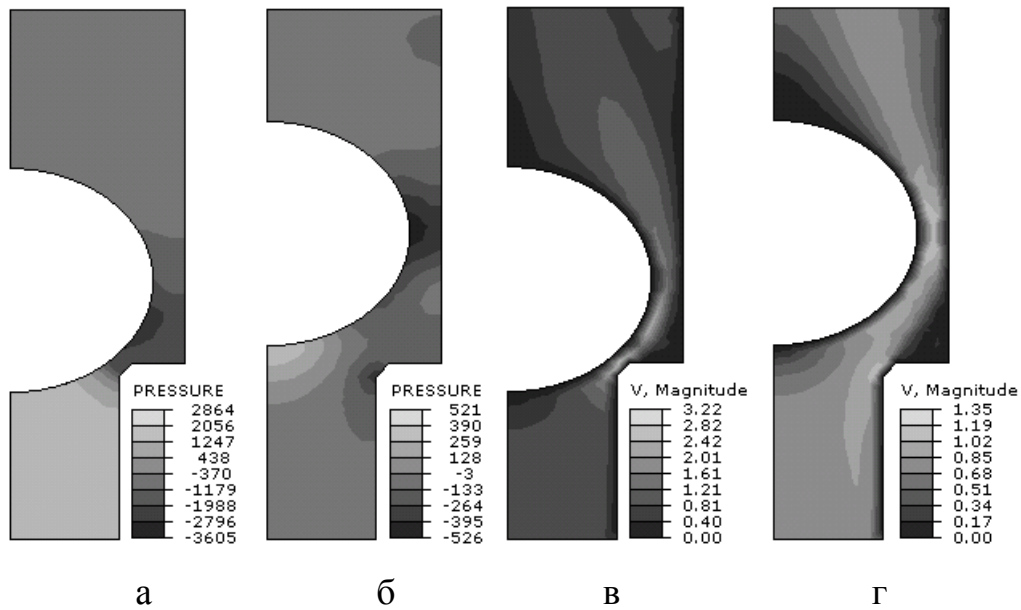
Зауважимо, що симуляція (для одного значення h) на комп'ютері з процесором Core 2 Duo триває біля 9 годин і не завжди може завершитись успішно (що залежить від якості сітки елементів). У разі неуспішного завершення обчислень зупиняється цикл і наступні задачі не виконуються. Можна підвищити надійність цієї програми розділивши її на два процеси таким чином, що тіло циклу алгоритму знаходиться в модулі `script.py`, а модуль `server.py` (лістинг Е.2) замість тіла циклу містить команди, які створюють новий процес. Функція `subprocess.Popen` створює новий процес `abq6123.exe cae noGUI = script.py`, в якому сценарій Abaqus `script.py` виконує перебудову геометрії моделі, генерує сітку елементів, виконує задачу та читає результати. Далі `server.py` очікує завершення цього процесу і результати записує у файл CSV. Функції модуля `pickleIPC` (лістинг Е.1) `writeTempFile` та `readTempFile` призначені для міжпроцесової взаємодії `server.py` та `script.py` шляхом запису та читання даних (у вигляді об'єктів Python) з тимчасового файлу. Ці функції використовують стандартний модуль Python `pickle` для збереження об'єктів Python у бінарних файлах. На початку модуля `script.py` (лістинг Е.3) необхідно додати команди для відкриття моделі та отримання даних з процесу `server.py`. Далі ідуть команди для перебудови геометрії моделі, генерації сітки елементів, виконання задачі та

читання результатів, так як це робиться в тілі циклу вказаного вище алгоритму. Кінець модуля повинен містити команди для запису результатів в тимчасовий файл з метою передачі їх процесу `server.py`. Додатково, шляхом копіювання, можна зберегти файл бази даних результатів `.odb` для його аналізу в ручному режимі.

Останні версії Abaqus/CAE дозволяють виконувати паралельні обчислення (якщо це передбачено ліцензією) на основі інтерфейсу MPI, а також на графічних процесорах. Проте, якщо відповідної ліцензії немає, то за допомогою Python можна просто організувати паралельну розподілену обчислювальну систему з паралелізмом задач. Для цього необхідно N об'єднаних в мережу комп'ютерів з установленою Abaqus/CAE поєднати програмою-сервером, яка виконується на одному з них. В ній, використовуючи стандартний модуль Python `SocketServer`, потрібно створити мережевий багатопотоковий сервер, а на інших комп'ютерах запуснути програму-клієнт. Сервер розділяє задачі між клієнтами та відсилає їм дані (у даному випадку значення h). Кожен клієнт запускає процес `abq6123.exe cae noGUI=script.py`, як це було показано вище, чекає результатів та відсилає їх програмі-серверу. Така клієнт-серверна обчислювальна система дозволить зменшити час розрахунків в приблизно N раз.

Результати моделювання для різної висоти підйому кульки h показані на рис. 3.21. Помітне збільшення тиску в нижній частині та швидкості в щілині для меншого значення h . Застосування описаних вище програм мовою Python дозволило отримати залежності перепаду тисків ΔP та максимальної швидкості V_{\max} від висоти підйому кульки h (рис. 3.22).

Особливістю розробленої моделі є можливість автоматизованого звернення до неї за результатами моделювання з зовнішніх програм. Зокрема, ця модель може використовуватись як компонент динамічної моделі клапана, розробленою мовою моделювання систем Modelica [110, 113]. Наприклад, в головному модулі програми задаються значення параметрів моделі та запускається модуль симуляції СЕМ в Abaqus/CAE. Звідти результати (залежності $\Delta P(h)$ або $\mu(h)$) передаються в модуль розрахунку динамічної моделі мовою Modelica. Розроблена модель може також використовуватись для оптимізації конструкції клапана.



а, б) – тиск (Па); в, г) – швидкість (м/с); а, в) – $h=2$ мм; б, г) – $h=6$ мм

Рисунок 3.21 – Результати симуляції

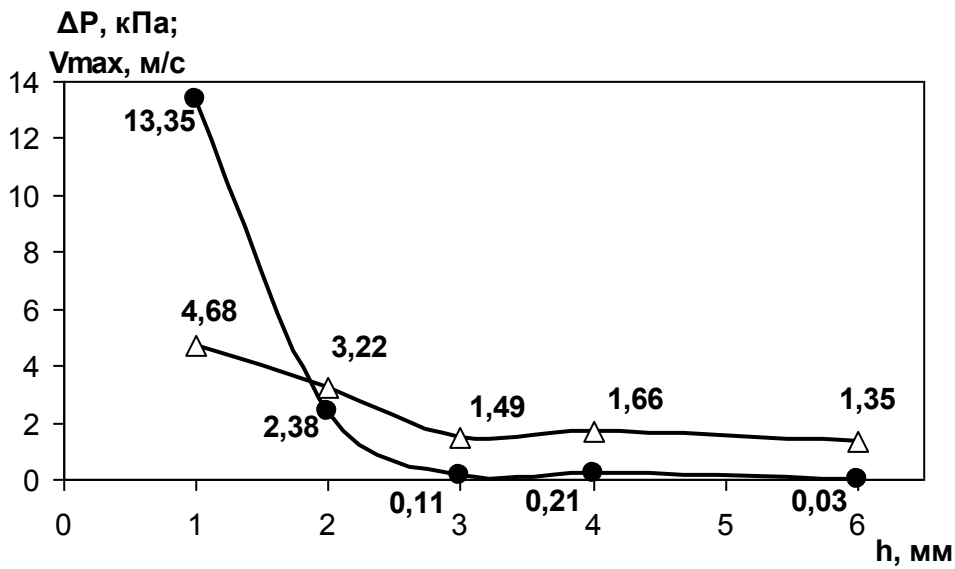


Рисунок 3.22 – Залежності перепаду тисків ΔP (●) та максимальної швидкості V_{\max} (Δ) від висоти підйому кульки h

3.4 Компонентно-орієнтоване акаузальне моделювання динамічних систем мовою Python на прикладі моделі колони насосних штанг

3.4.1 Опис моделі мовою Modelica

Як це було показано вище, компонентно-орієнтоване акаузальне гібридне моделювання ШСНУ можна реалізувати спеціалізованою мовою моделювання Modelica. Але спеціалізований характер, складність впровадження та вивчення таких мов дещо обмежує їхній розвиток та широке використання розробниками, які знають лише мови загального призначення. Нижче, на прикладах симуляції руху колони ШН, пропонуються принципи розробки подібної до Modelica, простої для розуміння та модифікації, системи компонентно-орієнтованого акаузального моделювання на основі мови програмування загального призначення Python [25, 28, 307].

Розглянемо колону довжиною 1510 м, яка разом з її практичною динамограмою описані в праці [98]. Верхня секція колони складається з 695 м ШН діаметром 22 мм, а нижня секція складається з 815 м ШН діаметром 19 мм. Ця колона матиме загальну масу 3961 кг, загальну вагу в рідині 34687 Н, коефіцієнт пружності 44650 Н/м, коефіцієнт демпфування 2121 Н·с/м. Вага рідини над насосом діаметром 43 мм становитиме 18499 Н.

Спочатку опишемо моделювання вільних коливань колони ШН за допомогою мови Modelica. Розробимо модель простого механічного поступального осцилятора, який складається з компонентів "маса", "пружина-демпфер" та "зафіксований фланець" (рис. 3.23). Компонент "пружина-демпфер" призначений для моделювання пружно-демпферних властивостей колони, компонент "маса" моделює її інерційні властивості, компонент "зафіксований фланець" моделює нерухому точку підвіски колони. В додатку (лістинг Є.1) наведено код модуля мовою Modelica, який описує цю модель. З метою спрощення моделі код наведених класів дещо відрізняються від відповідних класів стандартної бібліотеки Modelica.

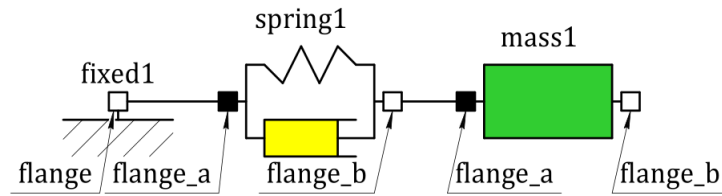


Рисунок 3.23 – Блок-схема моделі колони ШН

Клас мови Modelica описує множину однотипних об'єктів (компонентів). Так клас `Flange` описує порт (фланець) для з'єднання компонентів. Його змінна дійсного типу `s` відповідає переміщенню фланця. Її значення повинно бути рівним значенню змінних `s` інших фланців, з'єднаних з цим фланцем. Змінна дійсного типу `f` відповідає силі на фланці. Вона позначена ключовим словом `flow`, що означає рівність нулю суми усіх сил в точці з'єднання. Клас `Fixed` описує зафіксований фланець, наприклад такий як `fixed1` на рис. 3.23. Він містить змінну дійсного типу `s0`, яка відповідає переміщенню фланця, та об'єкт `flange` класу `Flange`, призначений для з'єднання цього компонента з іншими. Змінна `s0` позначена ключовим словом `parameter`, що означає можливість зміни її значення тільки до початку моделювання. Після ключового слова `equation` наведено рівняння, яке описує поведінку цього компонента – переміщення фланця `flange` повинно бути рівним значенню `s0`. Клас `Trans1` описує абстрактний компонент, який володіє двома фланцями типу `Flange` – `flange_a` і `flange_b`. Клас `Mass` описує зосереджену в точці масу, яка переміщується поступально. Прикладом такого компоненту є `mass1` на рис. 3.23. Команда `extends Trans1` означає успадкування членів класу `Trans1` таким чином, що вони стають членами класу `Mass`. Тобто компоненти `Mass` будуть теж володіти двома фланцями `flange_a` і `flange_b`. Крім цього, клас володіє параметром `m` (маса) та змінними `s` (переміщення), `v` (швидкість), `a` (прискорення). Запис `start=0` означає початкову умову за замовчуванням. Після ключового слова `equation` наведено систему диференціально-алгебраїчних рівнянь, яка описує поведінку цього компонента. Слово `der` означає похідну по часу `t`: $v = ds / dt$, $a = dv / dt$.

Клас `SpringDamper` описує поступальні пружину і демпфер, які з'єднані паралельно. Прикладом такого компоненту є `spring1` на рис. 3.23. Цей клас теж успадковує клас `Trans1`. Додатково клас володіє параметрами `c` (коефіцієнт жорсткості), `d` (коефіцієнт демпфування) та змінними `s_rel` (відносне переміщення фланців), `v_rel` (відносна швидкість фланців), `f` (сила на фланці `flange_b`). Після слова `equation` наведено систему диференціально-алгебраїчних рівнянь цього компонента. Клас `Oscillator` описує увесь осцилятор, показаний на рис. 3.23. Він містить три компоненти `mass1`, `spring1`, `fixed1`, які описуються класами `Mass`, `SpringDamper`, `Fixed` відповідно. В дужках вказані значення параметрів та початкові умови цих компонентів. Після слова `equation` наведено систему диференціально-алгебраїчних рівнянь, потрібних для з'єднання компонентів `mass1`, `spring1`, `fixed1` в одну систему. Так запис `connect(fixed1.flange, spring1.flange_a)` з'єднує фланці компонентів `fixed1` і `spring1` та створює систему рівнянь:

```
fixed1.flange.s = spring1.flange_a.s
fixed1.flange.f = -spring1.flange_a.f
```

Код моделі можна підготувати за допомогою будь-якого текстового редактора або модуля Modelica Development Tooling (MDT) [163] середовища розробки Eclipse. Для симуляції моделі знадобиться середовище OpenModelica [163]. Щоб розпочати обчислення в консолі MDT введіть: `simulate(Oscillator, stopTime=10)`. Після закінчення обчислень для побудови графічної залежності переміщення компоненту `mass1` від часу в консолі введіть: `plot(mass1.s)`.

Типовими етапами трансляції та симуляції моделі Modelica є [166]: трансляція (отримання "плоскої" множини рівнянь, констант, змінних та визначення функцій з Modelica-коду), аналіз (сортування рівнянь, перетворення матриці коефіцієнтів у трикутну форму), оптимізація (усунення деяких рівнянь, перетворення рівнянь у оператори присвоєння), генерація коду (отримання C-коду), компіляція (отримання виконуваного файлу) та симуляція. Однак існують альтернативні способи виконання Modelica [166].

3.4.2 Опис принципів моделювання в Python

Принципи компонентно-орієнтованого моделювання в Python описано нижче [25]. Також показано приклад реалізації аналогічної моделі колони ШН.

1. Компоненти описуються класами Python, які структурно схожі на класи Modelica і мають такі атрибути: постійні параметри та символи SymPy (аналоги параметрів і змінних у Modelica), символьні рівняння SymPy (різницеві або диференціально-алгебраїчні (DAE)), піни для під'єднання компонентів до системи (аналоги коннекторів у Modelica). Динамічна система, що складається з компонентів, також описується класом Python, атрибутами якого є списки компонентів та рівнянь (разом з додатковими рівняннями для з'єднання компонентів).

2. Якщо динамічна поведінка компонентів описується різницевими рівняннями, тоді користувач повинен описати ці рівняння в класі, замінивши похідні вибраною різницевою схемою (наприклад, методом Ейлера).

3. Початкові умови підставляються у ці різницеві рівняння, а невідомі знаходять шляхом розв'язування системи нелінійних рівнянь на кожному кроці або за допомогою функції, яка була створена спочатку за допомогою функції `lambdify` SymPy (яка переводить вираз SymPy в еквівалентну числову функцію), і обчислень невідомих значень на поточному кроці за відомими значеннями попереднього кроку без необхідності розв'язувати рівняння.

4. На кожному кроці оператор `if` перевіряє наявність дискретних подій, які залежать від стану чи часу. Під час обробки подій початкові умови, компоненти або рівняння можуть бути змінені.

5. Якщо динамічна поведінка компонентів описується DAE, то використовується інтерфейс Assimulo до розв'язувачів DAE, який має ефективну процедуру обробки розривів.

6. Пакет SymPy дозволяє довільну маніпуляцію рівняннями моделей та генерацією коду. Можна розв'язувати деякі алгебраїчні або диференціальні рівняння аналітично. Систему рівнянь DAE слід спростити, перетворити в звичайні диференціальні рівняння (ODE) і розв'язати за допомогою функції `SymPy dsolve`.

Мовою Python розроблено модуль `pycodyn` з аналогічними компонентами (лістинг Є.2). Додатково розроблено компонент "сила" для моделювання зовнішніх сил, що діють на колону. Поведінку компонентів описували за допомогою різницевих рівнянь. В результаті система компонентів, з'єднаних фланцями, буде описуватись системою різницевих рівнянь. Спочатку імпортуються необхідні модулі – `sympy` та стандартний математичний модуль `math`. Важливо розрізнити функції цих модулів, тому замість команди `from math import *` бажано застосовувати: `from sympy import *`; `import math`. Далі створено глобальну змінну `dt` – крок часу: `dt=0.1`. Якщо тільки потрібно отримати систему рівнянь в символьному вигляді, то ця змінна повинна бути екземпляром класу `Symbol` модуля `sympy`: `dt=Symbol('dt')`.

`Translational1D` – базовий клас механічних 1D компонентів, які переміщуються поступально. Функція конструктор `__init__` викликається під час створення об'єкта цього класу і має два параметра: назва компонента `name` та словник його атрибутів `args`. Для іменування атрибутів компонентів будемо використовувати наступні позначення. Символи на початку назви `x`, `v`, `a`, `f` означають відповідно переміщення, швидкість, прискорення і силу. Символ в кінці назви `p` означає значення в момент часу `t-dt`. Числовий індекс в кінці відповідає номеру фланця. Щоб розрізнити змінні різних компонентів в системі, кожна з них починається з назви компонента з наступним символом `"_"`. Наприклад назва `s1_x2p` означає переміщення другого фланця компонента `s1` в момент часу `t-dt`. Конструктор для кожної пари назва-значення словника `args` (окрім назв `name` і `self`) створює змінні SymPy – символьну класу `Symbol`, якщо її значення не відоме, або числову класу `Number`, якщо її значення відоме. Список `self.eq` містить рівняння компонента, а список `self.pins` – фланці компонента. Кожне рівняння створюється за допомогою SymPy класу `Eq`. Кожний фланець описується словником, ключами якого є `x`, `xp`, `f`, а значеннями – відповідні атрибути компонента (див. описи класів `Mass`, `SpringDamper`, `Force`). Функція `pinEqs` повертає список рівнянь для фланця компонента, який з'єднаний з фланцями інших компонентів. Вона має два параметра: індекс фланця компонента `pindex` (наприклад `0`) та список фланців інших компонентів `pins`.

Для механічних поступальних 1D компонентів необхідно, щоб переміщення компонентів на фланці були рівні, а сума сил на фланці була рівна нулю. Наприклад, якщо фланець 2 компонента `s1` з'єднаний з фланцем 1 компонента `m1` (рис. 3.24a), то функція `pinEqs` компонента `s1` поверне список рівнянь у форматі SymPy:

```
[s1_x2==m1_x1, s1_x2p==m1_x1p, s1_f2== -m1_f1]
```

Клас `Mass` описує зосереджену в точці масу, яка переміщується поступально. Він успадковує клас `Translational1D`. Конструктор класу `__init__` викликає конструктор базового класу `Translational1D` та передає йому параметри `name` та `locals()` – словник локальних змінних (`self`, `name`, `m`, `x`, `xp`, `v`, `vp`, `a`, `f1`, `f2`). Поведінка цього компонента описується системою рівнянь `self.eqs`. Наприклад для компонента `m1` (рис. 3.24a) :

```
[m1_m*m1_a == m1_f1+m1_f2, m1_a == (m1_v- m1_vp)/dt,
m1_v == (m1_x-m1_xp)/dt]
```

Окрім цих рівнянь для кожного фланця компонента можна сформувати список додаткових рівнянь за допомогою функції `pinEqs`, описаної вище. Першим елементом списку `self.pins` є словник `dict(x=self.x, xp=self.xp, f=self.f1)`, який означає, що переміщення на фланці `x`, `xp` будуть відповідно рівні атрибутам `self.x`, `self.xp` цього компонента, а сила на фланці `f` – рівна атрибуту `self.f1`. Аналогічно для другого елементу списку.

Клас `SpringDamper` описує поступальні 1D пружину і демпфер, які з'єднані паралельно. Він теж успадковує клас `Translational1D`. Окрім описаних вище атрибутів він володіє атрибутами: коефіцієнт жорсткості `c`, коефіцієнт демпфування `d`, відносна швидкість між фланцями `vrel`. Поведінка цього компонента описується системою рівнянь `self.eqs`. Наприклад для компонента `s1` (рис. 3.24a):

```
[s1_c*( s1_x2-s1_x1)+ s1_d*s1_vrel == s1_f2, -s1_f2 == s1_f1,
s1_vrel == (s1_x2-s1_x2p)/dt-(s1_x1-s1_x1p)/dt]
```

Цей компонент теж володіє двома фланцями і є можливість сформувати список додаткових рівнянь за допомогою функції `pinEqs`.

Клас `Force` описує 1D силу, точка прикладення якої переміщується поступально. Значення сили f може бути постійне або змінне. Він теж успадковує клас `Translational1D`. Володіє одним фланцем.

Клас `System` описує систему компонентів, з'єднаних фланцями. Конструктор класу `__init__` отримує два параметра – список компонентів `els` та список додаткових рівнянь `eqs`, які, як правило, формуються за допомогою функцій `pinEqs`. Компоненти системи зберігаються в списку `self.els` та словнику `self.elsd`. Список `self.eqs` містить усі рівняння системи і формується шляхом об'єднання рівнянь усіх компонентів з додатковими рівняннями `eqs`. Функція цього класу `solve` розв'язує стаціонарну задачу. Вона повертає розв'язок системи алгебраїчних рівнянь з умовами `ics` – словником з відомими значеннями змінних. Для розв'язування системи рівнянь вона може використовувати функцію `SymPy solve`, але її алгоритм дуже повільний. Можливо використати швидші алгоритми розв'язування рівнянь, наприклад функцію `scipy.optimize.root` з бібліотеки `SciPy`, яка підтримує багато ефективних методів розв'язування систем рівнянь. В цьому випадку виклик функції `SymPy solve(eqs)` потрібно замінити викликом функції `self.solveN(eqs)`, яка адаптує систему рівнянь для `SciPy` і розв'язує її за допомогою `scipy.optimize.root`. Функція `solveDyn` розв'язує нестаціонарну задачу. Вона отримує три параметри – словник з початковими умовами `d`, значення кінцевого часу `timeEnd` та функцію `fnBC`, яка повертає словник для оновлення граничних умов. Спочатку змінній часу `t` присвоюється початкове значення. В циклі `while` з умовою `t < timeEnd` виконуються наступні інструкції: змінним попереднього кроку (`xp`, `x1p`, `vp` тощо) присвоюються значення початкового стану `state`, значення граничних умов оновлюються, система рівнянь розв'язується шляхом виклику методу `self.solve`, розв'язки присвоюються словнику `state`, результати зберігаються, значення часу збільшується на `dt`. Після завершення циклу функція повертає результати у вигляді списків `T` і `Res`. Ці результати можна показати у вигляді графічних залежностей (рис. 3.24) за допомогою бібліотеки `matplotlib`. Проте використання методу `self.solve` може бути прийнятним лише для дуже частих

подій, які потребують оновлення рівнянь. Оскільки на кожному кроці цей метод створює та розв'язує систему нелінійних рівнянь, обчислення можуть бути дуже тривалими. У більшості випадків його слід замінити методом `solvn`, який на кожному кроці знаходить невідомі значення, передаючи знайдені на попередньому етапі значення методу `ceqsf`. На початку моделювання та після подій цей метод повинен бути створений за допомогою функції SymPy `lambdify`, яка перетворює вирази SymPy в лямбда-функції, що можуть бути використані для обчислення числових значень дуже швидко. Це робиться методом `createCurEqs`.

Обробка події виконується в кінці кожного кроку викликом визначеного користувачем методу обробки події `event(state)`. У ньому оператор `if` перевіряє визначену умову зі станом `state`. Якщо результат `True`, то подія обробляється, наприклад створюються нові граничні умови і викликається `createCurEqs`. Ви можете легко реалізувати моделювання систем змінної структури, викликаючи в методі `event` конструктор класу `System` (з новими значеннями `els`, `eqs`) та метод `createCurEqs`.

Якщо в компонентах використовуються диференціальні рівняння, то слід розрізняти функції та їхні похідні. Наявність символу "D" у назві змінної означає похідну. Наприклад, `m1_Dx` є похідною `m1_x`. Описи класів таких компонентів будуть більше схожі на класи Modelica. Наприклад Python-клас `Mass` показано в таблиці 3.3 (ліворуч) порівняно з тим самим класом у Modelica (праворуч).

Таблиця 3.3 – Порівняння класів `Mass` в `rusodyn` та в Modelica

<pre>class Mass(Translational1D): def __init__(self, name, m=1.0, x=None, v=None, a=None, f1=None, f2=None, Dx=None, Dv=None): Translational1D.__init__(self, name, locals()) self.eqs=[Eq(self.m*self.a,self.f1+self.f2), Eq(self.a, self.Dv), Eq(self.v, self.Dx)] self.pins=[dict(x=self.x, f=self.f1), dict(x=self.x, f=self.f2)]</pre>	<pre>model Mass extends Transl; parameter Real m(start=1); Real s; Real v(start=0); Real a(start=0); equation m*a=flange_a.f+flange_b.f; a=der(v); v=der(s); flange_a.s = s; flange_b.s = s; end Mass;</pre>
--	--

У цьому випадку для розв'язання DAE у формі $0=F(t, y, y')$ використовується метод `solveDAE` з модуля `pycodynDAE` (лістинг Є.3). Як інтерфейс з розв'язувачами ODE/DAE, такими як SUNDIALS IDA [308] або DASSL [309], використовували Assimulo. Метод `solveDAE` формує метод `residual` та початкові значення для часу, змінних стану та їхніх похідних, необхідних для розв'язувача DAE. Метод отримує значення часу t , стани y , похідні станів y' і повертає вектор нев'язок (нуль, якщо рішення знайдено). Списки аргументів для `residual` готуються методом `residualArgs`. Також можливо створити визначені користувачем функції `state_events` і `handle_event` для відстеження подій та їх обробки для задач з розривами у Assimulo.

Деякі задачі в `pycodyn`, які формулюються за допомогою різницевих або диференціальних рівнянь, можна розв'язувати у символьному виді за допомогою функцій SymPy `solve` та `dsolve`. Зокрема, функція `solve`, яка розв'язує рівняння та системи рівнянь у символьному виді, допомагає формувати згаданий вище метод `seqsf`. Функція `dsolve` розв'язує будь-який тип ODE, що підтримується. Тому DAE потрібно перетворити в ODE шляхом спрощення.

3.4.3 Симуляція вільних коливань штангової колони

Модель `Oscillator` мовою Modelica одноступеневої колони (рис. 3.23) для симуляції вільних коливань наведено у лістингу Є.1. Спочатку колона розтягується переміщенням нижнього кінця (`mass1.s`) на 1 м. Після вивільнення нижнього кінця почнуться вільні коливання. Значення параметрів: `mass1.m=3961.0`, `spring1.c=44650.0`, `spring1.d=2120.7` та з початковими умовами: `mass1.s=-1`, `mass1.v=0`. Параметри симуляції: кінцевий час 10 с., кількість інтервалів 500, точність $1e-006$, метод 'dassl'.

Розроблено аналогічну модель за допомогою `pycodyn` (лістинг Є.4). В окремому модулі створено компоненти: пружина-демпфер `s1` та маса `m1` (рядки 5, 6). Під час їх створення в дужках конструктору передаються значення атрибутів

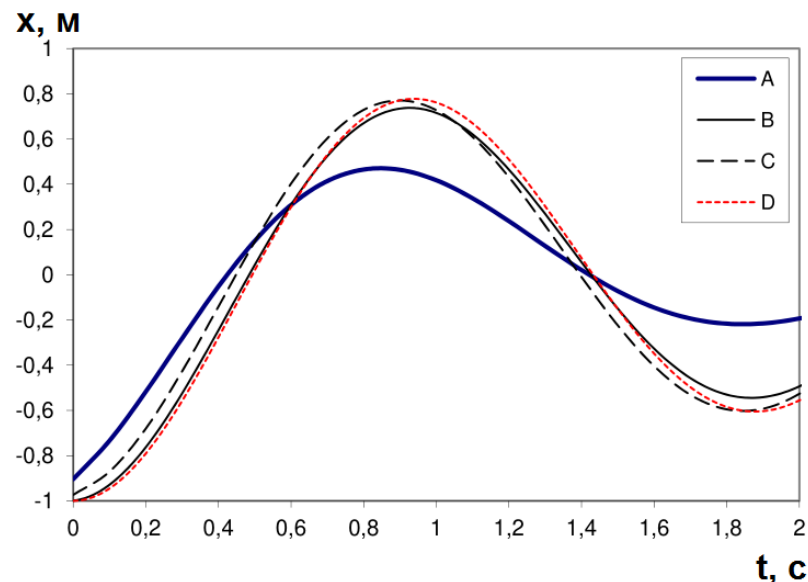
компонента – назва і відомі значення параметрів. Далі (рядки 7, 8) сформовано список додаткових рівнянь `peqs` моделі колони, які утворюються шляхом з'єднання фланців компонентів, та створено об'єкт системи компонентів (модель колони) `s`. Список системи рівнянь моделі колони можна надрукувати за допомогою команди `print s.eqs`. Для отримання рівнянь виключно у символічному вигляді числові значення параметрів `c`, `d`, `m` конструкторів потрібно замінити на `None`:

```
[s1_c*(-s1_x1 + s1_x2) + s1_d*s1_vrel == s1_f2,
-s1_f2 == s1_f1,
s1_vrel == -(s1_x1 - s1_x1p)/dt + (s1_x2 - s1_x2p)/dt,
m1_a*m1_m == m1_f1 + m1_f2,
m1_a == (m1_v - m1_vp)/dt,
m1_v == (m1_x - m1_xp)/dt,
s1_x2 == m1_x, s1_x2p == m1_xp, s1_f2 == -m1_f1]
```

Далі (рядки 11, 12) розв'язано стаціонарну задачу – колона розтягнута на 1 м. Граничні умови залежать від типу динамічної задачі. Якщо розв'язується задача вільних коливань, то переміщення точки підвіски `elsd['s1'].x1` і сила на плунжері `elsd['m1'].f2` приймаються рівними нулю. Функцію `fnBC` створено для оновлення граничних умов в момент часу `t` компонентів. Далі розв'язано динамічну задачу (рядок 21).

Порівняння моделей вільних коливань колони для Python та Modelica показане на рис. 3.24. Відмінності пояснюються застосуванням неоднакових різницевих схем в Python-моделі та розв'язувачі Modelica. За необхідності є можливість застосувати в Python-моделі більш точні різницеві схеми. Наприклад, якщо використовується метод трапецій (лістинг Є.5, Є.6), друге та третє рівняння для `Mass` повинні бути таким:

```
Eq((self.a+self.ap)/2, (self.v-self.vp)/dt),
Eq((self.v+self.vp)/2, (self.x-self.xp)/dt)
```



A – метод Ейлера з кроком часу $dt = 0,1$ с; B – метод Ейлера з кроком часу $dt = 0,01$ с; C – метод трапецій з кроком часу $dt = 0,1$ с; D – DASSL (Modelica), SUNDIALS IDA та точний розв'язок

Рисунок 3.24 – Переміщення плунжера (x) під час вільних коливань колони

Розглянемо використання компонентів з DAE та інтерфейсом Assimulo [310] з модуля `rusodynDAE` (лістинг Є.3). В окремому модулі (лістинг Є.7) створено компоненти і систему таким же чином і спочатку розв'яжемо статичну задачу – колона розтягнута на 1 м (рядки 5-17). Використано розв'язувач SUNDIALS IDA з абсолютною та відносною точністю $1e-06$ для розв'язання динамічної задачі – вільні коливання колони (рядки 18-22).

Показано також розв'язування рівняння в символічній формі, використовуючи ODE-розв'язувач `SymPy` (лістинг Є.8). Підставлено граничні умови в систему рівнянь ($m1.f2 = 0$, $s1.x1 = 0$, $s1.Dx1 = 0$), спрощено систему (рядки 10-20), а після заміни символів на функції та похідні (рядки 24, 25) отримано відомі рівняння гармонічного осцилятора:

$$\begin{aligned} & \text{Eq}(2120.0*m1_v(t)+44650.0*m1_x(t), -3961.0*Derivative(m1_v(t), t)), \\ & \text{Eq}(m1_v(t), Derivative(m1_x(t), t)) \end{aligned}$$

Рівняння розв'язано, використовуючи `dsolve` з початковими умовами $m1_x(0)=-1$, $m1_v(0)=0$ (рядок 27), і отримано відомий розв'язок:

$$\begin{aligned} & \text{Eq}(m1_v(t), 3.36816*\exp(-0.26761*t)*\sin(3.34676*t)), \\ & \text{Eq}(m1_x(t), -(\theta.08*\sin(3.34676*t) + \cos(3.34676*t))*\exp(-0.26761*t)) \end{aligned}$$

3.4.4 Симуляція процесу відкачування

Модель `Pumping` описує процес відкачування (лістинг Є.1). Довжина ходу штока – 2,1 м, кількість подвійних ходів за хвилину – 6,4. Під час ходу вниз вага ШН діє на `flange_b` компонента `mass1`. Під час ходу вверх до неї додається маса рідини. Це показано в секції алгоритму Modelica (рядки 85-90). Аналог цієї односекційної моделі в Python показаний у лістингу Є.9.

В окремому модулі (лістинг Є.10) побудовано модель колони ШН, яка містить дві секції. Модель кожної секції складається з трьох механічних поступальних 1D компонентів: "пружина-демпфер", "маса" та "сила" (рис. 3.25). Компонент "пружина-демпфер" призначений для моделювання пружно-демпферних властивостей секції колони, компонент "маса" моделює інерційні властивості секції, компонент "сила" моделює вагу секції в рідині та інші зовнішні сили, які діють на секцію. Верхня секція має масу 2112 кг, вага в рідині 18494 Н, коефіцієнт пружності 114926 Н/м, коефіцієнт демпфування 5458 Н·с/м. Нижня секція має масу 1850 кг, вага в рідині 16193 Н, коефіцієнт пружності 73021 Н/м, коефіцієнт демпфування 3468 Н·с/м.

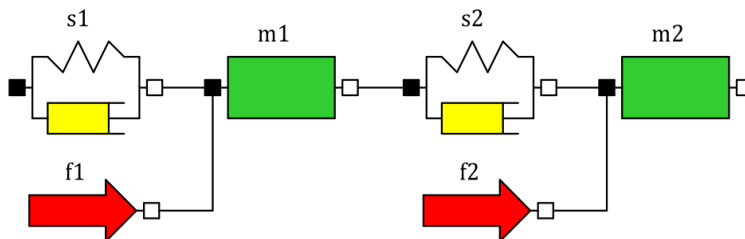


Рисунок 3.25 – Схема двосекційної моделі колони ШН

Присвоєно значення змінним ваги секцій ШН f_s і ваги рідини над плунжером насоса f_r (рядки 4, 5). Створено компоненти: пружина-демпфер першої секції s_1 , маса першої секції m_1 , сила ваги першої секції f_1 , пружина-демпфер другої секції s_2 , маса другої секції m_2 , сила ваги другої секції разом з силою ваги рідини f_2 (рядки 7-12). Сформовано список додаткових рівнянь моделі колони, які утворюються шляхом з'єднання фланців компонентів (рядки

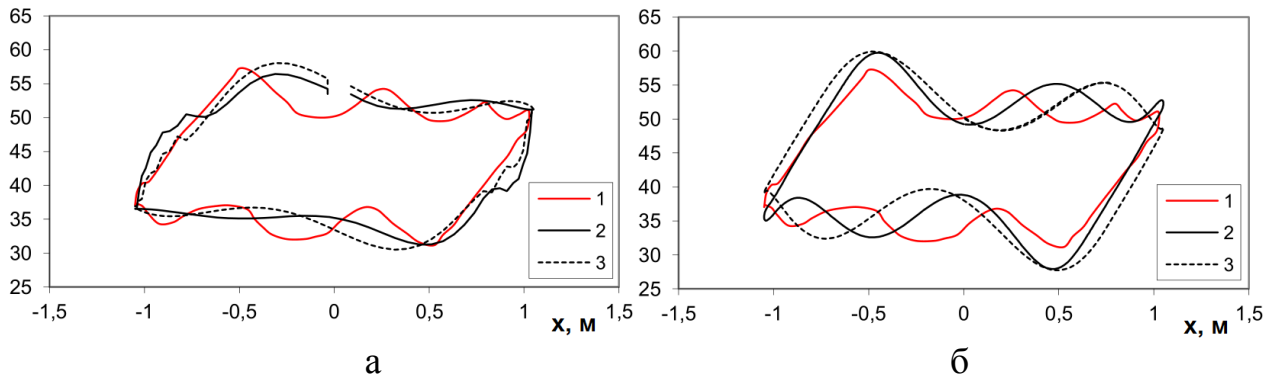
14-17). Створено об'єкт системи компонентів – модель колони (рядок 18). Повний список з системою різницевих рівнянь цієї системи `s.eqs`:

```
[s1_c*(-s1_x1 + s1_x2) + s1_d*s1_vrel == s1_f2,
-s1_f2 == s1_f1,
s1_vrel == -(s1_x1 - s1_x1p)/dt + (s1_x2 - s1_x2p)/dt,
m1_a*m1_m == m1_f1 + m1_f2,
m1_a == (m1_v - m1_vp)/dt,
m1_v == (m1_x - m1_xp)/dt,
s2_c*(-s2_x1 + s2_x2) + s2_d*s2_vrel == s2_f2,
-s2_f2 == s2_f1,
s2_vrel == -(s2_x1 - s2_x1p)/dt + (s2_x2 - s2_x2p)/dt,
m2_a*m2_m == m2_f1 + m2_f2,
m2_a == (m2_v - m2_vp)/dt,
m2_v == (m2_x - m2_xp)/dt,
s1_x2 == m1_x, s1_x2p == m1_xp,
s1_f2 == -m1_f1, m1_x == s2_x1,
m1_xp == s2_x1p, m1_x == f1_x,
m1_xp == f1_xp, m1_f2 == f1_f - s2_f1,
s2_x2 == m2_x, s2_x2p == m2_xp,
s2_f2 == -m2_f1, m2_x == f2_x,
m2_xp == f2_xp, m2_f2 == f2_f]
```

Розв'язано стаціонарну задачу – колона під максимальними статичним навантаженням (рядки 21, 22). Словник `d` містить результати. Щоб вивести значення переміщення нижнього кінця другої секції потрібно ввести команду `print(d[m2.x])`. Отримаємо результат -0.94. Це величина видовження колони під максимальним навантаженням.

Далі розв'язано нестационарну задачу – точка підвіски рухається згідно гармонічного закону. Довжина ходу точки підвіски 2.1 м, кількість подвійних ходів за хвилину 6,4. Функція `motion` (рядки 25-29) описує гармонічний закон руху точки підвісу колони і повертає її переміщення в момент часу `t`. Функція `force` (рядки 31-36) повертає значення сили на плунжері насоса `F` в залежності від значення його швидкості `v`. Якщо швидкість менша нуля (хід колони вниз), функція повертає значення ваги другої секції. В іншому випадку функція повертає суму ваги другої секції та ваги рідини над плунжером. Для забезпечення плавної зміни сили під час зміни знаку швидкості функцію `F(v)` слід згладжувати, наприклад за допомогою функції гіперболічного тангенса `math.tanh`. Створено функцію `fnBC` для оновлення граничних умов в момент часу `t` компонентів

`fnBC.vrs` та розв'язано задачу (рядки 38-45). Аргумент `d` – словник результатів, розрахованих на попередньому кроці. Результати (рис. 3.26) відповідають практичним динамограмам, які отримані на реальних ШСНУ.



1 – практична динамограма; 2 – двосекційна модель; 3 – односекційна модель;
а) – `rusodyn` з методом Ейлера, $dt=0.1$; б) – `rusodynDAE` з SUNDIALS IDA

Рисунок 3.26 – Результати моделювання – динамограми (кН)

Моделювання системи змінної структури (обрив другої секції) методом Ейлера ($dt=0.1$) показано в (лістинг Є.11). Якщо сила у верхній частині другої секції більша 56000 Н, то секція обривається (рис. 3.27). Користувацький метод `event` створений для обробки події (рядки 4-11). На кожному кроці цей метод перевіряє стан `state[s1.f1]>56000`. Якщо результат `True`, то відбувається подія. Оброблення цієї події полягає у зміні компонентів системи (після поломки залишаються лише `s1`, `m1`, `f1`), зміні додаткових рівнянь у точках з'єднання, зміні граничних умов `fnBC` (вага другої секції та вага рідини дорівнюють нулю). Метод `seqsf` також оновлюється за допомогою виклику `createCurEqs`.

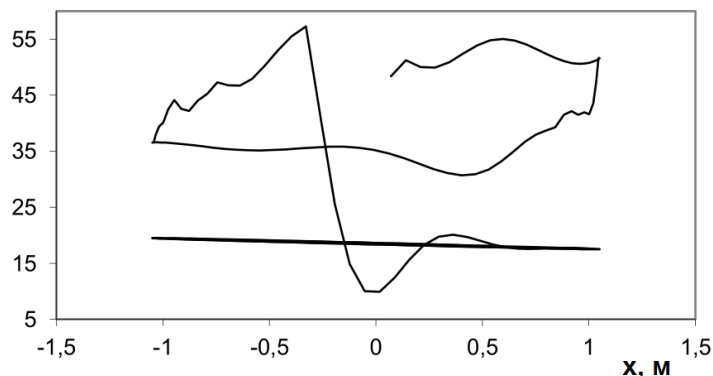


Рисунок 3.27 – Динамограма (кН) з симуляцією обриву другої секції

Аналогічні односекційні (лістинг Є.12) та двосекційні (лістинг Є.13) моделі, які основані на диференціальних рівняннях, розробляються за допомогою модуля `rusodynDAE`. Систему DAE слід доповнити рівняннями, що описують положення верхньої точки та силу, що діє на нижню точку (лістинг Є.12, рядки 21-23 та лістинг Є.13, рядки 29-31). Або можна просто замінити символи `s1.x1` та `m1.f2` на праві частини цих рівнянь.

3.4.5 Аналіз результатів симуляцій

Результати моделювання вільних коливань наведені на рис. 3.24. Частота вільних коливань відповідає теоретичній власній частоті гармонічного осцилятора $\omega=(j/m)^{1/2} = (44650/3961)^{1/2} = 3,357$ рад/с, де j – жорсткість, m – маса. Такі коливання виникають під час нормальної роботи насоса внаслідок різкого зняття або прикладання навантаження (клапан насоса відкривається або закривається) і помітні у верхній і нижній частині динамограми [98]. Відмінності пояснюються використанням неоднакових різницевоїх схем. Результати, отримані аналітично та за допомогою розв'язувачів IDA/DASSL, майже рівні, тому їх показано однією кривою на рис. Значення глобальної помилки (для $t=1$ с) для методів Ейлера ($dt=0.1$), трапецій ($dt=0.1$), Ейлера ($dt=0.01$), DASSL (OpenModelica), методів IDA SUNDIALS відповідно 0,344, 0,034, 0,046, 6,35E-05, 1,46E-05. Тривалість симуляції становить 0,25, 0,5, 2,46, 0,29, 0,028 секунди відповідно. Загальний час виконання модуля (загальний час симуляції для OpenModelica) становить 4,1; 5,5; 6,3; 5,1; 1,9 секунди відповідно. Ці дані були отримані для конфігурації: інтервал симуляції 0-10 секунд, процесор 2,5 ГГц, Python 3.7, NumPy 1.16.4, OpenModelica 1.12, SUNDIALS 2.6.

Результати моделювання процесу відкачування (рис. 3.26) відповідають практичним динамограмам, отриманим на реальних свердловинах [98]. Односекційні (лістинг Є.9) та двосекційні (лістинг Є.10) моделі, з методом Ейлера ($dt=0.1$), дещо спотворюють праву та ліву сторони динамограми та розгладжують верхню та нижню сторони (рис. 3.26а). Односекційні (лістинг Є.12)

та двосекційні (лістинг Є.13) моделі, з IDA, дають дещо більші значення максимального навантаження та дещо менші значення мінімального навантаження (рис. 3.26б). Двосекційна модель є більш адекватною. Динамограма моделі обриву секції (рис. 3.27) відповідає практичним динамограмам з їхніми типовими плоскими формами [98]. Помітно, що глобальна похибка методу Ейлера ($dt=0.1$) набагато більша інших, тому його не рекомендується використовувати для задачі вільних вібрацій колони. Крім того, процедура розв'язування в `rusodyn` має низьку продуктивність, якщо використовуються різниці рівняння. Надалі планується покращити продуктивність, наприклад використовуючи Cython. Відмінності з практичною динамограмою пояснюються тим, що реальна колона має більшу кількість ступенів вільності. Для більш адекватного моделювання потрібно збільшити кількість секцій моделі [24] або використовувати хвильове рівняння (диференціальне рівняння з частинними похідними) [148]. Крім того, важко дізнатися точне значення коефіцієнта демпфування, яке залежить від багатьох факторів [24]. В загальному всі ці моделі можуть бути використані для приблизного моделювання руху колони ШН.

Мова Python дозволяє просту модифікацію та вдосконалення пакету. Надалі планується розширити набір компонентів (наприклад, створити електричні та гідравлічні компоненти), розробити підтримку ієрархічних підсистем та інструментів для побудови моделей за допомогою діаграм компонентів. Для реалізації ієрархічних підсистем можливо перемістити функції для розв'язання рівнянь в окремий модуль і додати піни до класу `System`. Задачу у вигляді різницевого рівняння зазвичай важче сформулювати. Але `SymPy` можна використовувати для автоматизації перетворення диференціальних рівнянь у різниці рівняння. З метою спрощення коду модулів `rusodyn` та `rusodynDAE` алгоритми сортування, усунення та спрощення рівнянь реалізовані не повністю. Це планується здійснити в майбутньому за допомогою `SymPy`. На даний час розроблено тільки просту процедуру спрощення системи рівнянь (лістинг Є.14). Вихідний код вільно доступний за ліцензією GNU GPLv3 з відкритим кодом на GitHub [25].

3.5 Компонентно-орієнтоване моделювання кінематики механізмів мовою Python на прикладі механізму верстата-качалки

Було також поставлено завдання реалізувати компонентно-орієнтований підхід для моделювання кінематики механізмів універсальною імперативною мовою програмування Python. Опишемо принципи побудови програми для моделювання кінематики механізмів на прикладі механізму ВК СКД8-3-4000 [311] (лістинг Ж.1).

Наведений у лістингу Ж.1 клас `Frame` описує множину однотипних компонентів-ланок механізмів, крайні точки яких задані координатами x_1 , y_1 , x_2 , y_2 , а довжина – змінною L . Функція `eqs()` повертає систему рівнянь, яка описує поведінку компонента. В даному випадку система містить тільки одне рівняння, яке описує незмінну відстань L між точками. Функція `plot()` рисує компонент у вигляді лінії за допомогою графічної бібліотеки `matplotlib`.

Наступний клас `Connector` описує множину однотипних компонентів, які являють собою шарнірне з'єднання двох ланок. Тут e_1, e_2 – дві ланки, які з'єднуються точками 2 і 1 відповідно. Функція `eqs()` повертає систему рівнянь, яка описує рівність координат точок з'єднання ($e_1.x_2=e_2.x_1$, $e_1.y_2=e_2.y_1$). Функція `plot()` рисує компонент у вигляді червоної точки.

Аналогічно можна розробити клас `Connector2`, який описує нерухоме з'єднання двох ланок. Його функція `eqs()` крім рівнянь, які наведені у класі `Connector`, повинна повертати рівняння, яке описує рівність кутів повороту ланок ($\text{tg } a_1 = \text{tg } a_2$).

Наступний клас `System` описує систему компонентів – плоский механізм. Тут e – список компонентів механізму. Функція `eqs()` повертає систему усіх рівнянь, які описують поведінку механізму. В цю систему рівнянь автоматично додаються рівняння кожного компонента системи. Функція `plot()` рисує механізм шляхом виклику функції `plot()` усіх компонентів системи.

Далі на основі розроблених класів створено об'єкти: кривошип (`fr0`), шатун (`fr1`), заднє плече балансира (`fr2`), переднє плече балансира (`fr3`), шарнір

між кривошипом і шатуном ($con0$), шарнір між шатуном і балансиrom ($con1$), нерухоме з'єднання плечей балансира ($con2$), механізм ВК (s).

Для симуляції руху ВК можна поступово змінювати кут повороту кривошипа від початкового значення до кінцевого з невеликим кроком, наприклад у циклі `while`. Тіло цього циклу повинно містити команди обчислення координат точки кривошипа за кутом його повороту, виклик функції розв'язування системи рівнянь об'єкта `s`, запису або візуалізації результатів (`s.plot()`) та збільшення кута на крок. Для розв'язування системи рівнянь можна використати функцію `root()` з пакету `scipy.optimize` [170], яка розв'язує систему чисельним методом Левенберга-Марквардта. Функція `root()` потребує наближених значень коренів, які можна взяти з попередньої ітерації. Результатом симуляції будуть значення невідомих координат точок механізму в різні моменти часу. Після цього можна розрахувати швидкості та прискорення точок шляхом диференціювання результатів функціями `diff()` або `gradient()` з пакету `scipy` [170] (рис. 3.28).

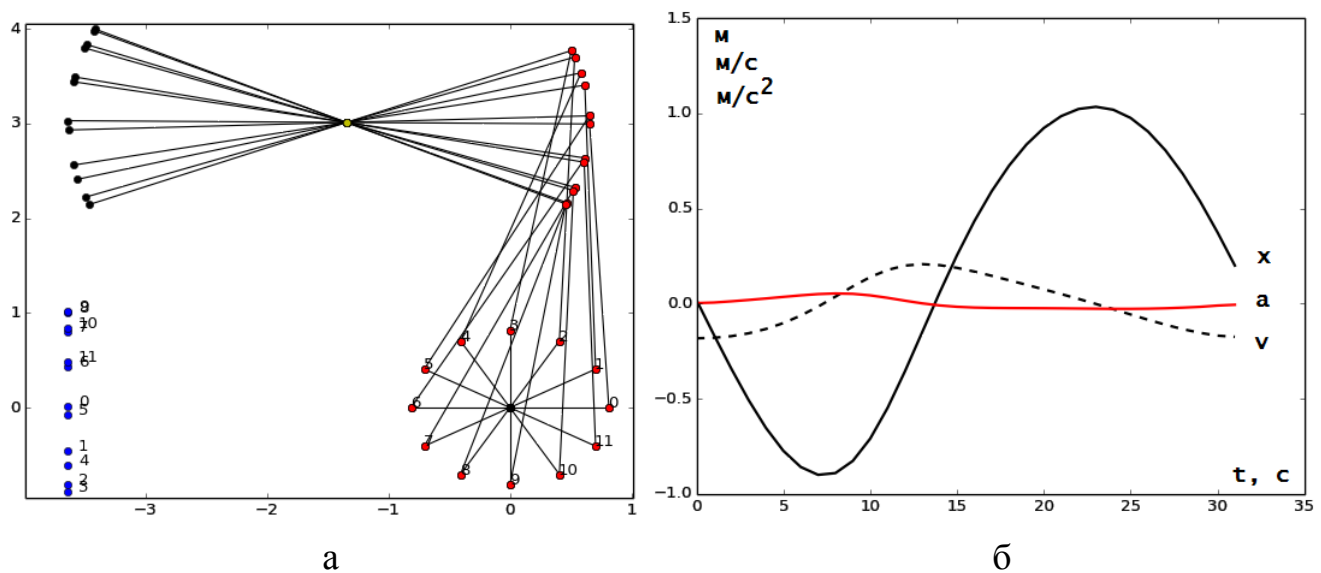


Рисунок 3.28 – Положення ланок механізму (а) і графік переміщення $x(t)$, швидкості $v(t)$ і прискорення $a(t)$ точки підвісу (б)

Запропоновані принципи можуть бути розвинені у повноцінну систему моделювання кінематики і динаміки різнотипних складних механізмів без

необхідності застосування спеціалізованих засобів моделювання. Для цього необхідно розробити компоненти, які описують кінематичні пари різного типу. Розроблена програма (лістинг Ж.1) може бути використана для експрес-аналізу кінематики механізмів ВК нового типу.

3.6 Принципи побудови тривимірної параметричної моделі верстата-качалки в SOLIDWORKS

З метою полегшення аналізу кінематичних і динамічних характеристик та оптимізації конструкції ВК розроблено його тривимірну параметричну модель [312]. Модель є подібною на об'єкт моделювання за геометрією та масою, але не містить більшість дрібних деталей (кріпильні деталі, підшипники, втулки, дрібні геометричні елементи), які не є суттєвими для моделювання динамічних та кінематичних характеристик в SOLIDWORKS Motion. Модель також подібна на об'єкт моделювання за жорсткістю деталей, що дозволяє моделювати їхню податливість в програмі кінематичного та динамічного аналізу механізмів MSC Adams. Модель складається з параметричних моделей деталей та вузлів у SOLIDWORKS та відповідних класів мовою програмування Python (лістинг 3.1), в яких описуються та розраховуються параметри цих моделей. Це дозволяє легко перебудувувати модель у разі зміни значень її параметрів, а також інтегрувати цю модель в ІС [30].

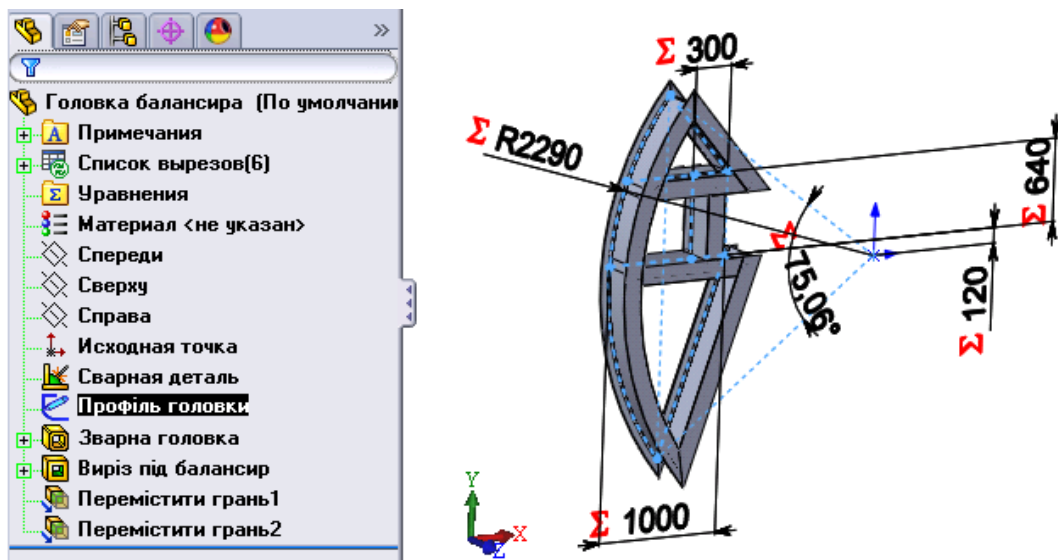
Клас моделі SOLIDWORKS `Swmodel` є базовим класом будь-якої параметричної моделі SOLIDWORKS. Він містить атрибути `fileName` (назва файлу моделі) та `d` (словник параметрів). Для розрахунку параметрів моделі цей клас містить функцію `create()`, а для перебудови моделі SOLIDWORKS із заданими значеннями параметрів – функцію `rebuildModel()`. Перебудову моделі можна реалізувати за допомогою інтерфейсу програмування SOLIDWORKS. Однак для доступу до функцій цього інтерфейсу необхідно встановити додаткові бібліотеки Python (наприклад `PyWin32`). Щоб цього не

робити, в класі SWmodel розроблено функцію runSWMacro(), в якій python програма створює процес, що виконує наступний сценарій Windows Script Host мовою VBScript:

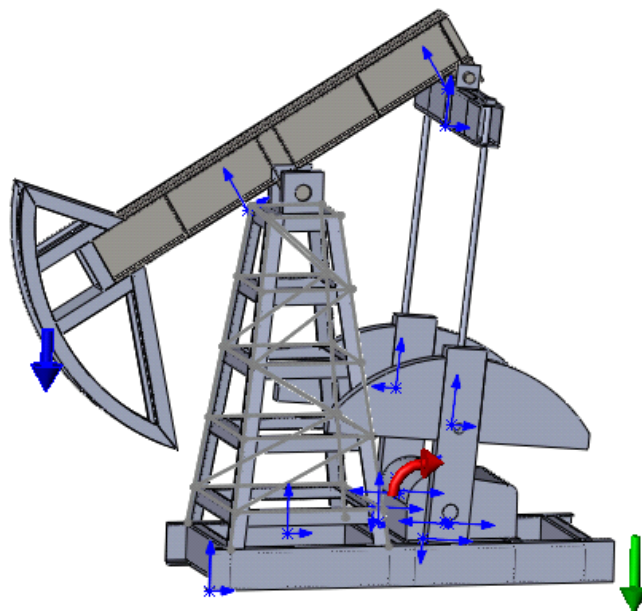
```
Dim swApp
Set swApp = CreateObject("SldWorks.Application")
boolstatus = swApp.RunMacro("{filePathName}", "{moduleName}",
                            "{procedureName}")
Set swApp=Nothing
```

У свою чергу цей сценарій може виконати будь-який макрос SOLIDWORKS з назвою filePathName. Іншим способом зміни значень параметрів моделі є зміна вмісту текстових файлів рівнянь моделі SOLIDWORKS. Для цього в класі SWmodel створено функцію write_dict_to_SW_equations(). Передбачено також функцію read_dict_from_SW_equations(), що виконує читання з цих файлів.

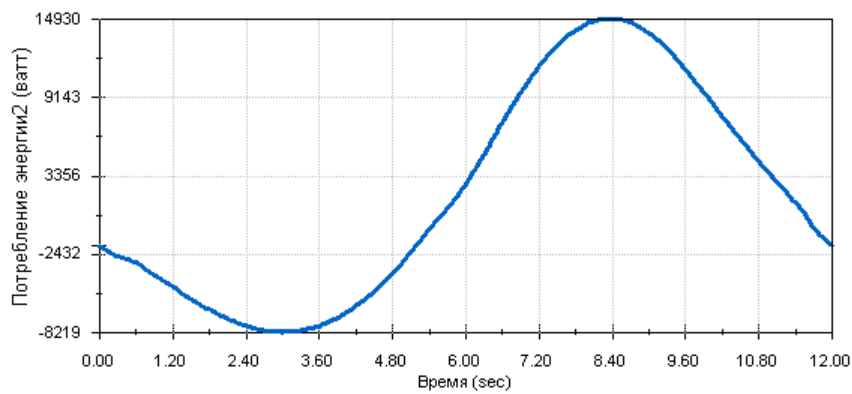
Інші класи програми описують моделі деталей та вузлів ВК та успадковують базовий клас SWmodel. Зокрема це класи деталі SWmodelPRT (з похідними класами Balansir, Shatun, Krivoshyp та ін.) та складальної одиниці SWmodelASM (з похідними класами PumpingUnit, BalansirVZbori та ін.). Клас Profil описує стандартні профілі, такі як двотавр, швелер і кутник, та може читати їхні параметри з бази даних (файл CSV). На рис. 3.29 показано модель головки балансира, ВК та результати симуляції з наступними значеннями параметрів: ВК СКД8-3-4000; навантаження на балансир $P_{min}=10000$ Н, $P_{max}=20000$ Н; довжина ходу полірованого штока 2000 мм; кількість гойдань 5 хв^{-1} ; кількість противаг 4; радіус центра мас противаг 914 мм. В моделі враховується сила гравітації. Ті деталі, які розташовані одна відносно іншої нерухомо, були об'єднані в жорсткі групи з метою зменшення часу розрахунку моделі.



а



б



в

Рисунок 3.29 – Модель головки балансира (а), ВК (б)

та результати симуляції у SOLIDWORKS Motion – ватметрограма (в)

Дана модель володіє вищою точністю та є зручнішою у використанні ніж відповідні математичні аналітичні моделі. Вона може використовуватись для удосконалення конструкції ВК та оптимізації їхніх параметрів. Наприклад, для оптимізації радіус центра мас противаг необхідно поступово змінювати цей радіус на малу величину до тих пір, поки неурівноваженість ВК (за даними ватметрограми) не зменшиться до мінімуму.

3.7 Висновки до розділу

1. Розроблена автоматна модель ШСНУ [294, 295] є простою для розуміння і модифікації, відповідає концепції моделювання "знизу вверх", дозволяє моделювати явища, які важко сформулювати в термінах диференціальних рівнянь, наприклад стохастичну поведінку або складні граничні умови чи накопичення пошкоджень. Програма виконана в стилі автоматного програмування популярною мовою Python, не потребує спеціалізованих засобів моделювання чи сторонніх бібліотек та допускає паралельні обчислення. На даний час програма використовує більше обчислювальних ресурсів ніж моделі на основі хвильового рівняння, тому наступним етапом досліджень буде оптимізація її алгоритму. Модель може бути використана для оптимізації параметрів ШСНУ і як основа для побудови складніших моделей.

2. Побудову компонентно-орієнтованої моделі ШСНУ слід починати з простих моделей пружної колони і гідравлічної частини. Базовими компонентами для побудови моделі пружної колони є "маса", "поступальна пружина з демпфером" і "поступальна сила". Для моделювання факторів, які розподілені нерівномірно вздовж колони, колону слід поділяти на підсистеми (секції), кожна з яких містить ці базові компоненти. У викривлених свердловинах для моделювання різних сил тертя найзручніше використовувати компоненти "поступальне тертя" та "поступальна сила". Для спрощення зміни значень параметрів компонентів бажано використовувати засоби MapleSim API. Розроблена модель [24, 26, 301] володіє достатньою для практичного

використання адекватністю та можливістю простої її модифікації та удосконалення. Модель може бути використана як САПР ШСНУ для обґрунтування нових конструкторських та технологічних рішень, зокрема уточнення комплектування колони ШН, застосування склопластикових ШН, нових типів приводу або систем адаптивної зміни частоти обертання електродвигуна для досягнення білярезонансних режимів роботи колони [23]. Можливими напрямками удосконалення цієї моделі є: модель руху пружної колони НКТ, змінна по глибині в'язкість рідини, уточнення моделі клапанів насоса, уточнення моделі тертя ШН об стінки НКТ. Наприклад, сили тертя ШН об стінки НКТ виникають у викривленій свердловині, але і можуть виникнути у нижній частині колони внаслідок її викривлення під час руху вниз.

3. В Abaqus/CAE розроблено параметричну СЕМ кульового зворотного клапана свердловинного штангового насоса для Abaqus/CFD [110]. Модель може бути використана для автоматизованого отримання залежностей коефіцієнта витрати клапана від висоти підйому кульки. Особливістю розробленої моделі є можливість автоматизованого звернення до неї за результатами моделювання з зовнішніх програм. Зокрема, ця модель може використовуватись як компонент динамічної моделі клапана, розробленою мовою моделювання систем Modelica.

4. Розвинуто алгоритмічні основи та реалізовано програмний каркас для створення подібних на Modelica, компонентно-орієнтованих акаузальних моделей динамічних систем мовою Python без необхідності вивчення та застосування спеціалізованих мов моделювання [25, 28, 307]. На їхній основі розроблено динамічні моделі штангових колон та ВК [311]. Для опису компонентів можуть бути використані різницеві або диференціальні рівняння. Використання різницевих рівнянь та модуля `rusodun` дозволяє спростити реалізацію гібридного моделювання, моделювання систем змінної структури та вимоги до модулів для символічної математики та для розв'язування рівнянь. Він також добре підходить для експериментів з різними можливостями моделювання. Зокрема, можна експериментувати з довільними різницевими схемами, робити довільні символічні маніпуляції та змінювати процедури чисельної симуляції. Однак модуль

rusodynDAE може забезпечити більш високу точність та продуктивність, використовуючи сторонні розв'язувачі DAE, які підходять для жорстких проблем. За допомогою SymPy деякі задачі можна розв'язувати у символічному вигляді. Порівняння результатів моделювання колон із практичними динамограмами та результатами моделювання мовою Modelica підтверджують адекватність моделей. Запропонований підхід спрощує розуміння системи, її модифікацію та вдосконалення, адаптацію для інших потреб, робить її доступною для значно більшої спільноти, спрощує інтеграцію в сторонні програми. Каркас може бути використаний для вивчення принципів компонентно-орієнтованого моделювання та проведення різного роду експериментів над його новими можливостями.

5. Розроблена тривимірна параметрична модель ВК володіє вищою точністю (подібна на об'єкт моделювання геометричними, інерційними і жорсткісними властивостями) та є зручнішою у використанні ніж відповідні математичні аналітичні моделі. Вона може використовуватись для удосконалення конструкції ВК та оптимізації їхніх параметрів. Завдяки використанню ООП, його принципу успадкування класів та високорівневих конструкцій Python модель має здатність до простого розширення та може легко інтегруватися в ІС [312].

Розроблені моделі можуть також бути використані для дослідження умов роботи елементів колон ШН та НКТ, зокрема для уточнення навантажень для їхніх СЕМ, що розглядаються в наступних розділах.

РОЗДІЛ 4

ПАРАМЕТРИЧНІ СКІНЧЕННО-ЕЛЕМЕНТНІ МОДЕЛІ РІЗЬБОВИХ З'ЄДНАНЬ ШСНУ

4.1 Принципи побудови параметричних скінченно-елементних моделей різьбових з'єднань в Abaqus/CAE

Для комплексного розв'язання проблеми незадовільної працездатності РЗ ШСНУ необхідно розробити принципи побудови та ефективного використання їхніх параметричних СЕМ. Нижче описані принципи побудови параметричних СЕМ РЗ в Abaqus/CAE мовою Python [93, 313-316]. Принципи полягають у застосуванні розробленого автором програмного каркасу на базі Abaqus API з алгоритмами, функціями, класами та об'єктами мовою Python для створення та використання елементів осесиметричних або тривимірних СЕМ РЗ та прикладних САПР РЗ. Основні елементи цих моделей показано на рис. И.1, параметричні ескізи деталей і сітка СЕМ – на рис. И.3-8. Допоміжні компоненти для побудови моделей наведено у модулі `tools` (лістинг И.1.).

Клас `Material` (лістинг И.1) описує поняття матеріалу і містить члени-дані, які описують механічні характеристики пружності та пластичності, та члени-функції, які повертають ці механічні характеристики в заданому форматі. Бібліотека матеріалів `matlib` може бути створена за допомогою Python-словника, де кожен елемент є парою "рядок назва матеріалу"- "об'єкт класу `Material`". В класі `Material` істинне напруження та деформація визначаються так [317]:

$$\sigma = \sigma_{ном} (1 + \varepsilon_{ном}), \quad \varepsilon = \ln(1 + \varepsilon_{ном}),$$

де $\sigma_{ном}$, $\varepsilon_{ном}$ – номінальне напруження і деформація.

Ділянка від σ_m до σ_v описується за допомогою степеневі залежності [317], яка потім апроксимується лініями

$$\varepsilon = \varepsilon_m \left(\frac{\sigma}{\sigma_m} \right)^n,$$

де σ , ε – напруження та відповідна деформація,

σ_m , ε_m – границя плинності та відповідна деформація,

n – показник, який визначається з умови проходження через точку $(\varepsilon_\sigma, \sigma_\sigma)$.

Істинна границя міцності та відповідна деформація визначаються так:

$$\sigma_\sigma = \sigma_{\sigma.ном} (1 + k\delta), \quad \varepsilon_\sigma = \ln(1 + k\delta),$$

де k – коефіцієнт (приймали $k=0,4$),

δ – відносне видовження.

Діаграма деформування на ділянці від σ_σ до σ_k описуються лінійною залежністю. Істинне напруження та деформація в момент руйнування [317]:

$$\sigma_k = K\sigma_{\sigma.ном} (1 - \psi), \quad \varepsilon_k = \ln(1 / (1 - \psi)),$$

де ψ – відносне звуження,

K – коефіцієнт руйнуючого навантаження (приймали $K=0,8$).

На рис. И.2 показані істинні діаграми деформування сталей для ШН і муфт (пластична ділянка).

Основою будь-яких параметричних моделей деталей є поняття розміру. Python-клас `Dim` (лістинг И.1) описує поняття розміру деталі. Будь-який розмір деталі має властивості: номінальне значення, верхнє відхилення, нижнє відхилення і дійсне значення. Використовуючи ці дані можна обчислити максимальний і мінімальний допустимі розміри. Отже клас `Dim` містить такі члени-дані: `n`, `ei`, `es`, `v` і члени-функції: `__init__()`, `min()`, `max()`. Словник розмірів РЗ певного типорозміру, наприклад `nkt114`, містить пари "рядок назва розміру": "об'єкт класу `Dim`", наприклад `'D':Dim(114.3, 0.0, 0.0)`. Словник стандартних топорозмірів РЗ, наприклад `nkt`, містить пари "типорозмір": "словник розмірів РЗ", наприклад `114:nkt114`. Тоді присвоїти дійсне значення розміру з назвою `'d'` РЗ типорозміром 114 можна так: `d=nkt[114]; d['d'].v=d['d'].max()/2`.

Програма може самостійно створювати ескізи профілів заготовок деталей РЗ та профілів їхньої різьби в осьовому перетині (див. функції `createSketch1-4`

листинга И.2). Для створення ескізу використовується метод моделі `ConstrainedSketch`, а методи ескізу `Line`, `VerticalConstraint`, `VerticalDimension`, `Parameter` створюють лінію, обмеження ескізу, розмір та параметер відповідно.

Для пошуку геометричних елементів моделі (точок, ребер, граней) в Abaqus Python API використовується функція `findAt`, параметром якої є послідовність з трьох координат точок. Тому потрібно створити характерні точки ребер моделі, на яких задаватимуться навантаження, граничні умови чи параметри контакту, наприклад `en1=((d['D'].v+d['d'].v)/2,d['L'].v+20,0.0)`.

Для надання значення параметру розміру ескізу використовується метод `setValues` (див. код функцій `set_values` та `set_values2` у листингу И.1).

Для створення осесиметричної геометричної моделі (Part) заготовки деталі РЗ (ніпеля або муфти) використовується метод моделі `Part` та метод деталі `BaseShell`, аргументом якого є об'єкт ескізу (див. код функції `createPart` у листингу И.1).

Побудова профілю різьби деталей РЗ є найскладнішим завданням, оскільки неточна побудова може призвести до некоректної геометрії або геометрії з дуже дрібними елементами, яка ускладнює сітку скінченних елементів. Функція `createCut(Part, Sketch, Begin, P, Fi, Len, X, Y, dx, dy)` (лістинг И.1) створює частину профілю різьби шляхом послідовного переміщення ескізу профілю на величину кроку і створення ним вирізу за алгоритмом:

```
i=Begin
ПОКИ i*P<=Len:
    Перемістити ескіз Sketch профілю інструмента в точку
        (X+dx*P*tan(Fi*pi/180)*i, Y+dy*P*i)
    Зробити виріз цим профілем в заготовці деталі Part
    i=i+1
```

Де `Part` – назва деталі, `Sketch` – назва ескізу, `Begin` – початок різьби (ціле), `P` – крок різьби, `Fi` – кут конуса конічної різьби (градуси), `Len` – довжина різьби, `X, Y` – початкові координати центра профілю, `dx` – радіальний напрямок подачі під час "нарізання" різьби (+1 – вправо, -1 – вліво), `dy` – осьовий напрямок подачі (+1 –

вверх, -1 – вниз), i – номер витка (0-перший). Приклади її використання є у функціях `createProfile` (лістинги И.3, И.4). Використовуючи функцію `createCut` можна створювати циліндричні та конічні різьби, збіги різьби та інші різьбові поверхні.

Матеріал деталі РЗ створюється за допомогою методу моделі `Material`, а механічні характеристики пружності пластичності створюються за допомогою методів матеріалу `Elastic` та `Plastic` (див. функцію `createMaterial` у лістингу И.1).

Секція (частина деталі з одного матеріалу) створюється методом моделі `HomogeneousSolidSection`, а присвоюється вона деталі за допомогою методу деталі `SectionAssignment` (див. функцію `createSectionAssign` у лістингу И.1).

Елемент збирання РЗ створюється методом базового збирання (`rootAssembly`) `Instance`, аргументом якого є деталь моделі (див. функцію `createAssemblyInstance` у лістингу И.1).

Крок статичного навантаження створюється методом моделі `StaticStep` з указанням попереднього кроку (див. функцію `createStep` у лістингу И.1). Можна, наприклад, створити два кроки навантаження. На першому створюється згинчування РЗ, на другому – зовнішнє навантаження.

Множина ребер, що контактують, створюється шляхом пошуку ребер функцією `findAt`. Для цього використовуються ребра, координати точок яких відомі (див. функцію `createContactSet` у лістингу И.1).

Властивості контакту (`interactionProperties`) створюються методом моделі `ContactProperty`. Зокрема в них вказується формулювання контакту, коефіцієнт тертя та можливість розділення поверхонь (див. функцію `createContactProperty` у лістингу И.1).

Модель контакту поверхонь створюються методом моделі `SurfaceToSurfaceContactStd`, аргументами якого є головна і підпорядкована поверхня, крок навантаження, властивості контакту, тип ковзання та інтерференції та ін. (див. функцію `createContact` у лістингу И.1).

Множини ребер для навантаження і граничних умов створюються шляхом їхнього пошуку функцією `findAt` за відомими координатами їхніх точок (див. функцію `createBCSet` у лістингу И.1).

Навантаження і граничні умови створюються на заданому кроці навантаження методами моделі `Pressure`, `DisplacementBC`, `EncastreBC` (див. функції `createBC_Pressure`, `createBC_Axis`, `createBC_Encastre` у лістингу И.1). Моделювання згвинчування муфтового РЗ ШН або замкового РЗ бурильних труб можна реалізувати за допомогою функції `BoltLoad`, якою задається видовження ніпеля або скорочення муфти під час згвинчування Δ (величина згвинчування) (див. функцію `createBC_BoltLoad` у лістингу И.1). Моделювання згвинчування муфтового РЗ НКТ виконується шляхом осьового зміщення елемента збирання методом `translate`, наприклад муфти відносно ніпеля на величину nA , яка кратна кроку різьби ($A=3,175$). Якщо $n=0$ – це згвинчування "вручну", якщо $n=1$ або $n=2$ – на верстаті (лістинг И.4).

Від розміру і кількості скінченних елементів суттєво залежить точність результатів і тривалість обчислень. Загальний розмір скінченних елементів деталі вказується методом збирання `seedPartInstance`. Вздовж дрібних ребер (наприклад різьби) слід вказати менший розмір (більшу кількість) скінченних елементів методом `seedEdgeByNumber`. Сітка скінченних елементів створюється методом `generateMesh` (див. функцію `createMesh` у лістингу И.1).

Об'єкт задачі створюється методом `mdb.Job`, розв'язується задача методом `submit`, а очікування її завершення виконується методом `waitForCompletion` (див. функцію `createJobSubmit` у лістингу И.1).

База даних результатів відкривається функцією `openOdb`. У ній на заданому кроці навантаження, як правило на останньому фреймі, знаходять об'єкт поля заданої величини (наприклад напруження) `fieldOutputs`. Метод цього об'єкта `getSubset` дозволяє отримати значення величини в заданій зоні РЗ (див. функції `readODB_path`, `readODB_set`, `readODB_set_`, `readODB_set2` у лістингу И.1). Таким чином можна знайти значення максимального напруження σ_m в заданій зоні РЗ. Наступним етапом є порівняння знайденого значення з

допустимим напруженням. За результатами порівняння може бути прийняте рішення про перехід до наступного наближення (зміни певного вхідного параметра, перебудови і розрахунку моделі), або про завершення ітерації. Таким чином реалізується оптимізація конструкції РЗ.

На базі цих принципів автором розроблено програми для побудови і аналізу моделей муфтових РЗ ШН, НКТ і замкових РЗ (лістинги И.2-7). Така методика ефективна для комплексного аналізу і оптимізації різнотипних РЗ. Для зменшення трудомісткості побудови параметричних геометричних моделей можна також використовувати такі САПР як SOLIDWORKS, CATIA, Pro/ENGINEER та їхні інтерфейси з Abaqus. Розглянемо програмну модель муфтового РЗ ШН (лістинги И.2, И.5) [93]. Геометрія моделі РЗ відповідає ГОСТ 13877-96. Геометричні параметри деталей описуються в програмі словниками `rod19`, `rod22`, `rod25`, а типорозміри РЗ – словником `rod`. Кожен параметр має номінальне значення, та верхнє и нижнє допустимі відхилення. Розраховувався найменш міцний стандартний варіант РЗ – з мінімальними допустимими розмірами різьби ніпеля і максимальними допустимими розмірами різьби муфти. Словник `d` містить значення розмірів ескізів деталей. Діаметральні розміри перетворюються в радіусні, наприклад зовнішній діаметр різьби $d['d_n']=d['d_n'].min()/2$. Створюються допоміжні параметри моделі, наприклад точка, яка належить верхньому торцю ШН $en1=(d['dn']/2, d['l1n']+2\theta, 0.0)$. Матеріали створюються за допомогою словника `matlib`, який містить бібліотеку матеріалів з механічними характеристиками відповідно ГОСТ (клас `Material`). Функція `power(8)` повертає словник еластичних і пластичних властивостей, де 8 – кількість ліній для апроксимації пластичної ділянки, наприклад `mat1=matlib['40'].power(8)`. Зовнішнє навантаження створюється відповідно напруженню розтягу в тілі ШН діаметром $d\theta$, тому величина напруження множиться на $d\theta^2/dn^2$, де dn – діаметр бурта ніпеля, наприклад `load2=-155.1e+6*d['d\theta']**2/d['dn']**2`.

Функція `create_nipple_coupling()` створює геометрію за ескізами заготовок деталей. Параметричні ескізи (рис. И.3) можуть бути створені вручну

або програмним шляхом (за допомогою функцій `createSketch1()`, `createSketch2()`, `createSketch3()`, `createSketch4()`). Функція `set_values()` присвоює значення параметрам ескізу. Функція `createPart()` створює осесиметричну деталь, а функція `createProfile()` – профіль різьби муфти і ніпеля. Іншим способом побудови геометрії є функції `create_nipple_coupling2()` і `create_nipple_coupling3()`, які будують геометрію з простих елементів. Для цього розроблено класи `Line` (лінія) та `N_angle` (багатокутник зі скругленнями). Ці класи мають функцію `drawAbaqus()`, яка рисує об'єкт на заданому ескізі моделі Abaqus. Функція `part_builder()` дозволяє будувати деталі з простих елементів шляхом додавання або вирізання фігури булевою операцією.

Розроблено також функції для побудови тривимірних моделей. Назва таких функцій закінчується на "3D", наприклад `createPart3D()`. Перелік інших функцій для побудови моделі, наведено нижче в порядку їх виклику. Усі ці функції є незалежними від моделі та описані в модулі `tools.py` (лістинг И.1): `createPartition()` – ділить поверхню деталі лінією, що необхідно для моделювання згвинчування РЗ функцією `boltload`; `createMaterial()` – створює механічні характеристики матеріалу; `createSectionAssign()` – створює і присвоює секції деталі; `createAssemblyInstance()` – створює елемент збирання; `createStep()` – створює крок навантаження; `createContactSet()` – створює множину ребер для контакту; `createContactProperty()` – установлює властивості контакту; `createContact()` – створює контакт; `createBCSet()` – створює множину ребер для граничних умов; `createBC_Pressure()` – створює зовнішнє навантаження (рис. И.4); `createBC_Axis()` – створює граничні умови на осі (неможливість переміщення в радіальному напрямку (рис. И.4)); `createBC_Encastre()` – створює закріплення (рис. И.4); `createBC_BoltLoad()` – створює зусилля згвинчування (рис. И.4); `createMesh()` – створює сітку скінченних елементів (рис. И.4); `createEdgesSet()` – створює множину ребер для результатів;

`createVerticesSet()` – створює множину вершин для результатів;
`createJobSubmit()` – створює задачу і виконує її.

Результати створюються шляхом виклику функції `createResults()`. Її реалізація залежить від постановки задачі. Наприклад, функція `SF_field()` розраховує поле коефіцієнта D , а функція `readODB_set()` (або `readODB_set2()`, `readODB_path()`) читає результати з останнього фрейму вказаного кроку навантаження на заданій множині геометричних елементів.

Для побудови моделі замкового РЗ використовуються ті ж принципи, що і для попередньої (лістинги И.3, И.6) [93]. Геометрія моделі РЗ відповідає ГОСТ 5286-75. Геометричні параметри деталей РЗ описуються в програмі словниками `zn80`, `zn95`, а типорозміри РЗ – словником `замок`. На рис. И.5 показні параметричні ескізи для побудови моделі РЗ, а на рис. И.6 – СЕМ з'єднання. Для побудови моделі муфтового РЗ НКТ використовуються ті ж принципи, що і для попередньої (лістинги И.4, И.7) [93]. Геометрія моделі РЗ відповідає ГОСТ 633-80. Геометричні параметри деталей РЗ описуються в програмі словниками `nkt114`, `nkt102`, а типорозміри РЗ – словником `nkt`. На рис. И.7 показні параметричні ескізи для побудови моделі РЗ, а на рис. И.8 – СЕМ з'єднання.

4.2 Принципи побудови параметричних скінченно-елементних моделей різьбових з'єднань на основі вільного програмного забезпечення

В підрозділі описано принципи розробки системи для скінченно-елементного аналізу та автоматизованого проектування осесиметричних моделей відповідальних РЗ нафтогазового обладнання на основі вільного програмного забезпечення [187]. До таких з'єднань відносяться РЗ ШН, замкові РЗ бурильних труб, РЗ НКТ. Розроблена прикладна САПР РЗ дозволить автоматизовано отримувати залежності напружень в заданих зонах від параметрів з'єднання і оптимізувати конструкцію. Для програмування використаємо Python 2.7.

Спочатку програма буде геометричну модель РЗ за допомогою `pythonOCC`. Модель повинна бути параметричною, тобто дозволяти змінювати значення

окремих параметрів. Побудову дуже зручно виконувати за допомогою відносно простих фігур, наприклад полігонів зі скругленнями, які потім утворюють складну форму шляхом виконання над ними булевих операцій (об'єднання тіл, виріз тілом, спільне тіло). Як альтернатива може бути використана відома відкрита САПР FreeCAD або відкритий пре- и постпроцесор для моделювання чисельними методами SALOME. Вони теж розроблені на основі OCCT та мають інтерфейс програмування мовою Python.

Після побудови геометрична модель зберігається у файлі `model.brep` (у внутрішньому форматі OCCT BRep) для передачі її Gmsh. Gmsh може бути виконана з Python в окремому процесі. Параметри для генерації сітки елементів можуть бути передані Gmsh в командному рядку або записані у файлі сценарію `.geo`. До цих параметрів належать: назва файлу геометричної моделі, алгоритм генерації сітки, тип елементів та їхній порядок, вихідний формат. Для осесиметричних моделей підходять елементи типу CAX3, CAX4, CAX6 або CAX8. Gmsh згенерує текстовий вхідний файл `model.inp`, який сумісний з Abaqus. Цей файл містить список вузлів сітки, список елементів на ребрах і список елементів на поверхнях. CalculiX потребує дуже незначної модифікації цього файлу, зокрема список елементів на ребрах потрібно перетворити у список вузлів на ребрах.

Для роботи розв'язувача CAX потрібно доповнити цей файл описами механічних характеристик матеріалу, контактних пар, граничних умов та навантажень на заданих вузлах, ребрах і поверхнях. В програмі повинні бути функції, які дозволяють знайти в моделі потрібний вузол, ребро чи поверхню. Наприклад, в геометричній моделі (`model.brep`) ідентифікувати ребро можна за допомогою точки на ньому, яка лежить між його крайніми вершинами. В CEM (`model.inp`) ребро можна знайти за допомогою точок його крайніх вузлів. Ці точки повинні збігатися з крайніми вершинами ребра в геометричній моделі. Таким чином можливо виявити, яке ребро в CEM відповідає заданому ребру геометричної моделі. Для контактних задач бажано також розробити функцію для автоматичного пошуку контактних пар. Для моделювання зусилля згвинчування в

CalculiX можна використати ключове слово *PRE-TENSION SECTION або застосувати модель теплового розширення (звуження) заданої ділянки деталі РЗ.

Повністю сформований файл model.inp передається розв'язувачу CCX, окремий процес якого може бути створений засобами Python. Після завершення розв'язування задачі (завершення процесу CCX) необхідно отримати результати з текстового файлу результатів model.frd шляхом його синтаксичного аналізу.

Отже загальний алгоритм програми передбачає виконання таких кроків: побудова геометричної моделі РЗ (model.brep); генерація сітки скінченних елементів (model.inp) Gmsh в окремому процесі; синтаксичний аналіз файлу model.inp і читання інформації про вузли, елементи і ребра; пошук назв ребер у файлі model.inp, на яких задаються граничні умови і навантаження; пошук пар ребер, що контактують; формування повного файлу model.inp з описом матеріалів, контактних пар, граничних умов і навантажень; розв'язування задачі CCX в окремому процесі; читання результатів з файлу model.frd. Оптимізаційні розрахунки можна легко організувати за допомогою циклу (або вкладених циклів), тілом якого є ці процедури.

Реалізує цей алгоритм програма ThreadsOCC (додаток I), яка створена мовою Python, є частиною ІС і призначена для моделювання РЗ МСЕ. За допомогою програми можна створювати параметричні геометричні та осесиметричні СЕМ РЗ. Для створення геометричних моделей використовується метод граничного подання, що реалізується бібліотекою pythonOCC 0.18.1. Для створення СЕМ використовується вільна програма CalculiX 2.15. Програма ThreadsOCC складається з модулів gost13877_96params.py, myBaseGeom.py, main.py, ccx_in.py, ccx_out.py. Створення моделей і симуляція відбувається автоматично за заданими параметрами моделі, тому програма може бути легко використана для автоматизованої оптимізації параметрів РЗ. Модуль gost13877_96params.py (лістинг I.1) описує основні та допоміжні параметри РЗ ШН. Основні параметри – це номінальні значення розмірів з відхиленнями відповідно стандарту ГОСТ 13877-96. Допоміжні параметри призначені для полегшення побудови моделі та залежать від основних (наприклад точка для

ідентифікації ребер геометричної моделі). Модуль установлює дійсні значення параметрів за допомогою функції `setModelParams`. За замовчуванням установлюються такі значення в межах допуску, які створюють найменш міцну конструкцію РЗ. В цьому модулі можна сворити і інші параметри моделі такі як параметри матеріалу та навантаження.

Модуль `main.py` (лістинг I.5) – це головний модуль програми. В ньому спочатку будується геометрична модель, потім СЕМ, відбувається симуляція і отримання результатів. Для спрощення побудови плоскої геометричної моделі розроблені такі функції як `poly` (створює полігон зі скругленнями або фасками на вершинах), `translate`, `rotate` (паралельне перенесення і обертання фігур), `cut_array` (створює масив вирізів на грані) та інші. Плоскі геометричні моделі ніпеля, муфти і РЗ створюється за допомогою функцій `face1`, `face2` і `mkCompound`. Модель зберігається у форматі `brep`. Модуль `ssx_inp.py` (лістинг I.3) містить функції для створення сітки FEM-моделі та `input`-файлу для `CalculiX`. Для створення сітки за `brep`-файлом використовується `Gmsh 4.4.0`. В `input`-файлі потрібно вказати умови контакту, зовнішнє навантаження та граничні умови на заданих лініях сітки. Щоб знайти ці лінії у файлі сітки, згенерованому `Gmsh`, виконується його синтаксичний аналіз (функція `ssx_inp.parse`), отримується множина вузлів, ліній і елементів (`nodes`, `lines`, `elements`). Потрібно також знайти відповідність між ребрами геометричної моделі та лініями сітки за допомогою таких функцій як `findEdge`, `findContEdges`, `ssx_inp.findLine` та ін. Після цього формується кінцевий варіант `input`-файлу і виконується симуляція в `CalculiX`. Результати симуляції записуються у файл `frd`. Модуль `ssx_out.py` (лістинг I.4) містить функції для синтаксичного аналізу файлу `frd`, отримання у заданому вузлі компонент напруження, обчислення головних напружень, σ_m , D [205].

Значення параметрів можуть бути передані в програму `main.py` з командного рядка, а результати виведені в її стандартний потік виведення. Нариклад для обчислення мінімального значення D в ніпелі з параметрами

`r3n=2.5 d_n=13.12` введіть в консолі `"python main.py r3n=2.5 d_n=13.12"` та отримайте результат: `FOS= -13.6997873264`.

4.3 Геометричні моделі різьбових з'єднань з відхиленнями на основі FreeCAD

Одним з основних параметрів, який впливає на надійність РЗ, є їхня геометрична точність. Геометрична точність залежить від похибок точіння різьби (похибки верстату, динамічної похибки, похибки від розмірного спрацювання інструмента та інших). Виявляти залежності показників надійності від цих факторів дозволить побудова параметричної геометричної 3D моделі РЗ з різними відхиленнями геометричних параметрів від їхніх номінальних значень. Ця достатньо складна геометрична задача може бути розв'язана за допомогою сучасних геометричних ядер, зокрема Open CASCADE Technology, на якому основана FreeCAD. Розроблено Python-програму (додаток Й) з використанням FreeCAD API для симуляції різьб з різними відхиленнями геометричних параметрів, які виникають під час точіння [8, 13]. Алгоритм оснований на наближенні складної траєкторії переміщення інструмента сплайнами та використанні булевої операції «cut» над тілами заготовки та інструмента. Для симуляції різних похибок обробки в програмі є можливість зміни значень геометричних параметрів заготовки і різця та параметрів траєкторії переміщення різця відносно заготовки. Перевірка придатності різьби з відхиленнями реалізована шляхом булевих операцій над тілами гранично допустимої та реальної деталі. Моделі РЗ, побудовані за допомогою цієї програми, можуть бути використані для симуляції напружено-деформованого стану МСЕ, визначення залежності показників надійності від цих відхилень та обґрунтування значень допусків різьб. За допомогою розробленої системи, основаної на Python та FreeCAD, були створені та проаналізовані осесиметричні СЕМ РЗ ШН з відхиленнями кута профілю та кроку різьби. Повністю код програми наведено в лістингу Й.1.

Спочатку програма імпортує необхідні модулі (рядки 6-10). Функції модуля FreeCAD дозволяють працювати з документами FreeCAD. Модуль FreeCADGui містить усе, що стосується графічного інтерфейсу користувача та 3D видів [318]. Модуль Part – це прямий зв'язок між FreeCAD та ядром Open CASCADE. За допомогою цього модуля ви можете створити геометричні моделі шляхом граничного подання BREP (boundary representation) топологічних форм (вершина, ребро, цикл, грань, поверхня, тверде тіло, сукупність твердих тіл, сукупність будь-яких форм) [318]. Модуль numpy забезпечує функції для швидких операцій над багатовимірними масивами. Модуль dims (dimsNKT) містить об'єкт `d`, атрибути якого є різними параметрами P3 (лістинг Й.2). Якщо не використовується внутрішній інтерпретатор Python, то слід вказати шлях до модулів FreeCAD (рядки 2-5).

Можливо використовувати параметричні ескізи FreeCAD для спрощення розробки моделей. Функція `rebuildSketch(dim, sk)` може бути використана для оновлення параметричного ескізу (рядок 39). Параметр `dim` – це словник з парами (ідентифікатор обмеження ескізу, значення розміру), а параметр `sk` – назва ескізу. Функція викликає метод `setDatum(k, v)` для кожної пари в `dim`, щоб встановити значення `v` обмеження з ID `k`, перебудовує документ і повертає грань. Функція `revolve` (рядок 50) створює тверде тіло, обертаючи грань `f` навколо осі `Y`.

Щоб створити ідеальну гвинтову лінію, використовуйте функцію `makeLongHelix` з модуля Part. Щоб уникнути помилок під час створення довгих різьб, не використовуйте функцію `makeHelix`. Потрібно повернути гвинтову лінію так, щоб її вісь збігалася з віссю `Y`. Ці операції виконуються за допомогою функції `helix(r, h, p, fi)` (рядок 54), де `r` – радіус, `h` – висота, `p` – крок, `fi` – кут конуса (рад).

Різьба на ніпелі (або на муфті) створюється функцією `makeThread(f, h, s)`, де `f` – грань інструмента, `h` – гвинтова лінія, `s` – тверде тіло заготовки (рядок 61).

За допомогою цих функцій можна створити номінальні тверді тіла ніпеля або муфти (рядки 90-98).

Розглянемо побудову моделі різьб з різними відхиленнями. Для імітації різних похибок обробки в програмі можливо змінити значення геометричних параметрів заготовки та інструмента, а також параметри траєкторії інструмента відносно заготовки. Траєкторія відносного руху заготовки та інструмента являє собою циліндричну або конічну спіральну криву з параметричними рівняннями (4.1), де t – параметр (рад), r – початковий радіус, φ – кут конуса (рад), p – крок, $a = \tan(\varphi)p/(2\pi)$, $b = p/(2\pi)$.

$$\begin{aligned}x &= (r+at)\cos(-t), \\z &= (r+at)\sin(-t), \\y &= bt.\end{aligned}\tag{4.1}$$

Функція `helixPoints(r, h, p, fi, n)` повертає масив точок гвинтової лінії (рядок 16). Параметр r – менший радіус, h – висота, p – крок, fi – кут конуса, n – кількість точок за один виток. Тут t , x , y , z – масиви numpy.

Можна моделювати різні систематичні або випадкові похибки обробки, використовуючи рівняння спіралі з відхиленнями (4.2), де x_0, y_0, z_0 – координати точок номінальної гвинтової лінії; $A_x, A_y, A_z, \Delta x, \Delta y, \Delta z$ – константи або випадкові величини. Для номінальної різьби $A_x = A_y = A_z = 1$, $\Delta x = \Delta y = \Delta z = 0$. Наприклад, константа A_x може бути використана для імітації радіального биття шпинделя, а змінна Δy – для імітації похибки кроку. Щоб імітувати випадкові похибки обробки, змінні $A_x, A_y, A_z, \Delta x, \Delta y, \Delta z$ повинні бути випадковими змінними з певними розподілами ймовірностей.

$$\begin{aligned}x &= A_x x_0 + \Delta x, \\z &= A_z z_0 + \Delta z, \\y &= A_y y_0 + \Delta y.\end{aligned}\tag{4.2}$$

Функція `perror(points)` повертає координати точок гвинтової лінії з відхиленнями, де $\Delta x, \Delta y, \Delta z$ – випадкові величини з трикутним розподілом (рядок 29). Параметр `points` – це точки номінальної спіралі. Користувач може модифікувати цю функцію для імітації інших похибок.

Алгоритм створення різьб оснований на наближенні траєкторії руху інструмента за допомогою сплайнів або поліліній. Щоб наблизити гвинтову лінію з відхиленнями полілінією, використовуйте функцію `helix2(points)`, де `points` – це точки лінії з відхиленнями (рядок 66). Можна створити різьбу з відхиленнями різними способами. Наприклад, API-функцію `makePipeShell` можна використовувати для створення гвинтового тіла за допомогою сплайна (або полілінії) та списку профілів інструментів у кожній точці траєкторії. Але найпростіший і універсальний спосіб – використовувати операції з фігурою «лофт» і «виріз» для кожної точки `helix2(points)` (рис. 4.1). Функція `makeThread2(f, h, s)` створює таку різьбу за допомогою функцій `copy`, `rotate`, `translate`, `makeLoft`, `cut`. Параметр `f` – грань інструмента, `h` – цикл гвинтової лінії, `s` – тіло заготовки (рядок 72). Обчислення можуть бути тривалими, якщо точок багато.

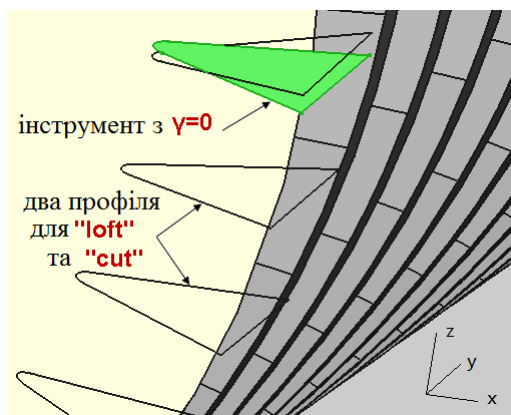
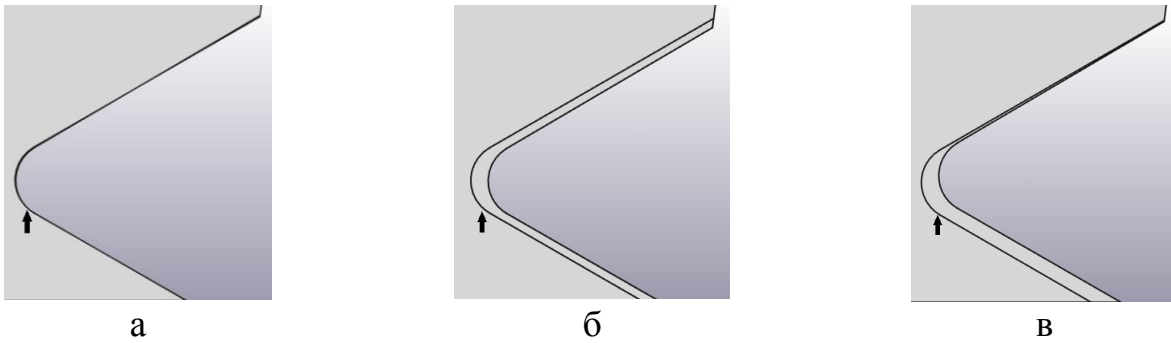


Рисунок 4.1 – Створення різьби з відхиленнями

Можна також імітувати нахил передньої поверхні різця та ненульовий передній кут γ . Для цього потрібно повернути грань на кут γ навколо лінії, яка паралельна осі спіралі та проходить через середину профілю (рис. 4.1). Код (рядки 118-122) створює ніпель з різьбою, яка має відхилення.

Перевірка придатності різьби з відхиленнями може бути реалізована булевими операціями на гранично допустимих та реальних формах. Показано (рядки 125-129) код для обчислення об'єму або площі симетричної різниці (XOR) для номінальних і реальних різьб. На рис. 4.2 показано максимальні відхилення між номінальним ($\gamma=0^\circ$, $A_x=0$, $\Delta y=0$) та реальним профілями, коли є поєднання різних похибок. Номінальний профіль позначений стрілкою, а площа XOR знаходиться між профілями. Відхилення майже не видно на рис. 4.2а. У цьому

випадку, за даними [102], відхилення кута профілю не більше $0,47^\circ$. На рис. 4.3 показано номінальну 3D різьбу ($\Delta x=0$, $\Delta y=0$, $\Delta z=0$) та 3D різьбу з відхиленнями за рівнянням 2, де Δx , Δy , Δz – випадкові величини з трикутним розподілом (з параметрами $a=-0,2$ мм, $c=0$ мм, $b=0,2$ мм). Обчислений об'єм XOR дорівнює $133,79$ мм³.



а) – $\gamma = -10^\circ$, $\Delta y=0$, $A_x=0$, XOR-площа= $0,035$ мм²; б) – $\gamma = -10^\circ$, $\Delta y=0$, $A_x=1,01$ мм, XOR-площа= $0,841$ мм²; в) – $\gamma = -10^\circ$, $\Delta y=0,1$ мм, $A_x=1,01$ мм, XOR-площа= $0,774$ мм²

Рисунок 4.2 – Відхилення між номінальним та реальним профілями у разі поєднання різних похибок

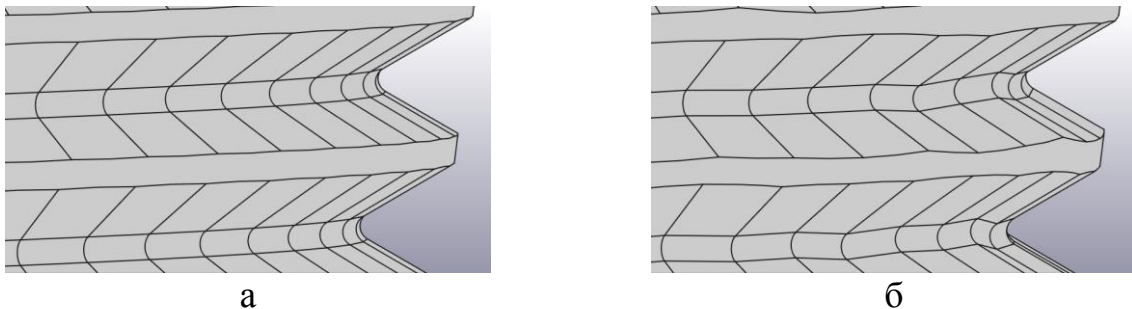


Рисунок 4.3 – Номінальна різьба (а) і різьба з випадковими відхиленнями (б)

Розроблена система може бути використана для побудови 3D та 2D геометричних моделей РЗ з відхиленнями, обґрунтування їхніх допусків і оптимізації геометричних параметрів з додатковим програмним забезпеченням для FEA.

4.4 Опис методики оптимізації різьбових з'єднань

Основними вимогами для здійснення оптимізаційних обчислень є наявність системи FEA Abaqus 6.10 або PythonOCC(FreeCAD)+Gmsh+CalculiX та набору програм (додатки И, I, Й), розроблених автором. Далі описана методика оптимізації РЗ [93].

1 Постановка задачі.

1.1 Постановка оптимізаційної задачі. Вибір керованих змінних (параметрів), їхніх обмежень та критеріїв оптимізації. Параметрами можуть бути геометричні параметри, механічні характеристики матеріалів, зусилля згвинчування, зовнішні навантаження. Оптимізація може бути виконана за критеріями статичної та втомної міцності, герметичності, стійкості до самовідгвинчування. Відповідними числовими критеріями є σ_m , D , контактний тиск p між витками різьби та іншими поверхнями РЗ.

1.2 Визначення виду СЕМ в залежності від наявності в ній несиметричної геометрії та навантажень. Модель може бути тривимірною з гвинтовою різьбою, тривимірною з квазірізьбою, плоскою, осесиметричною.

1.3 Визначення способу моделювання моменту згвинчування в залежності від виду РЗ і його моделі. Згвинчування можна моделювати: моментом згвинчування, зміщенням контактних поверхонь, болтовим навантаженням, температурною деформацією, полем деформації.

1.4 Визначення виду і величини зовнішніх навантажень (статичних, динамічних, температурних). Якщо одним з критеріїв оптимізації є коефіцієнт D , то вибирають два значення зовнішнього навантаження – мінімальне і максимальне навантаження циклу.

1.5 Визначення параметрів вихідної моделі. Наприклад, відповідно ГОСТ на задане РЗ визначаються значення параметрів РЗ потрібного типорозміру (геометричні параметри, механічні характеристики матеріалів, моменти згвинчування тощо).

2 Побудова параметричної СЕМ РЗ.

2.1 Створення бази даних параметрів вихідної моделі. Параметри вихідної моделі кодуються мовою Python. Визначаються допоміжні параметри, необхідні для побудови моделі та отримання результатів.

2.2 Побудова параметричної геометричної моделі РЗ. Вибирають спосіб побудови геометричної моделі. Для можливості простої модифікації геометрії в широких межах рекомендується розбити геометрію деталі РЗ на сукупність елементів простої форми, здатних до легкої модифікації. Засобами Python ці елементи можна описати у вигляді класів. За відсутності необхідності модифікації

геометрії в широких межах кількість і складність форми елементів можна обмежити.

3 Оптимізація параметрів РЗ.

3.1 Створення алгоритму розрахунків та бази даних результатів. Для оптимізації можуть бути використані методи планування експерименту. Це дозволить суттєво зменшити час розрахунків. Зокрема, для побудови поверхонь відклику і знаходження їхніх екстремумів рекомендується побудова центральних композиційних планів. В найпростішому випадку алгоритм розрахунків базується на методі повного перебору. Наприклад для параметрів L (довжина канавки) і Δ (величина згвинчування):

ДЛЯ КОЖНОГО значення L від 15 до 45 мм з кроком 10 мм:

ДЛЯ КОЖНОГО значення Δ від 0 до 0,3 мм з кроком 0,05:

Побудова (перебудова) моделі,
симуляція, збереження результатів

3.2 Обробка результатів. Для обробки результатів можуть бути використані різноманітні статистичні та інші математичні системи, наприклад Excel, SPSS, Statistica, Maple, SciPy. Наприклад, функція Maple `Fit` підганяє модель функції до даних методом найменших квадратів, функція `NLPsolve` знаходить екстремум функції в заданих межах аргументів, функції `plot3d` і `plot` будують графіки функцій двох і однієї змінної відповідно.

3.3 Оптимізація за критерієм статичної міцності. Отримують залежність максимального еквівалентного напруження (або еквівалентного напруження в небезпечній зоні) від параметрів оптимізації. Знаходять мінімальне значення еквівалентного напруження і значення відповідних параметрів моделі.

3.4 Оптимізація за критерієм втомної міцності. Отримують залежність максимального значення D (або значення D в небезпечній зоні) від параметрів оптимізації. Знаходять максимальне значення D і значення відповідних параметрів.

3.5 Оптимізація за критерієм герметичності. Отримують залежність контактного тиску p від параметрів оптимізації. Знаходять максимальне значення контактного тиску і значення відповідних параметрів.

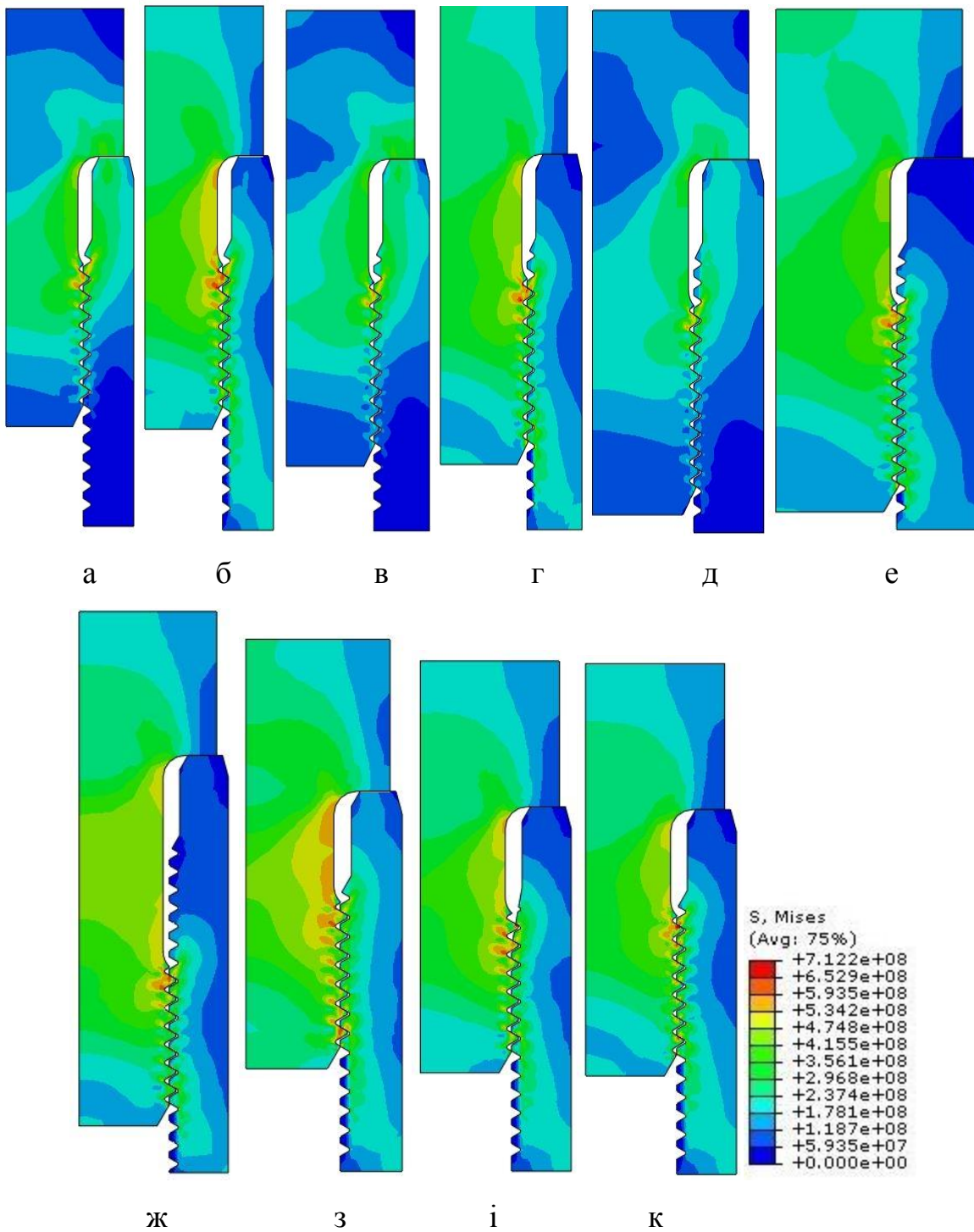
3.6 Оптимізація за кількома критеріями. Найчастіше може бути кілька критеріїв оптимізації, наприклад максимальні значення D і p . В такому випадку отримують залежності цих критеріїв від параметрів оптимізації та наносять на один графік. З графіка знаходять екстремальне значення одного критерію, тоді як інші повинні знаходитись в допустимих межах (відомий метод зміни обмежень). Або застосовують інші методи багатокритеріальної оптимізації.

4.5 Удосконалення різьбового з'єднання насосних штанг за критерієм статичної міцності

На рис. 4.4а-е показані результати симуляції осесиметричних моделей стандартних муфтових РЗ ШН діаметром 19, 22, 25 мм (ГОСТ 13877-96) в Abaqus [93, 196]. Ці результати узгоджуються з даними, отриманими за допомогою інших систем FEA [1, 19, 69, 70, 319].

На рис. 4.4ж-к показані результати симуляції РЗ з подовженою удвічі зарізьбовою канавкою, збільшеним на 0,005 мм кроком різьби муфти, зі зменшеним утричі кутом зрізу перших витків різьби муфти і зі збільшеним на 5° кутом профілю різьби ніпеля [93, 196]. Помітно, що збільшення кроку різьби муфти не зменшує напруження у перших впадинах різьби ніпеля, але вирівнює його вздовж витків (рис. 4.4з). У РЗ з подовженою зарізьбовою канавкою дещо зменшуються напруження, навіть після збільшення величини згвинчування Δ до 0,134 мм (рис. 4.4ж) [20]. Суттєве зменшення напружень у перших впадинах різьби ніпеля спостерігається у разі корекції перших витків різьби муфти (рис. 4.4і) і збільшенні кута профілю різьби ніпеля (рис. 4.4к).

Проаналізуємо вплив на розподіл еквівалентних напружень у впадинах різьби ніпеля таких параметрів РЗ: зусилля згвинчування, довжини зарізьбової канавки, кроку різьби муфти, кута зрізу перших витків різьби муфти, кута профілю різьби ніпеля, радіуса впадин різьби ніпеля, зовнішнього діаметра різьби ніпеля, внутрішнього діаметра різьби муфти, радіуса скруглень зарізьбової канавки [314, 315, 320]. Відомі подібні дослідження для РЗ іншого типу і призначення [71].



верхній ряд: а, б) – ШН19; в, г) – ШН22; д, е) – ШН25;

а, в, д) – $\sigma_p=0$ МПа; б, г, е) – $\sigma_p=276$ МПа;

нижній ряд: ж-к) – ШН19 мм для $\sigma_p=276$ МПа: ж) – зарізьбова канавка довжиною

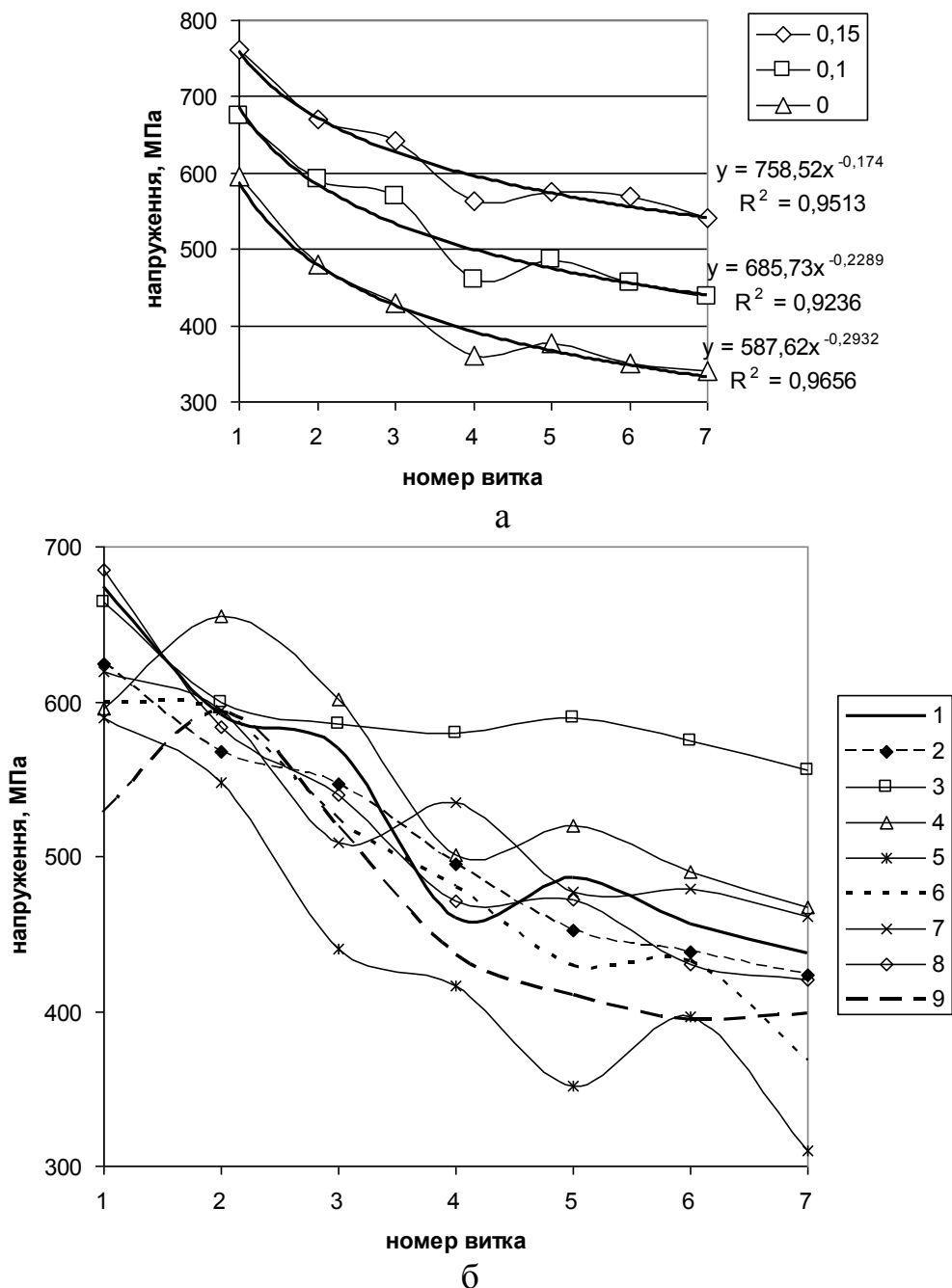
30мм ($\Delta=0,134$ мм); з) – крок різьби муфти 2,545мм; і) – кут зрізу перших витків

різьби муфти 10° ; к) – кут профілю різьби ніпеля 65°

Рисунок 4.4 – Розподіл напружень σ_m (Па) в РЗ ШН для $\Delta=0,1$ мм

На рис. 4.5а зображено графік розподілу еквівалентних напружень σ_m у впадинах різьби ніпеля від величини згвинчування Δ [320]. Як бачимо, навантаження зростає від останніх до перших витків за законом гіперболічного косинуса. Для РЗ такого типу це підтверджується і іншими джерелами [71]. Водночас помітно, що збільшення моменту згвинчування призводить до більш рівномірного розподілу навантаження.

У разі збільшення довжини зарізьбової канавки удвічі, збільшується податливість ніпеля. Враховуючи це, для забезпечення зусилля згвинчування, яке відповідає напруженню розтягу в ніпелі $0,6\sigma_m$, потрібно збільшити величину Δ до 0,134 мм [19, 20]. Проте, навіть після збільшення величини згвинчування, у перших трьох впадинах різьби ніпеля спостерігаються менші напруження, ніж у стандартного РЗ (крива 2 на рис. 4.5б). Збільшення кроку різьби муфти на 0,005мм призводить до суттєвого вирівнювання напружень у витках різьби ніпеля, але напруження в перших витках майже не зменшуються (крива 3 на рис. 4.5б). Напруження у зарізьбовій канавці зростають. Зменшення кута зрізу (корекції) перших витків різьби муфти утричі (до 10°) дає змогу розвантажити лише перший виток ніпеля (крива 4 на рис. 4.5б). Водночас зменшується також напруження в зарізьбовій канавці. Збільшення кута профілю різьби ніпеля на 5° суттєво зменшує напруження у впадинах його різьби (крива 5 на рис. 4.5б). Це пояснюється зміною області контакту витків: більше навантажується основа витка різьби ніпеля [321]. Зменшується також напруження в зарізьбовій канавці. Збільшення радіуса впадин різьби ніпеля до максимального допустимого (0,36мм) зменшує напруження в усіх впадинах ніпеля, але збільшує напруження в зарізьбовій канавці, оскільки зростає жорсткість різьбової частини ніпеля (крива 6 на рис. 4.5б). У РЗ з максимальним допустимим зовнішнім діаметром різьби ніпеля (крива 7 на рис. 4.5б) напруження у першій впадині менші, ніж у РЗ з мінімальним допустимим внутрішнім діаметром різьби муфти (крива 8 на рис. 4.5б), тому допуск на зовнішній діаметр різьби ніпеля повинен бути якомога меншим.

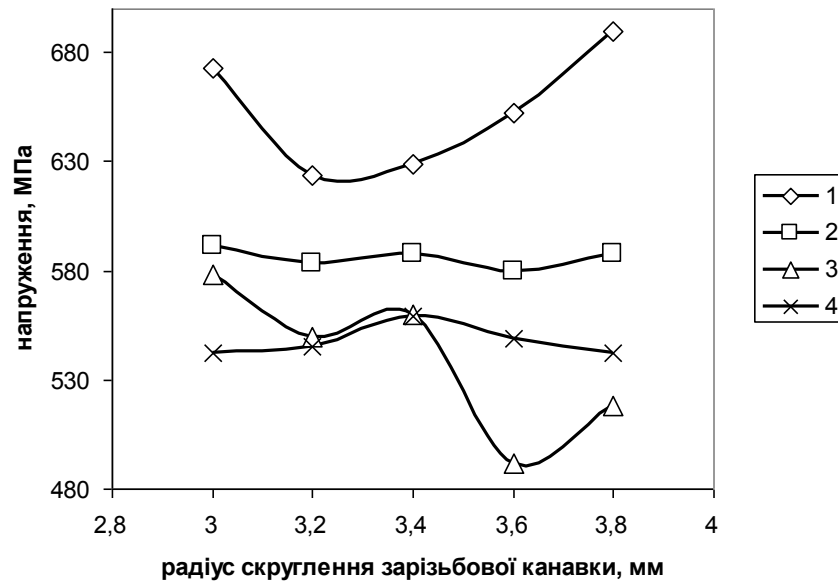


а) – $\Delta=0,15$ мм (\diamond), $\Delta=0,1$ мм (\square), $\Delta=0$ мм (Δ); б) – $\Delta=0,1$ мм:

1 – найбільш небезпечна конфігурація стандартного РЗ; 2 – довжина зарізьбової канавки 30 мм; 3 – крок різьби муфти 2,545 мм; 4 – кут зрізу перших витків різьби муфти 10° ; 5 – кут профілю різьби ніпеля 65° ; 6 – максимальний допустимий радіус впадин різьби ніпеля; 7 – максимальний допустимий зовнішній діаметр різьби ніпеля; 8 – мінімальний допустимий внутрішній діаметр різьби муфти; 9 – комплексна зміна параметрів

Рисунок 4.5 – Розподіл σ_m у впадинах різьби ніпеля РЗ ШН19 для $\sigma_p=276$ МПа

Зміна радіусів скруглення зарізьбової канавки може впливати на напруження як у самій зарізьбовій канавці, так і у перших впадинах різьби ніпеля (рис. 4.6). З рис. видно, що оптимальна величина радіуса лежить в межах 3,2...3,6 мм [320].



1, 2 – у першій (1) і другій (2) впадинах різьби ніпеля;

3, 4 – у першому (3) і другому (4) радіусі скруглення зарізьбової канавки

Рисунок 4.6 – Залежність σ_m від величини радіусів скруглення зарізьбової канавки

Проаналізуємо РЗ з комплексною зміною параметрів (кут зрізу перших витків різьби муфти – 10° , кут профілю різьби ніпеля – 65° , радіус заокруглення зарізьбової канавки ніпеля – 3,2 мм, радіус впадин різьби ніпеля – 0,36 мм, максимальний зовнішній діаметр різьби ніпеля – 26,952 мм, мінімальний внутрішній діаметр різьби муфти – 24,25 мм). З рис. 4.5б (крива 9) видно, що одночасна комплексна зміна параметрів не завжди підсумовує позитивні ефекти

від окремих змін, проте спостерігається значне зменшення напружень порівняно з початковим варіантом (рис. 4.7) [314, 315].

Відомо також, що величина нерівномірності напружень в РЗ залежать від матеріалів його деталей [71]. ГОСТ 13877-96 рекомендує виготовляти ШН зі сталей 40, 20Н2М, 30ХМА, 15Н3МА, 15Х2НМФ, 15Х2ГМФ, 14Х3ГМЮ, а муфти – 20Н2М, 20ХН2М (для ШН зі сталі 15Н3МА), 40, 45 (для ШН з усіх інших сталей). За подібністю механічних характеристик ці сталі можна поділити на такі групи: перша (40, 45), друга (20Н2М, 20ХН2М, 30ХМА), третя (15Н3МА), четверта (15Х2НМФ, 15Х2ГМФ, 14Х3ГМЮ). Автором поставлена задача виявити залежність напружень в небезпечних зонах РЗ від характеристик стандартних матеріалів деталей [12, 93]. Досліджувалось стандартне муфтове РЗ ШН діаметром 19мм. Оптимальний момент згвинчування повинен утворювати напруження в ніпелі $0,5...0,7\sigma_T$ [71]. З цих міркувань підбирали величину згвинчування $\Delta=0,1$ мм. До РЗ прикладалось зовнішнє навантаження розтягу, що відповідає $\sigma_p=276$ МПа. Матеріал деталей РЗ – сталь з такими механічними характеристиками: $E=2,1 \cdot 10^{11}$ Па, $\nu=0,28$. Характеристики пластичності задаються в Abaqus® у вигляді пластичної ділянки істинної діаграми деформування (ε - σ).

Результати досліджень наведено в табл. 4.1 та на рис. 4.8. Помітно, що зі збільшенням міцності сталі ніпеля напруження в небезпечних зонах РЗ зростають. Підвищується нерівномірність розподілу навантажень по витках різьби ніпеля, особливо у разі малих зовнішніх навантажень. Підвищення міцності сталі муфти практично не впливає на напруження в небезпечних зонах ніпеля, проте підвищуються напруження в останній робочій западині муфти. Отримані результати мають орієнтовний характер. Для їх уточнення необхідно в моделях для кожної комбінації матеріалів ніпеля і муфти створювати оптимальний момент згвинчування шляхом підбору величини Δ [322].

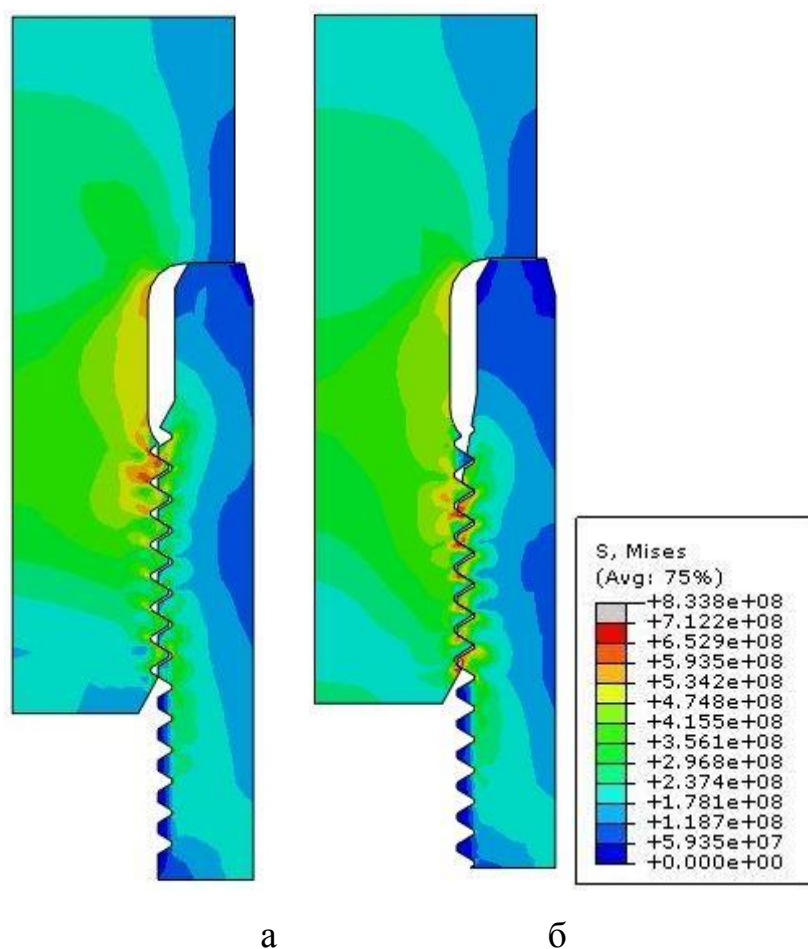
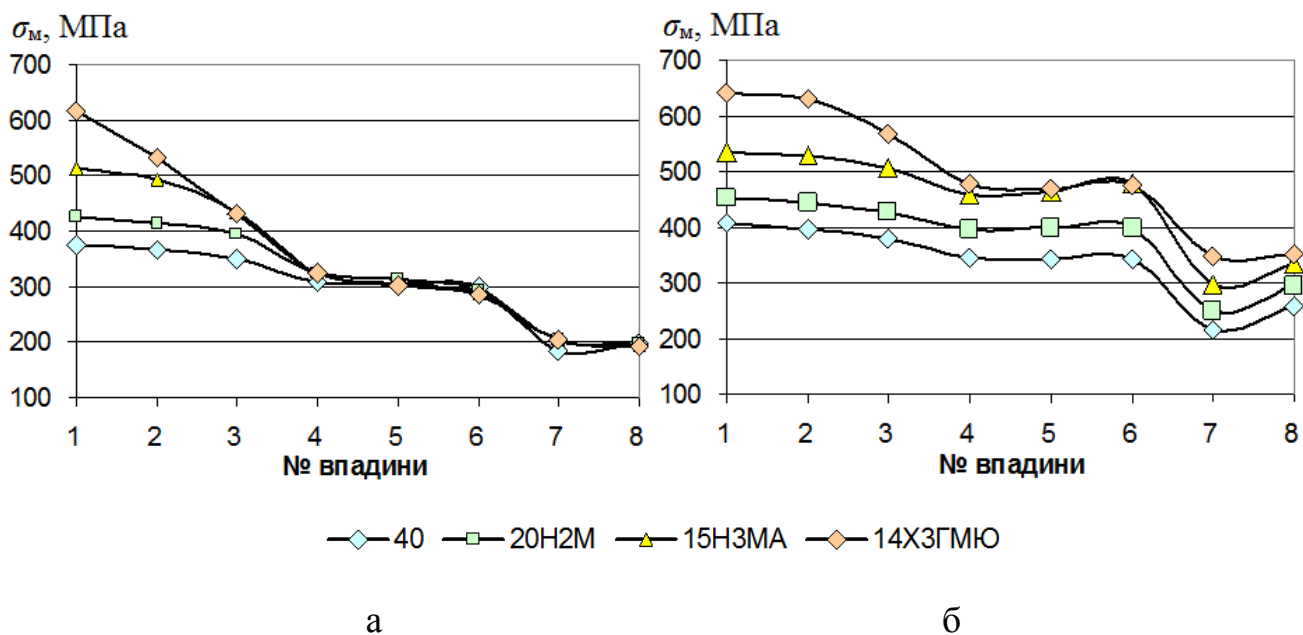


Рисунок 4.7 – Розподіл σ_m (Па) у стандартному (а) і покращеному (б) РЗ 3Н19



а) – без зовнішнього навантаження; б) – зовнішнє навантаження ($\sigma_p=276$ МПа)

Рисунок 4.8 – Залежність σ_m у впадинах різьби ніпеля від матеріалу ніпеля

Таблиця 4.1 – Залежність σ_m (МПа) в місцях концентрації напружень муфтового РЗ ШН діаметром 19 мм (ГОСТ 13877-96) від матеріалів деталей РЗ

Сталь ніпеля	40	20Н2М	15Н3МА	14Х3ГМЮ
Без зовнішнього навантаження, $\Delta=0,1$ мм Для сталей муфти 40, 20Н2М, 15Н3МА*, 14Х3ГМЮ* (в середньому)				
Зарізьбова канавка ніпеля	339	391	446	463
Впадина різьби ніпеля №	1	375	424	512
	2	367	413	492
	3	349	393	433
	4	307	324	325
	5	305	313	305
	6	298	293	287
	7	184	203	205
	8	198	199	194
Остання робоча впадина різьби муфти	196			
Зовнішнє навантаження, яке відповідає $\sigma_p=276$ МПа Для сталей муфти 40, 20Н2М, 15Н3МА*, 14Х3ГМЮ* (в середньому)				
Зарізьбова канавка ніпеля	366	416	503	585
Впадина різьби ніпеля №	1	408	453	533
	2	396	445	529
	3	378	428	507
	4	346	395	459
	5	343	398	462
	6	343	399	477
	7	216	249	297
	8	258	295	334
Сталь муфти	Остання робоча впадина різьби муфти			
40	341	350	360	362
20Н2М	383	396	404	406
15Н3МА*	426	445	463	467
14Х3ГМЮ*	460	489	510	515

Примітка :* – не рекомендується ГОСТ 13877-96

4.6 Удосконалення різьбового з'єднання насосних штанг за критеріями втомної міцності

Для обґрунтування доцільності збільшення довжини розвантажувальної (зарізьбової) канавки необхідно отримати залежності коефіцієнта запасу D або циклічної довговічності N в найбільш небезпечній зоні (поверхня першої западини різьби ніпеля) від довжини канавки L та величини згвинчування Δ . Додатково для максимального навантаження циклу необхідно обчислити значення контактної тиску на упорному бурті P_c , та значення еквівалентних напружень в першій западині різьби ніпеля σ_{m1} і в центрі зарізьбової канавки σ_{m2} (на їхній поверхні). Необхідність розрахунку контактної тиску P_c зумовлена тим, що зі збільшенням довжини розвантажувальної канавки жорсткість ніпеля зменшується, тому зменшується і величина Δ , яка використовується для контролю величини затягування РЗ. Для отримання цих залежностей в системі Abaqus® 6.11 автором розроблено СЕМ муфтового РЗ ШН (ГОСТ 13877-96) та макрос мовою Python для автоматизації процесу зміни параметрів, перебудови моделі, розрахунку та отримання результатів [198, 313, 316] (лістинги И.1, И.2, И.5).

Розглядали модель РЗ ШН діаметром 19 мм. Матеріал деталей РЗ – сталь 40 (ГОСТ 13877-96). Моделюється пластичність матеріалів та тертя між витками різьби. РЗ сприймає зовнішнє навантаження розтягу, яке створює в тілі ШН напруження σ_p . Створювали два кроки навантаження, які відповідають мінімальному і максимальному навантаженню циклу. Довжину зарізьбової канавки L змінювали від стандартної довжини 15 мм до 45 мм з кроком 10 мм. Величину згвинчування Δ змінювали від 0 до 0,3 мм з кроком 0,05 мм. Алгоритм розрахунків для здійснення оптимізації реалізується у модулі main.py (лістинг И.5).

Значення коефіцієнта запасу D обчислювали за критерієм Сайнса (1.8) [205]. У формулі (1.8) σ_a та σ_m обчислювали за (1.9). Додатково обчислювали значення коефіцієнта запасу D за (1.7), де σ_a обчислювали за (1.9), а σ_m – за (1.10). Для розрахунку D використовували значення констант $\sigma_{-1}=207$ МПа та $\psi_\sigma=0,33$. Для розрахунку циклічної довговічності N використовували критерій Брауна-Міллера (1.11), який дає найбільш реалістичні значення довговічності для пластичних металів. Для цього застосовували програму fe-safe® 6. Для розрахунку значень N в базі даних fe-safe був вибраний її аналог сталі 40 – сталь SAE1040. Розглядали цикл навантаження $\sigma_p=0\dots170$ МПа. Також з метою підбору величини згвинчування, яка б забезпечила нерозкриття стику РЗ, розглядали цикл $\sigma_p = 0\dots340$ МПа [198, 323].

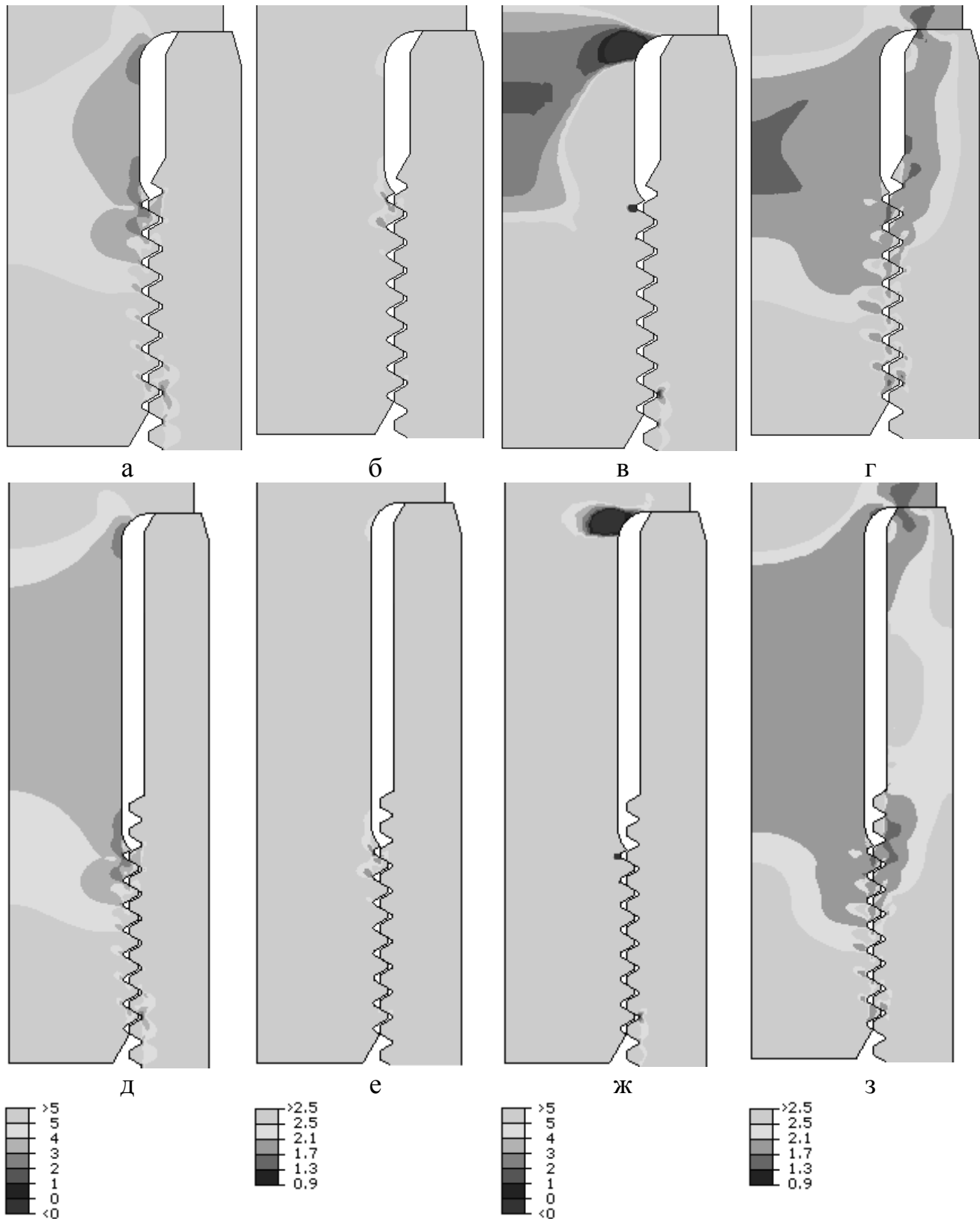
Результати (рис. 4.9) показують, що значення D , обчислені за допомогою (1.8) [93, 323] та (1.7) задовільно ідентифікують небезпечні зони РЗ – зарізьбову канавку, першу западину різьби ніпеля та останню западину різьби муфти. Це також стосується значень N [198]. Але розподіли цих значень (рис. 4.9) та їхні залежності $D(\Delta, L)$ (рис. 4.10) мають різний характер. Формула (1.8) показує підвищення D у разі збільшення L , але дає завищені значення D для малих Δ (рис. 4.10а), що суперечить практичним спостереженням. У незгвинченому з'єднанні понижені значення D займають більшу площу (рис. 4.9а,д). Після згвинчування ця площа зменшується і концентрується біля скруглення канавки і першої западини різьби ніпеля (рис. 4.9в,ж). Значення D в цих зонах зменшуються. Якщо значення D розраховане за (1.7), то спостерігається протилежне явище (рис.4.9б,е,г,з). Залежність мінімального значення D в з'єднанні $D(\Delta, L)$, обчислена за (1.8), має локальні мінімуми в точках (0,1, 15), (0,15, 25), (0,2, 35), (0,25, 45). Ці мінімуми менш помітні, якщо замість мінімального значення D в з'єднанні використовуються значення D на поверхні першої западини ніпеля (рис. 4.10а).

Формула (1.7) (рис. 4.10б) більше відповідає дійсності – занижені або завищені моменти згвинчування можуть призводити до погіршення втомної міцності [71]. Залежність $\lg N(\Delta, L)$ має подібний характер, але не показує зменшення $\lg N$ для великих Δ і має мінімуми в зоні $\Delta=0,1..0,15$ мм (рис. 4.10в).

Із залежностей (рис. 4.11) можна визначити Δ для заданого значення P_c або σ_{m2} [93, 323]. Для прикладу значенню $P_c=190$ МПа відповідають точки залежності $P_c(\Delta, L)$ (мм): (0,1, 15), (0,15, 25), (0,25, 35), (0,3, 45). В цьому випадку σ_{m2} знаходяться в межах 310...330 МПа, σ_{m1} – в межах 380...390 МПа, значення D , отримане за (1.7), є близьким до 1,6, а відповідні значення залежності $\lg N(\Delta, L)$ є наступними: 5,05, 6,14, 7, 7. Проте залежність $\lg N(\Delta, L)$ показує, що значення Δ для $L=45$ мм повинні бути більшими 0,2 мм, а для інших значень L – більшими 0,15 мм (рис. 4.10в). Таким чином для циклу $\sigma_p = 0...170$ МПа допустимо вибрати значення: $\lg N(0,15, 15)=5,46$ ($N=288403$), $\lg N(0,2, 25)=6,52$ ($N=3311311$), $\lg N(0,25, 35)=7$ ($N=10^7$). Так як після збільшення L з 35 мм до 45 мм значення D , $\lg N$, σ_{m1} змінюються не значно, то значення $L=35$ мм може бути прийняте за оптимальне.

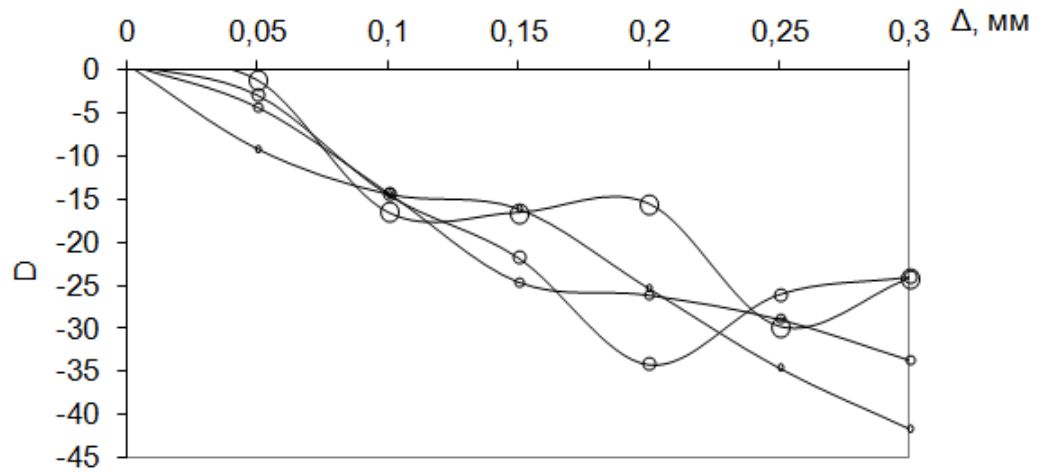
Такі ж обчислення були проведені для підвищених циклічних навантажень [198, 323] (рис. 4.12). Для циклу $\sigma_p = 0...340$ МПа допустимо прийняти: $\lg N(0,15, 15)=4,47$ ($N=29512$), $\lg N(0,2, 25)=5,21$ ($N=162181$), $\lg N(0,2, 35)=7$ ($N=10^7$). В цьому випадку позитивний ефект від збільшення L є вищим.

Отже оптимальні значення Δ для $L=25$ мм збільшують довговічність N мінімум в 5 раз, а для $L=35$ мм – в 35 раз. Розроблена модель може бути використана для аналізу та оптимізації РЗ іншого типорозміру чи конструкції. Результати досліджень можуть бути використані для уточнення моментів згвинчування ШН [324] з врахуванням критерію втомної міцності.

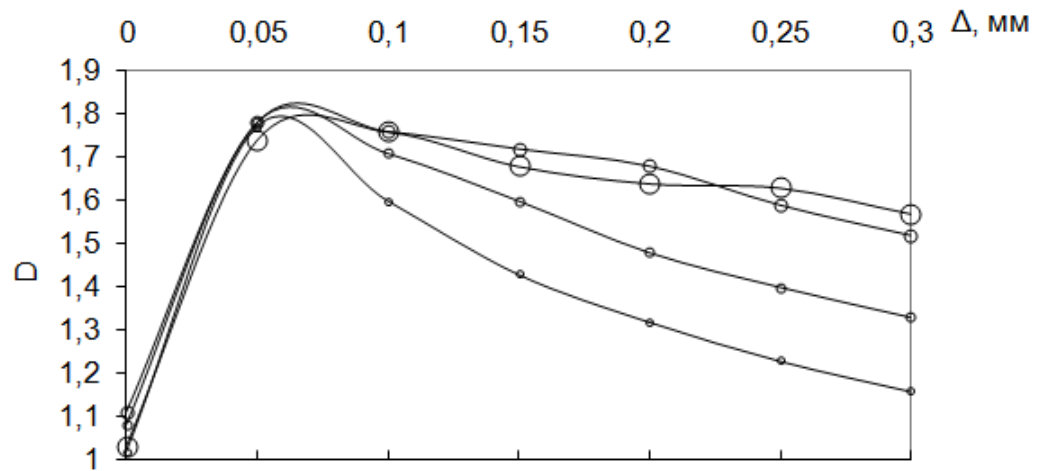


а-г) – $L=15$ мм; д-з) – $L=35$ мм; а, б, д, е) – $\Delta=0$; в, г) – $\Delta=0,1$ мм; ж, з) – $\Delta=0,2$ мм;
 а, в, д, ж) – D обчислено за (1.8); б, г, е, з) – D обчислено за (1.7)

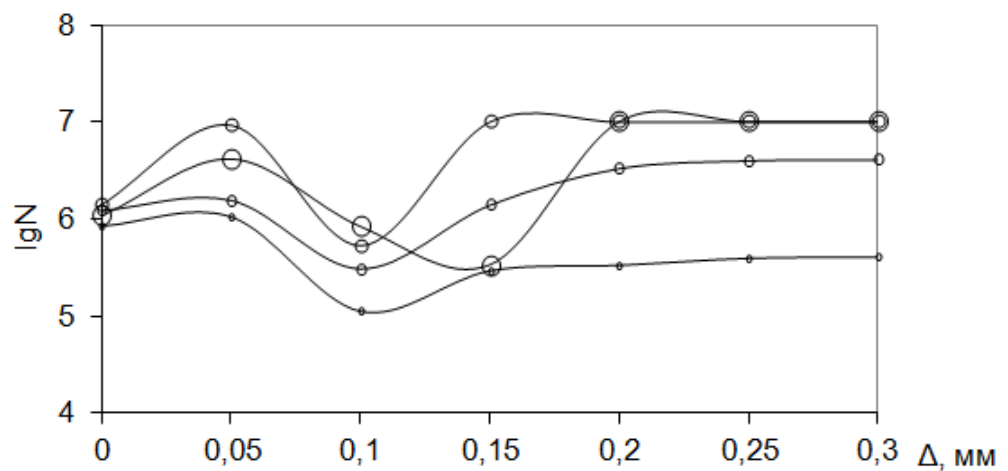
Рисунок 4.9 – Розподіл коефіцієнта запасу D для циклу навантаження $\sigma_p = 0 \dots 170$ МПа



а



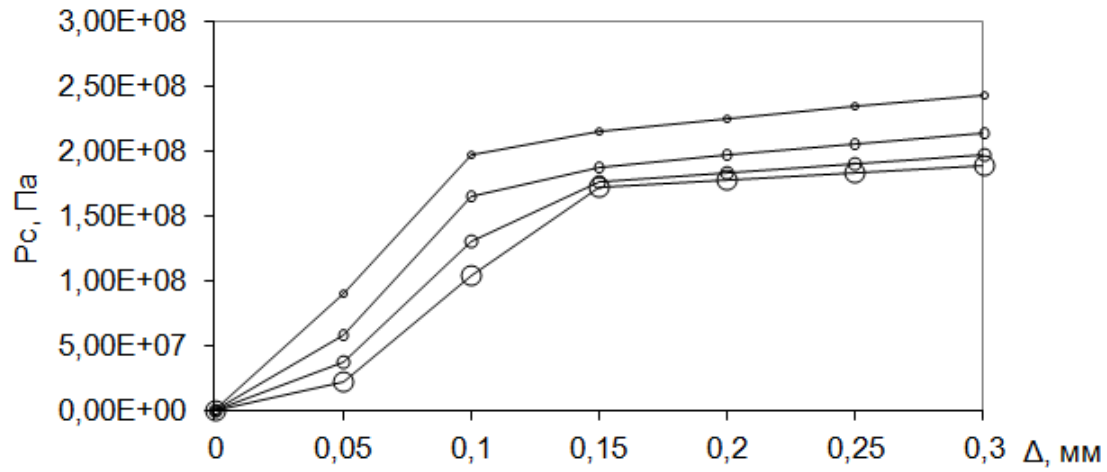
б



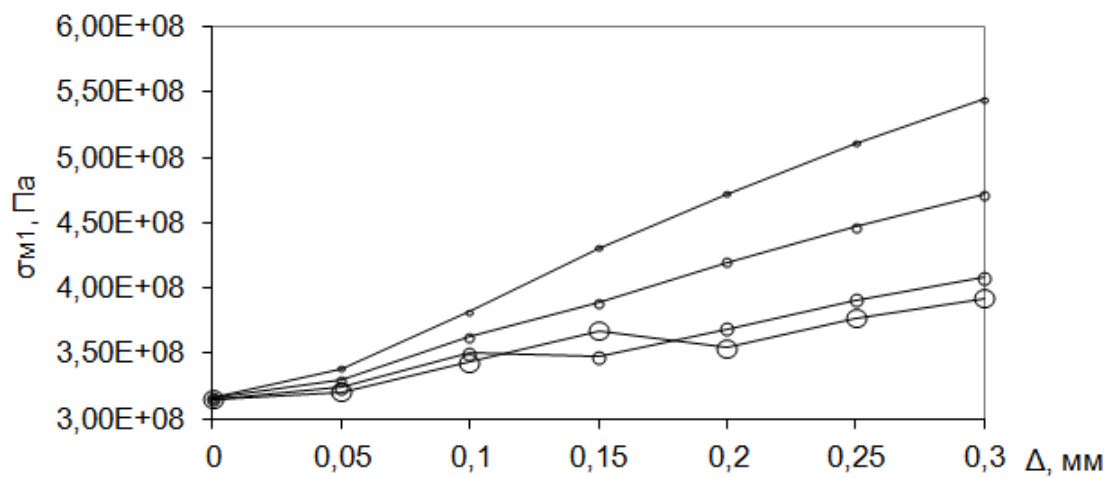
в

○ – $L=15$ мм; ○ – $L=25$ мм; ○ – $L=35$ мм; ○ – $L=45$ мм;
 обчислені за допомогою: а) – (1.8), б) – (1.7), в) – (1.11)

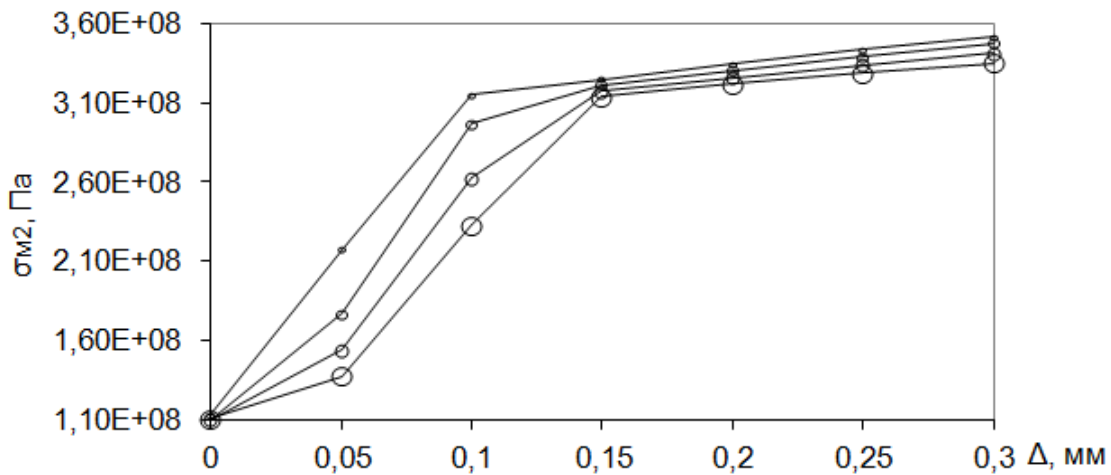
Рисунок 4.10 – Залежності D та $\lg N$ від Δ для циклу $\sigma_p=0\dots 170$ МПа



а



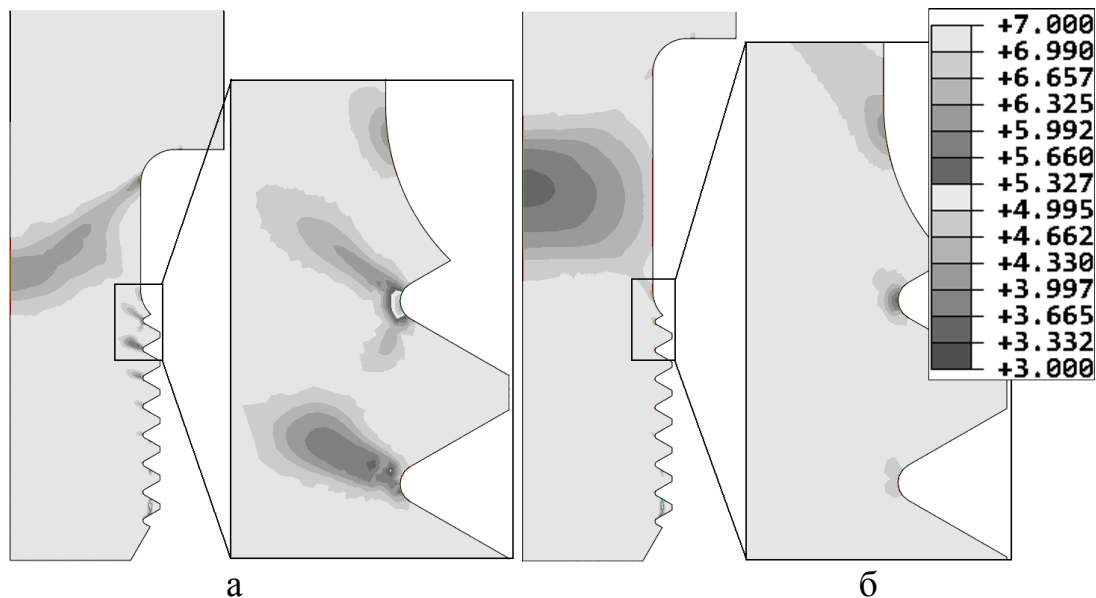
б



в

○ – $L=15$ мм; ○ – $L=25$ мм; ○ – $L=35$ мм; ○ – $L=45$ мм

Рисунок 4.11 – Залежності P_c , σ_{m1} , σ_{m2} від Δ для циклу $\sigma_p=0\dots170$ МПа



а) – $L=15$ мм, $\Delta=0,1$ мм; б) – $L=25$ мм, $\Delta=0,2$ мм

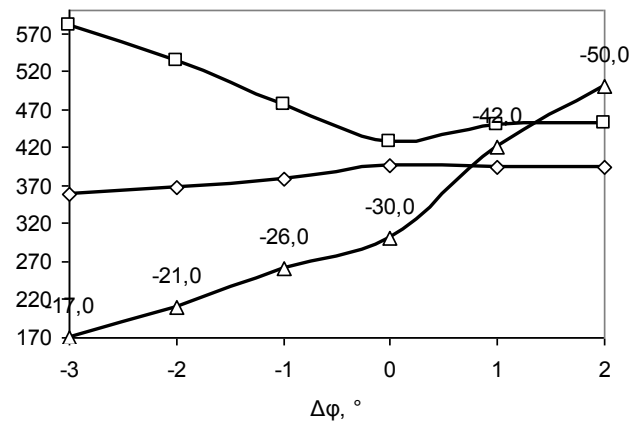
Рисунок 4.12 – Розподіл логарифму циклічної довговічності в ніпелі ШН для циклу $\sigma_p=0\dots340$ МПа

4.7 Скінченно-елементний аналіз різбових з'єднань ШН з відхиленнями

4.7.1 Відхилення кута профілю різби

Моделі РЗ, побудовані за допомогою FreeCAD-API (додаток Й), можуть бути використані для імітації напружено-деформованого стану МСЕ та обґрунтування значень допусків різби. Використовували генератор скінченно-елементної сітки Gmsh 2.7 та FEA-розв'язувач CalculiX 2.12. Розроблено осесиметричні моделі РЗ ШН діаметром 19 мм за ГОСТ 13877 (еквівалент 3/4 in. API Spec 11B). Відповідно до ГОСТ 13877 різба ніпеля повинна бути накатана, а різба муфти може бути накатана або виготовлена іншим методом, наприклад нарізана різцем. Матеріал деталей – сталь 40 з $E=210$ ГПа, $\nu=0,28$, $\sigma_T=314$ МПа, $\sigma_B=559$ МПа. Моделювали пластичність матеріалу і тертя. Розмір елементів сітки 0,2 мм. Створено два кроки зовнішнього осевого навантаження розтягу, яке створює в тілі ШН напруження $\sigma_{pmin} = 0$ МПа і $\sigma_{pmax} = 170$ МПа. Згвинчування моделювали осевою деформацією Δ упорної частини муфти. Для кожного варіанту конструкції підбирали таке значення Δ , яке створює контактний тиск на упорі 312 МПа для σ_{pmax} . Для стандартної конструкції $\Delta = 0,12$ мм. Для орієнтовного обчислення значень D використовували залежність Сайнса (1.8) з $\sigma_{-1} = 207$ МПа.

На рис. 4.13 показані залежності напружень σ_m і коефіцієнта запасу D від відхилення кута профілю різьби муфти $\Delta\varphi = \Delta\varphi_1 = \Delta\varphi_2$ [8]. Індекс 1 позначає ненавантажену сторону профілю, а індекс 2 – навантажену. Помітно, що від'ємні значення $\Delta\varphi$ дещо зменшують значення напружень і збільшують значення D в першій навантаженій западині різьби ніпеля. Але контактні тиски зростають в зоні мінімального діаметра різьби муфти (рис. 4.14а). Додатні значення $\Delta\varphi$ різко зменшують значення D в першій западині різьби ніпеля, але майже не змінюють еквівалентні напруження в ній для σ_{pmax} .



□, ◇ – σ_m ; △ – D ; □ – максимальні значення в РЗ;

◇, △ – значення в першій навантаженій западині різьби ніпеля

Рисунок 4.13 – Залежності σ_m (для σ_{pmax}) і D від $\Delta\varphi$

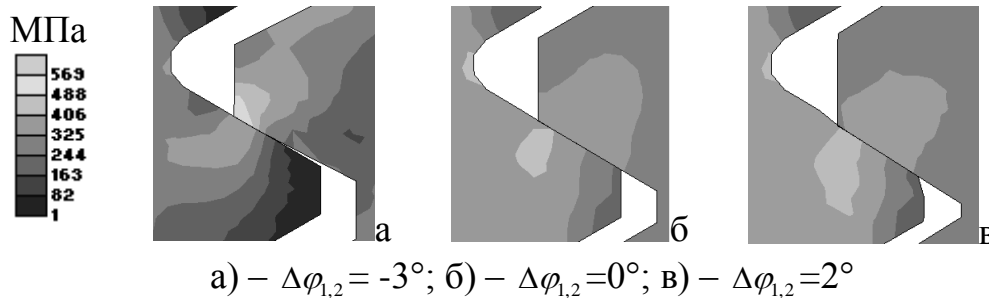
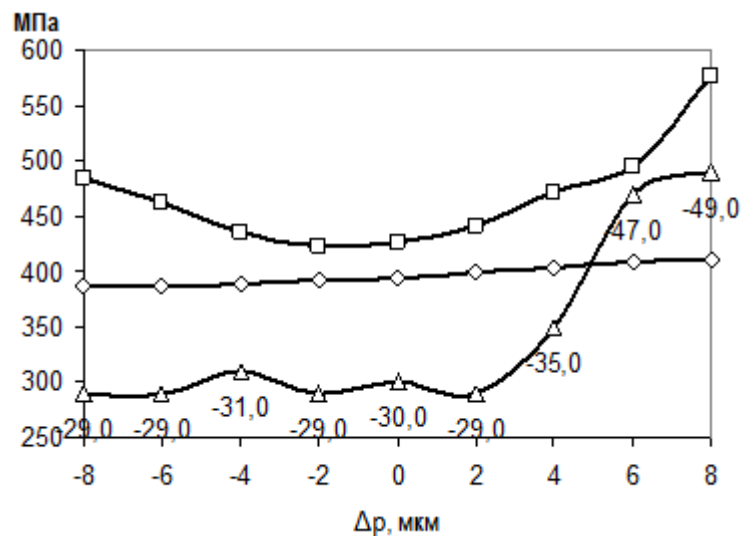


Рисунок 4.14 – Розподіл σ_m в перших витках різьби для σ_{pmax} і різних значень $\Delta\varphi_{1,2}$

4.7.2 Відхилення кроку різьби

На рис. 4.15 показані залежності напружень σ_m і значень D від Δp – величини зменшення кроку різьби муфти на кожному витку (починаючи з першого) [8]. Додатні значення Δp зменшують крок різьби муфти, а від'ємні збільшують. Додатні значення Δp є причиною різкого зменшення значень D внаслідок нерівномірного навантаження на витки (рис. 4.16, 4.17). Напруження,

які показано на рис. 4.17а, можуть з'явитись після багатократного згвинчування РЗ як наслідок пластичної деформації та зношування перших витків. Від'ємні значення Δp вирівнюють навантаження в різьбі, дещо зменшуючи напруження в першій навантаженій западині ніпеля (рис. 4.15-4.17) і дещо зменшують область з від'ємними значеннями D (рис. 4.17). Таким чином потрібно уникати значень $\Delta p > 0$ і намагатись забезпечити $\Delta p < 0$. У випадку нарізання різьби різцем на верстаті з ЧПК це можна досягти шляхом зменшення подачі на Δp за один оберт шпинделя.



□, ◇ – σ_m ; △ – D ; □ – максимальні значення в РЗ;

◇, △ – значення в першій навантаженій западині різьби ніпеля

Рисунок 4.15 – Залежність σ_m (для σ_{rmax}) і D від Δp

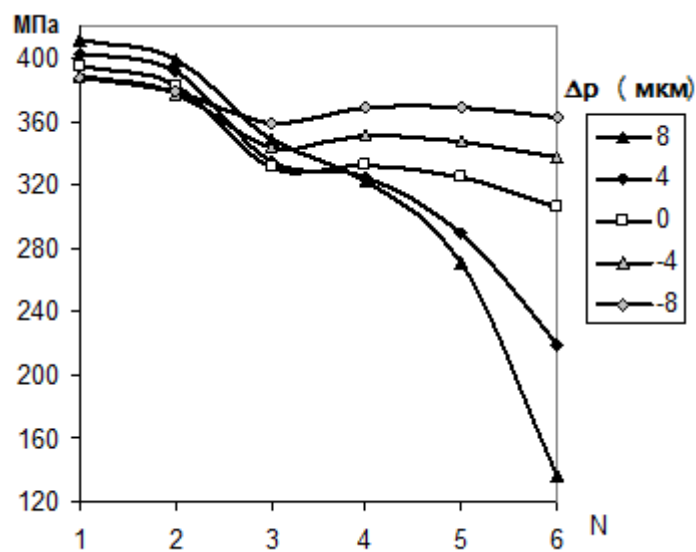
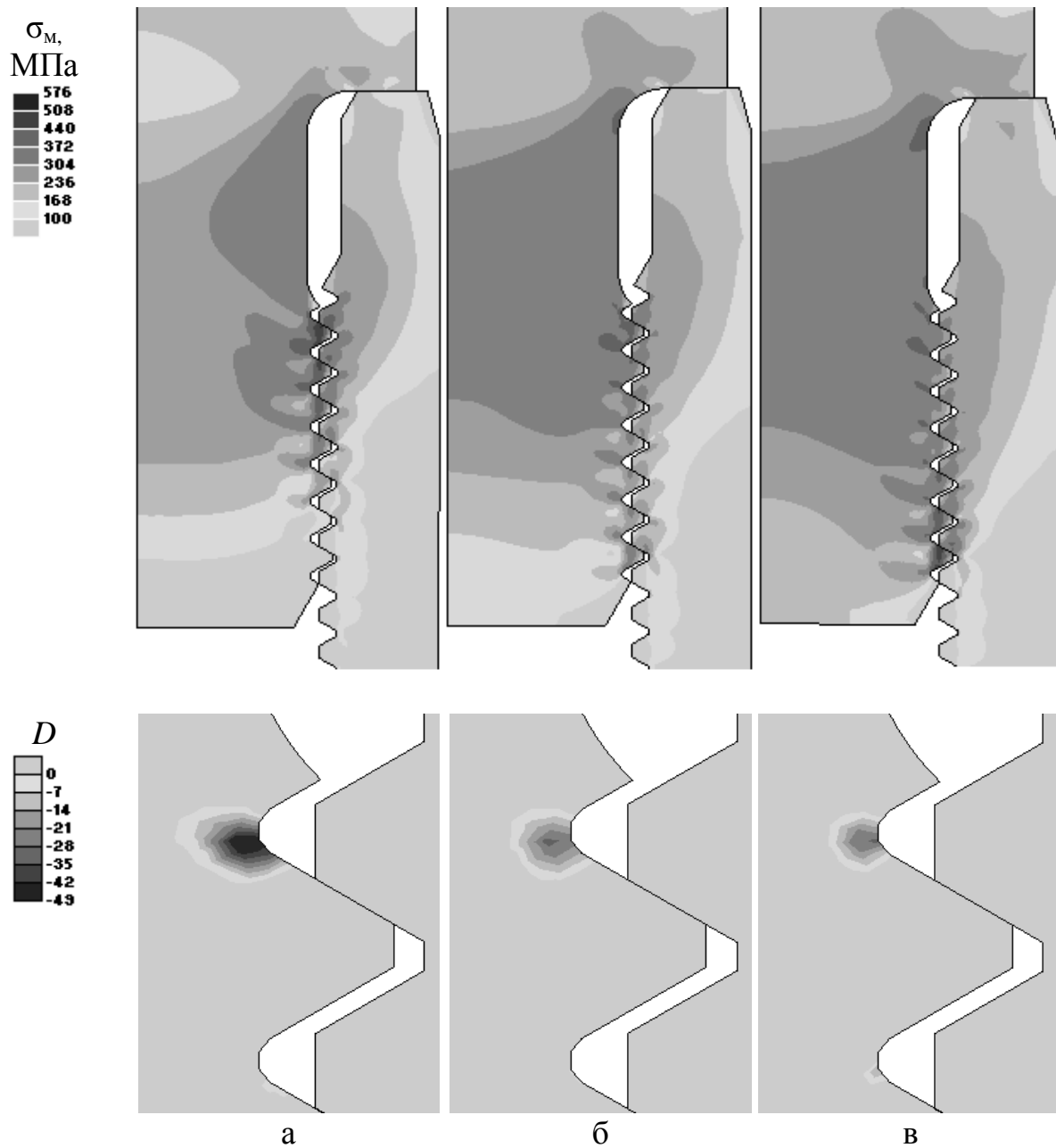


Рисунок 4.16 – Значення напружень σ_m (для σ_{rmax}) в западині N різьби ніпеля для різних значень Δp (мкм)



а) – $\Delta p = +8$ мкм; б) – $\Delta p = 0$ мкм; в) – $\Delta p = -8$ мкм;

верхній ряд – розподіл σ_M (для σ_{pmax});

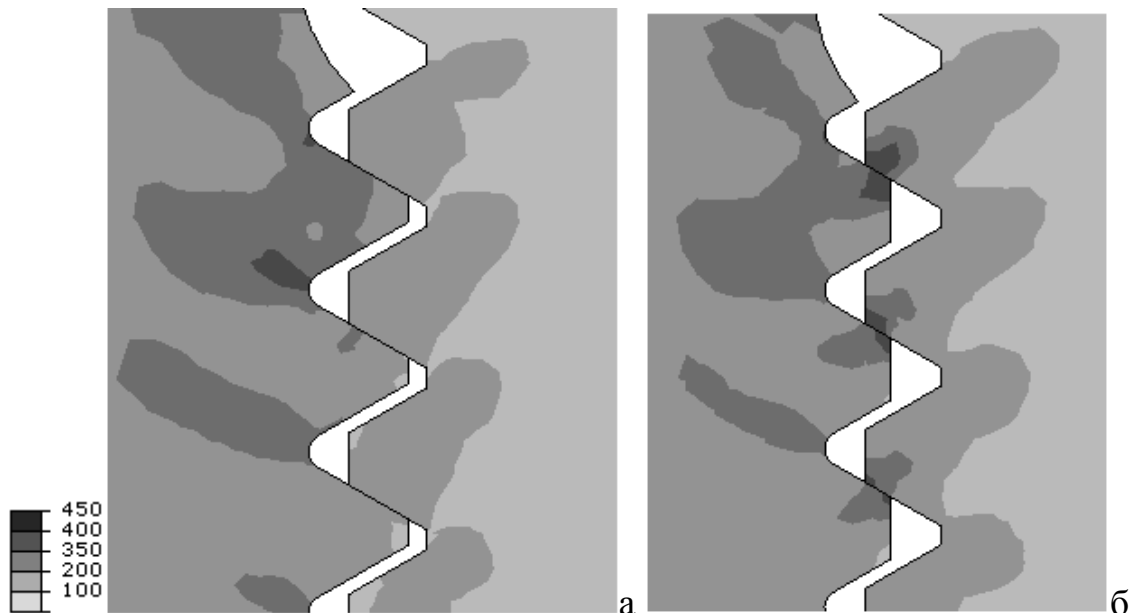
нижній ряд – розподіл D в першому навантаженому витку різьби

Рисунок 4.17 – Результати симуляції РЗ ШН з різними значеннями Δp

4.7.3 Відхилення зовнішнього діаметра різьби ніпеля штанги

В результаті частих згвинчувань різьба ніпеля може зношуватись по зовнішньому діаметру. Необхідно виявити залежності показників надійності РЗ від величини зовнішнього діаметра ніпеля і обґрунтувати його допустимі значення. Для цього розглянуто осисиметричну СЕМ РЗ ШН 19 мм за ГОСТ 13877 (еквівалент 3/4 in. API Spec 11B) [11]. Враховували пластичність матеріалу і тертя. Розглядали два кроки зовнішнього осьового навантаження, яке створює напруження в тілі $\sigma_{\min} = 0$ МПа and $\sigma_{\max} = 276$ МПа. Згвинчування імітували осьовою деформацією упорного кінця муфти на $\Delta = 0,1$ мм. Контактний тиск обчислювали на робочій стороні профілю в точках 1 (мінімальний діаметр різьби муфти) та 2 (максимальний діаметр різьби ніпеля).

Максимальний діаметр різьби ніпеля d змінювали з 26,624 мм (максимально допустимий за ГОСТ) до 25,624 мм. Мінімальний діаметр різьби муфти постійний – 24,79 мм (максимальний допустимий за ГОСТ). Помітно, що зменшення d впливає на розподіл напружень (рис. 4.18).



а) – $d = 26,624$ мм; б) – $d = 25,624$ мм

Рисунок 4.18 – Розподіл напружень σ_m (МПа) в перших витках

У випадку зменшення діаметра d найбільш помітним є зменшення втомної міцності в зоні робочої останньої (відносно торця) западини різьби муфти (рис. 4.19).

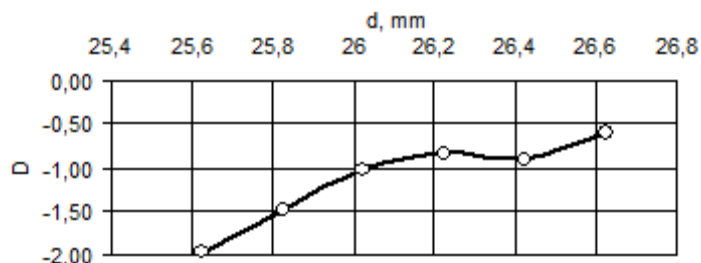


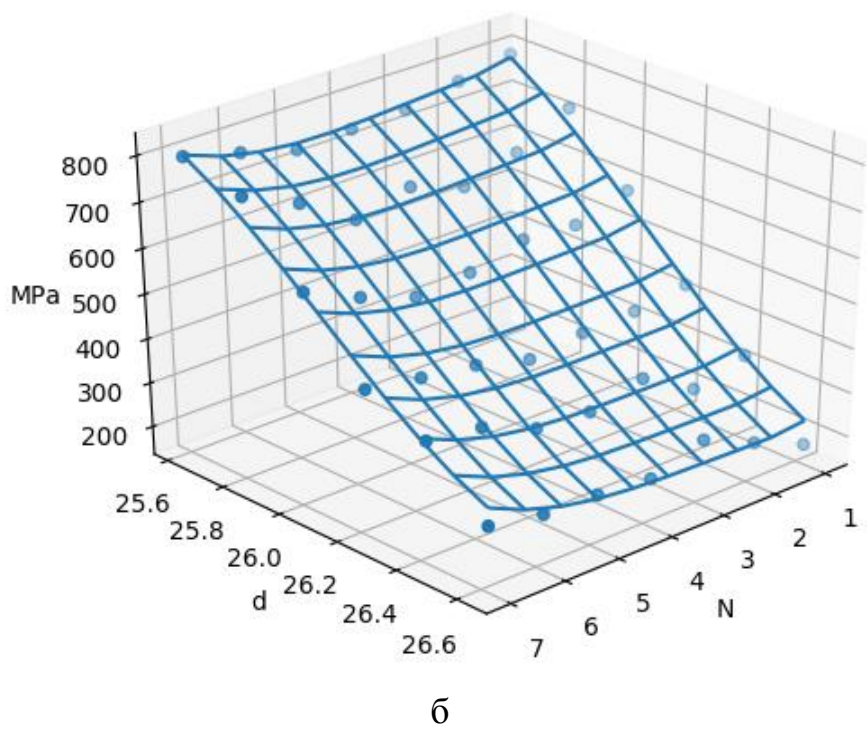
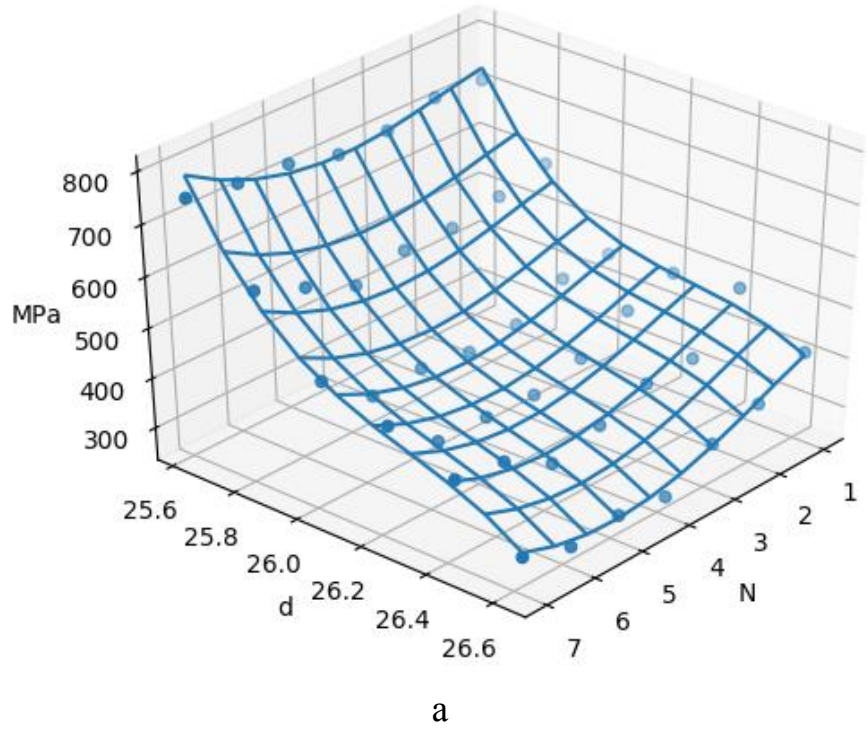
Рисунок 4.19 – Залежність коефіцієнта запасу D в останній робочій западині різьби муфти від діаметра d

На основі результатів скінченно-елементного аналізу шукали регресійну залежність контактних тисків P_c в точках 1 і 2 від номера витка різьби N і зовнішнього діаметра d

$$P_c = a_0 + a_1 N^{p_0} + a_2 N^{p_1} + a_3 N^{p_2} + a_4 d^{p_0} + a_5 d^{p_1} + a_6 d^{p_2} + a_7 N d,$$

де a_i – коефіцієнти, p_i – степені.

Значення коефіцієнтів і показників степеня для точки 1 (рис. 4.20а): $a = (16730205,6, 612,225399, 15,9368037, 0,278069341, -2431458,31, 87636,6121, -1092,26421, -23,2280329)$; $p = (0,9, 1,9, 2,9)$; $R^2 = 0,97$. Значення коефіцієнтів і показників степеня для точки 2 (рис. 4.20б): $a = (-85251526,6, -2157,06639, -136,373263, 56,2253835, 68049001,0, -70806,1914, 19700,4893, 11,9405460)$; $p = (0,1, 1,9, 2,2)$; $R^2 = 0,98$. З цих залежностей видно, що для значень d менших за 26,2 мм контактні тиски різко зростають. Залежності $P_c(N, d)$ можуть бути використані для вибору допустимих значень максимального діаметра d різьби ніпеля.



а) – в точці 1; б) – в точці 2

Рисунок 4.20 – Залежності P_c від N і d

4.8 Обґрунтування застосування двоопорних різьбових з'єднань порожнистих насосних штанг

В підрозділі за допомогою ітеративного проектування, яке базується на параметричному геометричному моделюванні та МСЕ, обґрунтовано доцільність застосування двоопорних РЗ для порожнистих ШН та удосконалено їхню конструкцію [9, 14]. Базовою гіпотезою є можливість застосування двоопорних РЗ бурильних труб [74, 75] для порожнистих ШН. Для підтвердження базової гіпотези застосовано методику дослідження та проектування, яка використовує геометричну та скінченно-елементну параметричні моделі РЗ, що здатні до внесення нових елементів, базу знань з результатами досліджень РЗ та передбачає виконання послідовних або паралельних циклів проектування (ітерації), що можуть складатися з таких етапів:

- 1.Формулювання множини вимог до РЗ та гіпотез щодо його удосконалення.
 - 2.Аналіз та відбір цих вимог і гіпотез. Додання в параметричну модель елемента, який їх реалізує, та вибір допустимих значень його параметрів.
 - 3.Симуляція моделі МСЕ та оптимізація параметрів нового елемента.
 - 4.Оцінка результатів симуляції шляхом порівняння з попередніми ітераціями.
- Прийняття чи відхилення гіпотези. Запис результатів до бази знань.

У зв'язку зі складністю формалізації задачі, наявності багатьох вимог, параметрів та критеріїв оптимальності ця методика повинна опиратись на ІС, яка ізоморфна до інших складних систем і володіє загальносистемними закономірностями [213]. З цих закономірностей випливає, що на різних етапах проектування повинні використовуватись різноманітні методи, які доповнюють одне одного, наприклад якісні та кількісні, евристичні та детерміновані, аналітичні та синтетичні, теоретичні та емпіричні. Так етап 1 повинен, в основному, використовувати якісні та евристичні методи (морфологічні, "мозкової атаки", експертних оцінок) та більше опиратись на інтуїцію і досвід проектувальника. Процес проектування може мати самоподібну структуру – етапи

ітерації можуть містити ізоморфні до неї субітерації. Для прикладу, в етапі 3 такі субітерації призначені для оптимізації параметрів.

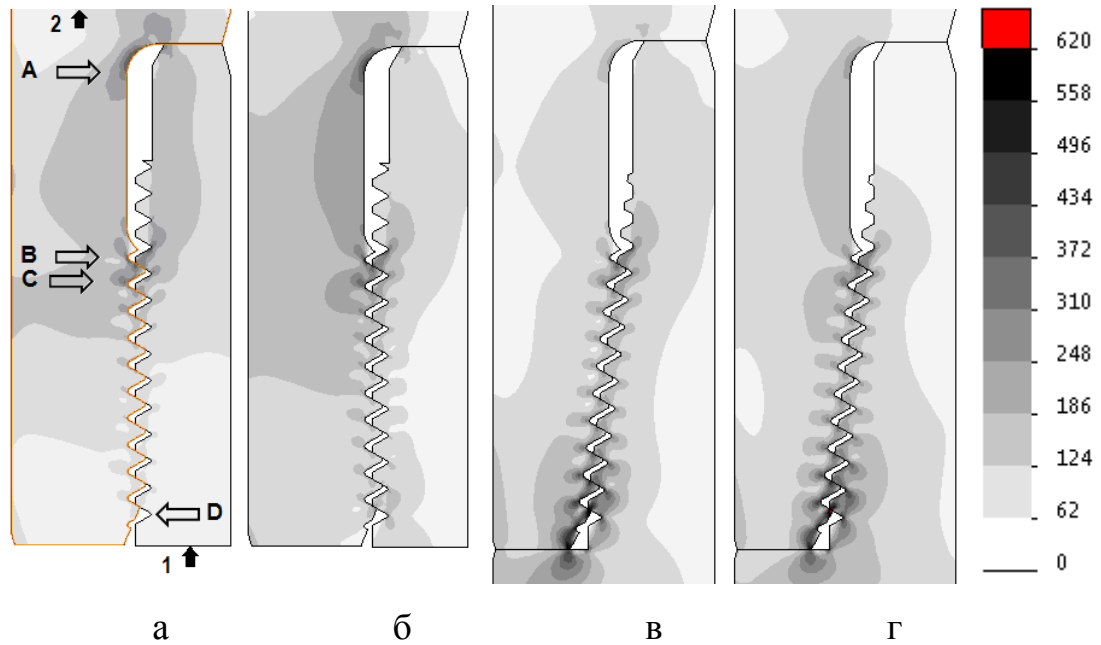
На першому етапі дослідження проаналізовано осесиметричну СЕМ замкового РЗ 3-66 замка ЗН-80 з нормальним прохідним отвором за ГОСТ 5286-75 (одноопорне РЗ) [325-327]. Розв'язувалась задача з пружною моделлю матеріалу та врахуванням тертя між поверхнями контакту. У разі відсутності зовнішнього навантаження найвищі напруження спостерігаються в першому витку ніпеля, а під час дії зовнішнього навантаження величиною 1 МН – в першому витку ніпеля і останньому витку муфти (рис. К.1). На практиці ці зони володіють низькою втомною міцністю. Моделі свідчать про можливість проникнення середовища в зону перших витків ніпеля (навіть за відсутності зовнішнього навантаження) внаслідок збільшення зазорів між неробочими сторонами профілю (рис. К.1). На рис. для візуалізації деформації збільшено в 100 раз. Під час зовнішнього навантаження величиною 1 МН ці зазори значно зростають і з'являється зазор в упорному торці. Локальні напруження таких великих величин (до 1 ГПа) пояснюється застосуванням пружної моделі матеріалу.

Проводилося також моделювання в середовищі Abaqus з врахуванням тертя між поверхнями контакту і пластичності матеріалу (сталь 40) [196, 314, 315]. Збільшення Δ (рис. К.2в) і застосування муфти з пластичнішого матеріалу (рис. К.2г) дещо вирівнює напруження у впадинах різьби ніпеля, але й майже не змінює їх у перших впадинах [315]. Для обчислення D використовували два значення σ_p : 0 і 155,1 МПа [93]. Розглядали моделі з різними границями плинності матеріалу муфти: 300, 400, 500 МПа. Величину згвинчування Δ змінювали від 0 до 0,3 мм з кроком 0,05 мм. Алгоритм розрахунків для здійснення оптимізації реалізується у модулі main2.py (лістинг И.6). Помітно, що найбільш небезпечними з точки зору втоми зонами є перші три впадини різьби ніпеля (рис. К.2д). Порівняння з розподілом D РЗ ШН показує нижчу міцність замкового РЗ. У першу чергу це

пояснюється відсутністю зарізьбової канавки. Результати (рис. К.3) показали, що РЗ з муфтою зі сталі з $\sigma_T=300$ МПа є дещо кращим з точки зору втомної міцності [93]. Проте така сталь володіє меншою стійкістю до зношування. Діаграму (рис. К.3) можна використовувати для обґрунтування вибору матеріалу муфти та величини згвинчування.

Параметрична геометрична модель РЗ ШН розроблена в SOLIDWORKS. Осесиметричні СЕМ розроблено в SOLIDWORKS Simulation 2018. Матеріалом деталей є сталь з σ_T 620 МПа та σ_B 724 МПа з моделлю пластичності за Мізесом. Вибрано опцію великих деформацій (Large Strain Option). Створено глобальний контакт поверхонь без проникнення, а на упорних торцях – контакт "гаряча посадка" (Shrink Fit), що дозволяє моделювати натяги. Введено коефіцієнт тертя спокою поверхонь в умовах змащення 0,16. На нижньому торці муфти 1 (рис. 4.21а) створено граничну умову симетрії, а на верхній торці ніпеля 2 діє тиск, який утворює в тілі ШН напруження розтягу $\sigma_p=0$ МПа (перший крок) та $\sigma_p=150$ МПа (другий крок). Тип сітки "Blended curvature-based mesh", максимальний розмір елементів – 2 мм, мінімальний – 0,2 мм, мінімальна кількість елементів на колі – 8, показник росту розміру елементів – 1,1. Такі налаштування дозволяють отримати дрібну сітку в зоні різьби і велику в інших зонах. Для обчислення значень коефіцієнта запасу D для циклу навантаження $\sigma_p=0..150$ МПа використовували залежність Сайнса (1.8) зі значенням σ_{-1} 207 МПа. Автором розроблено програму для автоматизації обчислення значення D (лістинг Н.1).

Спочатку порівнювали стандартне РЗ порожнистих ШН діаметром 25 мм (ГОСТ 31825-2012) та двоопорне РЗ з конічною різьбою з кутом нахилу до осі $7,125^\circ$. Інші параметри двоопорного РЗ відповідають стандартному. За даними експлуатації стандартних РЗ [1] найменшу втомну міцність мають зони А, В, С, D (рис. 4.21). В цих зонах розраховані значення D (рис. 4.22) для різних величин натягів на основній Δ_1 і додатковій Δ_2 опорах. Стандартне РЗ з натягом 60 мкм на рис. позначено як "60std".



а, б) – стандартне з $\Delta_1=0,06$ мм; в, г) – конічне двоопорне РЗ з $\Delta_1=\Delta_2=0,04$ мм;
 а, в) – $\sigma_p=0$ МПа; б, г) – $\sigma_p=150$ МПа

Рисунок 4.21 – Напруження σ_m (МПа) в РЗ порожнистих ШН

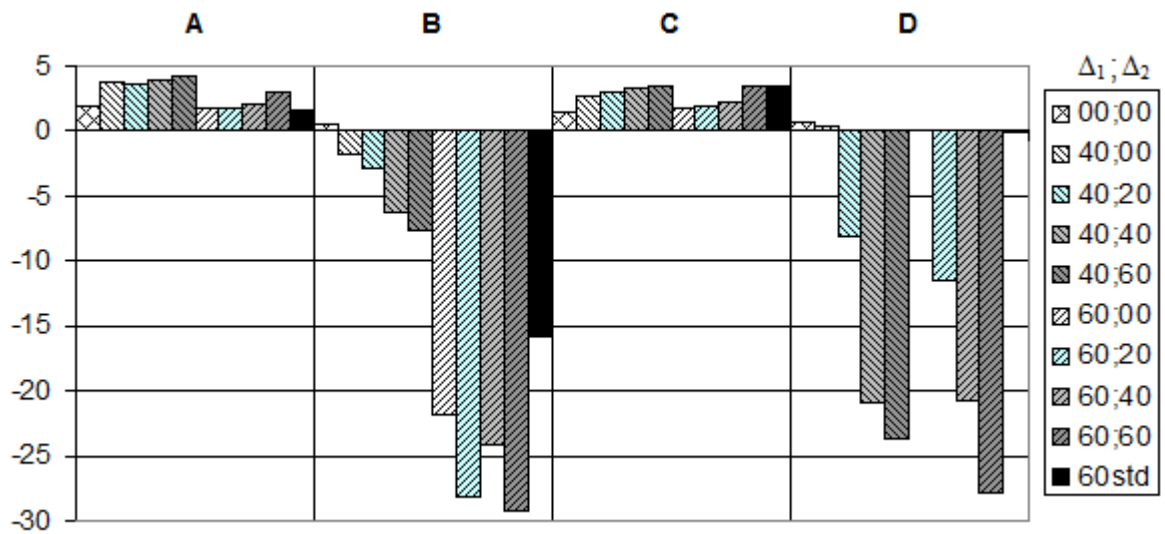


Рисунок 4.22 – Значення D у небезпечних зонах РЗ для різних величин натягів Δ_1 та Δ_2 (мкм)

Порівняння варіантів "60std" та "60;00" (рис. 4.22) підтверджує гіпотезу про нижчу втомну міцність конічного ніпеля, але втомна міцність конічної муфти підвищується незначно. У незгвинченого РЗ ($\Delta_1=\Delta_2=0$) значення D в зонах А, С найменше, а в зонах В, D – найбільше. Проте експлуатація незгвинченого РЗ неприпустима. Натяг $\Delta_1=0,06$ мм зменшує значення D в усіх цих зонах (в зоні В суттєво) у порівнянні з натягом 0,04 мм. Отже, в даному випадку, оптимальне значення Δ_1 потрібно шукати біля значення 0,04 мм.

В двоопорному РЗ розподіл σ_m по витках різьби ніпеля більш рівномірний (рис. 4.21). Внаслідок значної жорсткості муфти збільшення натягу Δ_2 різко підвищує напруження в зоні D, контактні напруження на додатковій опорі (рис. 4.21), значення D в зонах А, С і різко зменшує його в зонах В, D (рис. 4.22). Більше це властиво для натягу $\Delta_1=0,06$ мм. Найбільше зменшується коефіцієнт запасу в зоні D. Збільшення натягу Δ_2 може бути корисним для запобігання самовідгвинчуванню РЗ і підвищення герметичності, але суттєво зменшує втомну міцність в зоні D, що вимагає застосування матеріалу муфти з більшою границею витривалості, або додаткових конструктивних рішень для зменшення концентрації напружень у зонах В, D (зарізьбова канавка заданої довжини, зріз витків та ін.). Найбільш перспективним є комплексне рішення – зарізьбова канавка, корекція перших витків або попереднє деформування РЗ високим моментом згвинчування [315].

Оптимальним натягами в конічному двоопорному варіанті РЗ (рис. 4.21в, г) можуть бути $\Delta_1=0,04\pm 0,01$, $\Delta_2=0,04^{+0,02}$. Вони забезпечують вищу втомну міцність ніпеля, ніж стандартне РЗ, але їх застосування вимагає додаткового забезпечення втомної міцності муфти. Натяг Δ_2 не повинен бути значно більшим Δ_1 .

Наступним етапом є дослідження двоопорних конічних РЗ із зарізьбовою канавкою на ніпелі та муфті. Нижче описано послідовність ітерацій дослідження.

Ітерація 1. Вимога – зменшити концентрацію напружень у зоні D. Гіпотеза – застосування зарізьбової канавки муфти, жорсткість якої нижча за жорсткість різьбової частини (рис. 4.23а). На жорсткість канавки впливають її діаметр і довжина. У першому наближенні прийнято довжину 16 мм та діаметр 30 мм, який

рівний мінімальному діаметру різьби. Без змін залишено зарізьбову канавку ніпеля (діаметр 31,04 мм, довжина 20 мм). Округлення канавок має максимальні можливі значення радіусів – 3 мм. Натяги $\Delta_1=0,06$ мм, $\Delta_2=0,06$ мм. У порівнянні з аналогічним РЗ без канавок "60;60" (рис. 4.22) досягнуто дещо вищого значення D у зонах В і D (рис. 4.24). Розподіл напружень (рис. 4.25) більш рівномірний. Контактні напруження на додатковій опорі можна зменшити шляхом зменшення жорсткостей канавки на муфті або додаткового плеча ніпеля.

Ітерація 2. Вимога – збільшити допуск на величину натягу Δ_2 . Гіпотеза – застосування пружного елемента між торцями ніпеля і муфти. Жорсткість елемента повинна забезпечувати максимальний можливий контактний тиск на торці та пружну деформацію величиною 0-0,2 мм. Технологічно найпростіше виготовити на циліндричній частині канавки шириною h_k і глибиною t_k , які розташовані на зовнішній і внутрішній поверхні циліндра з кроком p_k (рис. 4.23б). Методом перебору по сітці знайдені оптимальні значення $h_k=0.5$ мм, $t_k=4.5$ мм, $p_k=6$ мм. Симуляція такого РЗ з натягами $\Delta_1=0,06$ мм, $\Delta_2=0,2$ мм показала значне збільшення D в зоні D (рис. 4.24) у порівнянні з ітерацією 1. Недоліком є зменшення контактних напружень на додатковій опорі з 200 МПа до 100 МПа (рис. 4.23б) та дещо менше навантаження останніх і середніх витків (рис. 4.25). Такий варіант РЗ слід використовувати тоді, коли немає можливості контролювати натяг Δ_2 .

Ітерація 3. Вимога – зменшення концентрації напружень в скругленнях зарізьбових канавок. Гіпотеза – застосування зарізьбових канавок еліптичного профілю для вирівнювання напружень в них. Спочатку дослідимо вплив таких канавок на РЗ, отримане на ітерації 1. Використаємо асиметричне скруглення канавок еліптичного профілю (рис. 4.23в) двома радіусами на ніпелі (4 мм та 10 мм) і на муфті (4 мм та 8 мм). Збільшення радіусів скруглення призводить до збільшення жорсткості канавки, тому мінімальний діаметр канавки зменшено до 28 мм на ніпелі та збільшено до 35 мм на муфті. Діаметр канавки муфти не повинен бути дуже великим у зв'язку з можливістю спрацювання зовнішньої поверхні муфти. Симуляція з натягами $\Delta_1=0,06$ мм, $\Delta_2=0,06$ мм показала менші

значення еквівалентних напружень в канавках, їх більш рівномірний розподіл вздовж канавки (рис. 4.23в) та збільшення D в усіх зонах (рис. 4.24). Найбільш небезпечною залишається зона В ($D= -11$). Модифікація канавки призвела до зменшення її жорсткості та зменшення контактних напружень на торцях, що вимагає збільшення величини натягу. Розподіл напружень (рис. 4.25) дещо гірший, ніж у ітерації 1.

Ітерація 4. Вимога і гіпотеза – як у ітерації 2. Симуляція з $\Delta_1=0,06$ мм та $\Delta_2=0,2$ мм показала, що застосування пружного елемента не зменшує втомну міцність РЗ (рис. 4.24). Небезпечною залишається зона В ($D= -8$). Розподіл напружень (рис. 4.25) кращий, ніж у стандартного РЗ, але гірший, ніж у ітераціях 1-3. У порівнянні з ітерацією 2 контактні напруження на додатковій опорі незначно зменшились (рис. 4.23г).

Ітерація 5. У зв'язку з можливістю пошкодження пружної частини необхідно довести працездатність результату ітерації 4 для натягу $\Delta_2=0$. Результати симуляції для $\Delta_1=0,06$ мм показують, що втомна міцність РЗ не погіршується (рис. 4.24). Розподіл напружень (рис. 4.23д, 4.25) дещо кращий, ніж у стандартного.

Ітерація 6. Вимога – збільшити контактні напруження на основній опорі. Методом перебору знайдено значення натягу $\Delta_1=0,07$ мм, який забезпечує орієнтовне значення контактних напружень на основній опорі 100 МПа для $\sigma_p=150$ МПа (рис. 4.23е). Натяг $\Delta_2=0,2$ мм. Результати показують, що збільшення натягу Δ_1 до 0,07 мм не суттєво погіршує втомну міцність РЗ (рис. 4.24). Небезпечною залишається зона В ($D= -10$), але вона міцніша, ніж в стандартному РЗ ($D= -16$). Розподіл напружень (рис. 4.25) дещо гірший, ніж у ітерації 2.

Ітерація 7. Перевірка працездатності РЗ під час втрати натягу $\Delta_2=0$. Натяг $\Delta_1=0,07$ мм. В цьому випадку значення D в зоні В зменшується до -11, але є більшим, ніж в стандартному (рис. 4.24). Розподіл напружень (рис. 4.25) є кращим, ніж у стандартного РЗ.

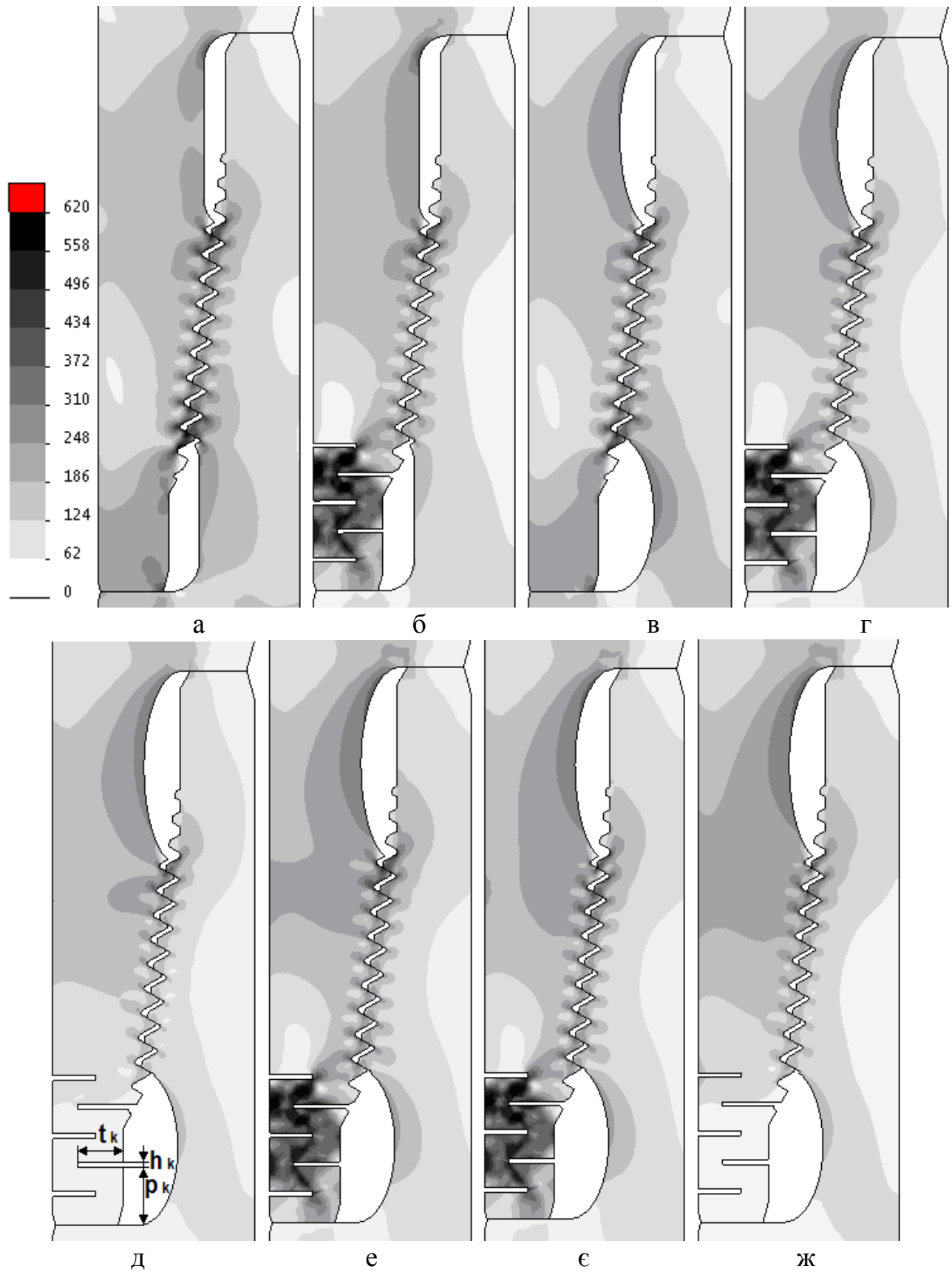


Рисунок 4.23 – Напруження σ_m (МПа) в РЗ порожнистих ШН для $\sigma_p=150$ МПа

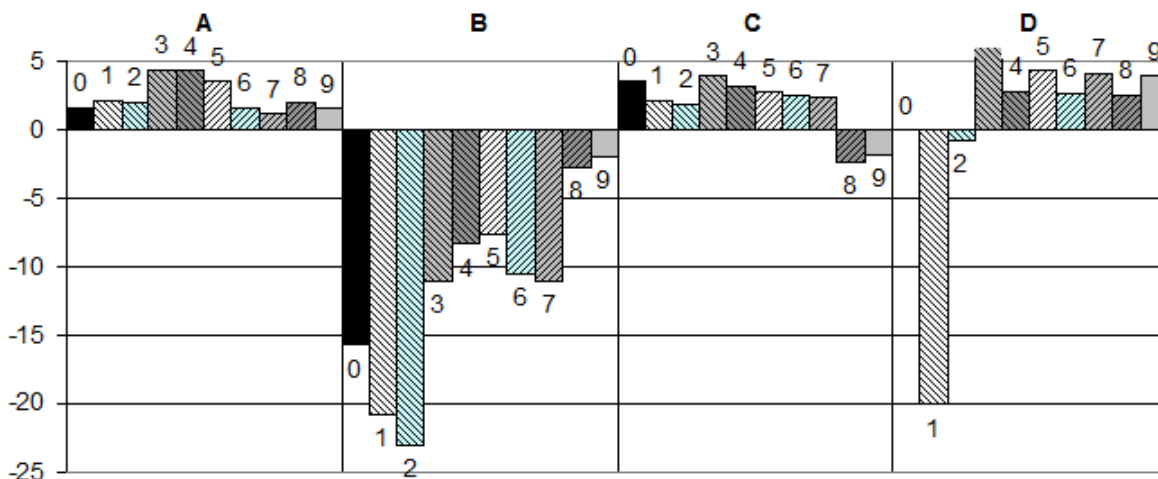
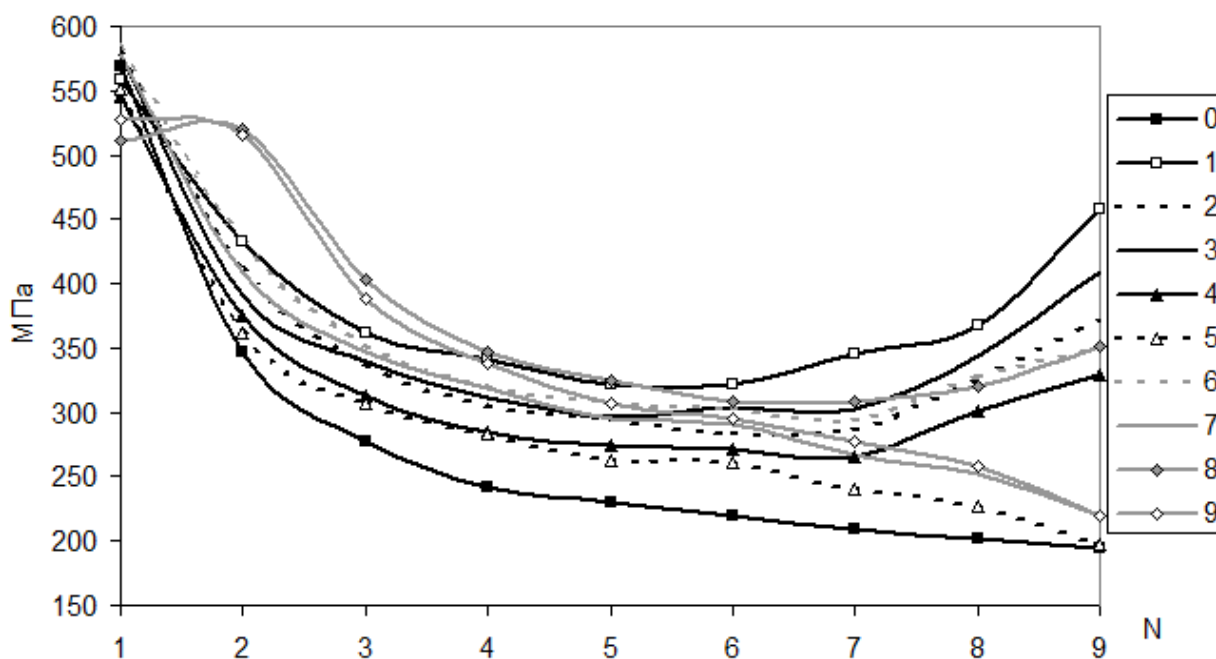


Рисунок 4.24 – Значення D у небезпечних зонах РЗ для результатів ітерації 1-9



0 – стандартне; 1-9 – ітерації

Рисунок 4.25 – Значення σ_m в западинах різьби ніпеля

Ітерація 8. Вимога – збільшити значення D у зоні В. Для цього можна застосувати відомі рішення – корекція перших витків або попереднє деформування РЗ високим моментом згвинчування з метою перерозподілу навантаження на інші витки. В даному випадку створена асиметрична фаска $1,5 \times 0,01$ мм на вершині робочій стороні першого витка ніпеля. Це створює зазор величиною $3..9$ мкм між профілем витка ніпеля і муфти. Натяги $\Delta_1 = 0,07$ мм,

$\Delta_2=0,2$ мм. Результат – значне зменшення напружень (рис. 4.23є) та збільшення значення D в зоні В (рис. 4.24). Тепер значення D у зонах В і С майже рівні. Розподіл напружень (рис. 4.25) гірший, ніж у ітерацій 1, 2, 3, але кращий, ніж у інших ітерацій.

Ітерація 9. Перевірка роботоздатності РЗ під час втрати натягу Δ_2 . ($\Delta_1=0,07$ мм, $\Delta_2=0$). Значення D майже не змінились (рис. 4.24). Розподіл напружень (рис. 4.23ж, 4.25) кращий, ніж у ітерації 7.

У двох останніх ітераціях зменшено навантаження на перший виток ніпеля, але збільшено на наступні три. Ці витки теж можна коректувати, для перерозподілу навантаження на середню частину різьби. Проте обчислення оптимальної величини корекції на кожному витку є предметом окремих досліджень.

Оскільки наявність конуса не підвищує втомної міцності РЗ і не значно змінює площі поперечного перетину деталей, а операції згвинчування-розгвинчування не є частими, то слід також розглянути можливість застосування двоопорного циліндричного РЗ. Таке РЗ можна легко впровадити без значної модифікації стандартного ніпеля. Потрібно тільки коригувати шляхом точіння форму канавки і виготовити нові муфти із запропонованою в статті конструкцією (рис. 4.23в-ж). Пружний елемент в зоні додаткової опори може бути окремою деталлю РЗ. Симуляція такого РЗ показала, що розподіл еквівалентних напружень подібний на розподіл на рис. 4.23є, а значення D в зонах А, В, С, D рівне 2,3; -3,6; -0,9; 0. Це майже відповідає варіанту 8 (рис. 4.24).

4.9 Аналіз з'єднання насосно-компресорних труб за критерієм герметичності

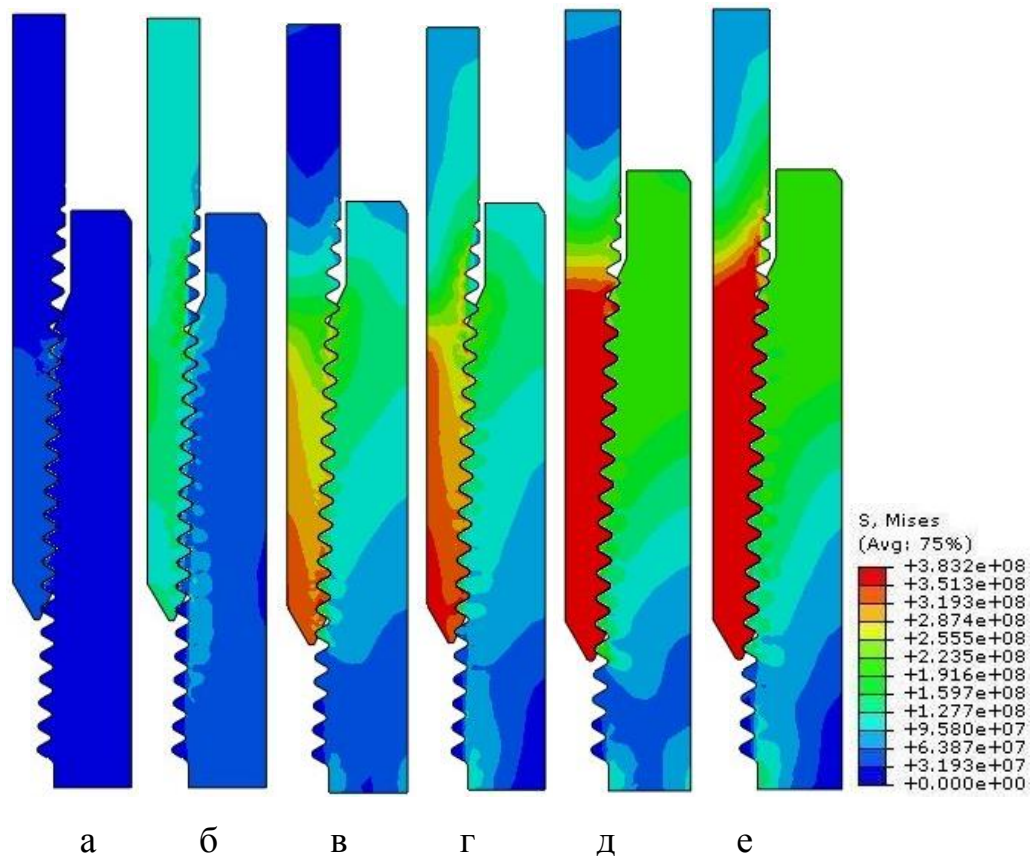
Для виявлення характеру розподілу напружень в муфтовому РЗ НКТ була розроблена осесиметрична СЕМ стандартного муфтового РЗ гладких НКТ умовним діаметром 114 мм ГОСТ 633-80 з використанням програми Ansys® [7, 94]. Модель матеріалу – пружна ($E=2,1 \cdot 10^{11}$ Па, $\nu=0,28$). Враховували тертя між

поверхніми контакту (коефіцієнт тертя $f=0,1$). Для моделювання зусилля згинчування РЗ використано можливості Ansys® для визначення початкового проникнення між поверхніми контакту (initial penetration).

На рис. Л.1 показано напружено-деформований стан РЗ з натягами $A=0$ мм (на верстаті), $A=3,175$ мм та $A=6,35$ мм (вручну) у разі відсутності зовнішнього навантаження (а) та під час дії зовнішнього навантаження, що відповідає σ_p 100МПа (б). На рис. деформації умовно збільшено в 10 раз. Локальні напруження таких великих величин (до 1 ГПа) пояснюється застосуванням пружної СЕМ, де не враховуються пластичні деформації. У незгинченому РЗ під час дії зовнішнього осьового навантаження розтягу розкривається контакт між усіма витками (рис. Л.2). Найбільші напруження спостерігаються у першій робочій западині ніпеля. У згинчених РЗ великі значення напружень спостерігаються біля торця ніпеля та в перших витках РЗ. У згинченого на муфтонаверточному верстаті РЗ ($A=0$ мм) ці напруження перевищують границю плинності. Ця картина напружень непогано збігається з результатами, отриманими методом тензометрування та заморожування моделей з оптично активного матеріалу [328]. Розподіл навантажень між витками характерний для РЗ такого типу (стяжки) [71]: найбільше навантажуються перші витки, дещо менше – останні, і найменше – середні (рис. Л.2). Однак розподіл контактного тиску по поверхні витка нерівномірний і найбільша ця нерівномірність саме в середніх витках. В них найбільше значення контактного тиску – біля вершини витків ніпеля. У разі згинчування РЗ це явище поширюється на всі витки. Під час дії зовнішнього навантаження розтягу на правій стороні 3, 4, 5-го витків (починаючи зі збігу) ніпеля навіть у згинчених РЗ контактний тиск майже рівний нулю (рис. Л.2 в, д).

Аналізували також СЕМ з пружно-пластичною моделлю матеріалу (сталь 40) [93]. На рис. 4.26 показано розподіл еквівалентних напружень в осесиметричній пластичній СЕМ РЗ гладких НКТ умовним діаметром 114 мм (ГОСТ 633-80), згинченого з натягами 6,5 мм (згинчування вручну), 3,325 мм, 0,15 мм (згинчування на верстаті) і під час зовнішнього навантаження, яке створює напруження в тілі НКТ 100 МПа. Зусилля згинчування моделювалось

шляхом зміщення муфти в осьовому напрямку на величину, кратну кроку різьби. Моделювання дає змогу виявити розподіл навантажень та зазорів між витками різьби, які характеризують монолітність РЗ, і тому можуть бути критеріями руйнування, розгвинчування та втрати герметичності РЗ [196, 314, 315]. Ці результати подібні на результати СЕМ з пружною моделлю матеріалу [7, 94].



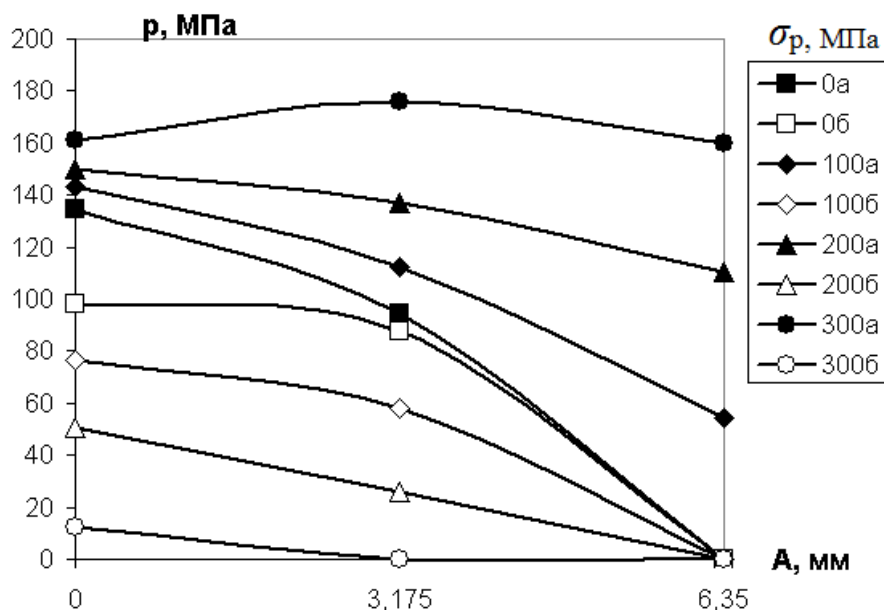
а, в, д) – $\sigma_p=0$ МПа; б, г, е) – $\sigma_p=100$ МПа;

а, б) – $A=6,5$ мм; в, г) – $A=3,325$ мм; д, е) – $A=0,15$ мм

Рисунок 4.26 – Розподіл σ_m (Па) в муфтовому РЗ НКТ з умовним діаметром 114мм

На рис. 4.27 показана емпірична залежність середнього контактного тиску на сторонах витків різьби від величини натягу для різних значень зовнішнього навантаження [93]. Середній контактний тиск підраховувався на перших 13 витках різьби ніпеля (від його торця). Для побудови цієї залежності натяг A змінювали так: 0; 3,175; 6,35 мм, а зовнішнє навантаження, що відповідає σ_p , так: 0, 100, 200, 300 МПа. Алгоритм для здійснення оптимізації реалізується у модулі main3.py (лістинг И.7). Під робочими сторонами витків слід розуміти ті, які під час зовнішнього навантаження розтягу сприймають більше навантаження. Ця

залежність дозволяє обґрунтувати вибір натягу РЗ для забезпечення його максимальної герметичності.



а) – робочі (навантажені) сторони; б) – неробочі

Рисунок 4.27 – Емпірична залежність середнього контактної тиску p на сторонах витків різьби від натягу A для зовнішнього навантаження, що відповідає σ_p

Розподіл контактних тисків по конкретним виткам можна побачити на рис. 4.28. Використовуючи ці дані, за допомогою програми Maplesoft Maple, отримано теоретичні залежності у вигляді поліномів і знайдено їхні максимуми [93]. Для робочих сторін витків:

$$p = 137755875,728 + 26208,467\sigma_p + 153,641\sigma_p^2 - 11034648,201A - 1565917,372A^2 + 70672,940\sigma_p A \text{ (Па)},$$

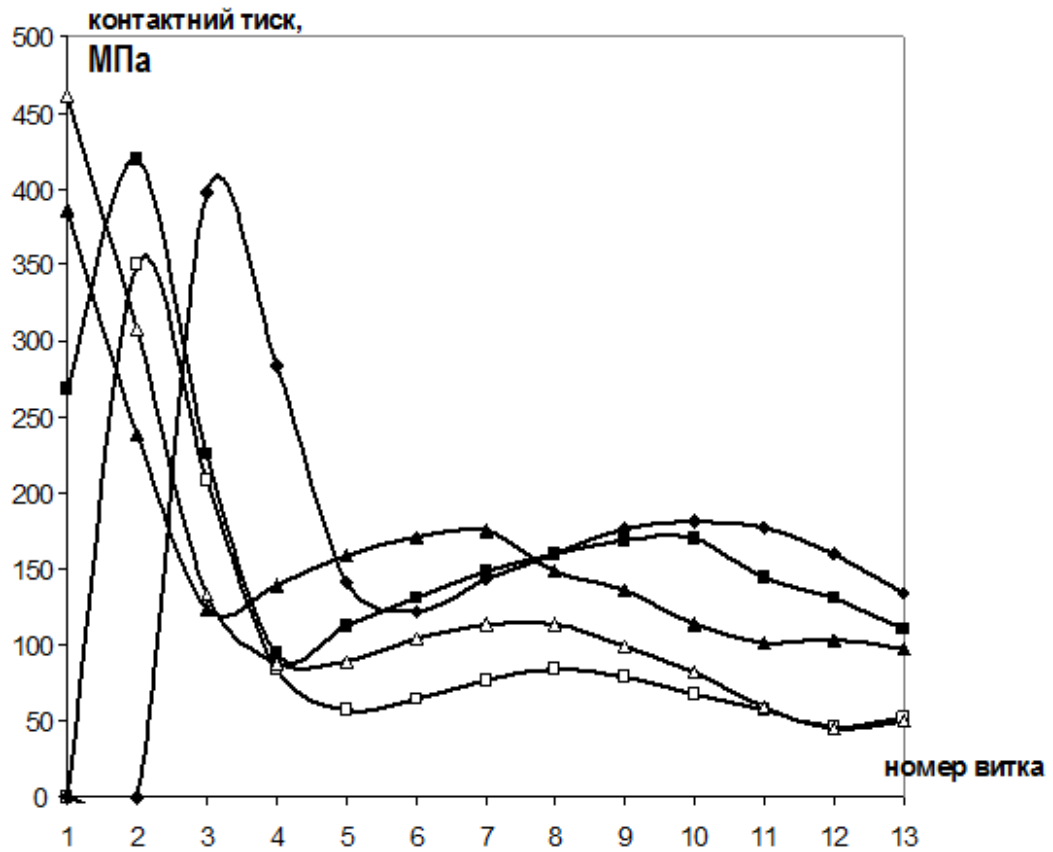
$$p_{\max} = 175949722,8 \text{ Па } (\sigma_p = 300 \text{ МПа}, A = 3,246 \text{ мм}).$$

Для неробочих сторін витків:

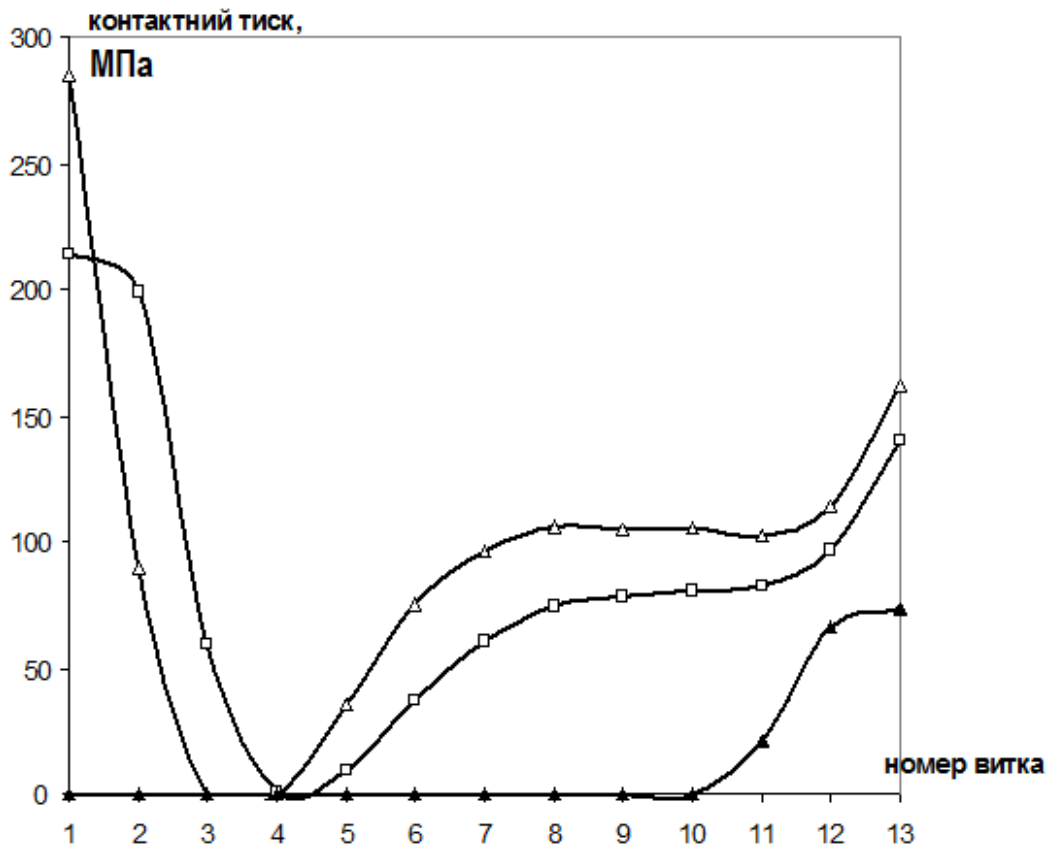
$$p = 108534427,316 - 300357,627\sigma_p - 112,730\sigma_p^2 - 7775735,702A - 1305561,992A^2 + 44603,412\sigma_p A \text{ (Па)},$$

$$p_{\max} = 108534427,316 \text{ Па } (\sigma_p = 0 \text{ МПа}, A = 0 \text{ мм}).$$

На рис. 4.29 можна побачити відповідні графіки та визначити теоретичні максимуми контактної тиску для $\sigma_p = 100, 200, 300$ МПа.



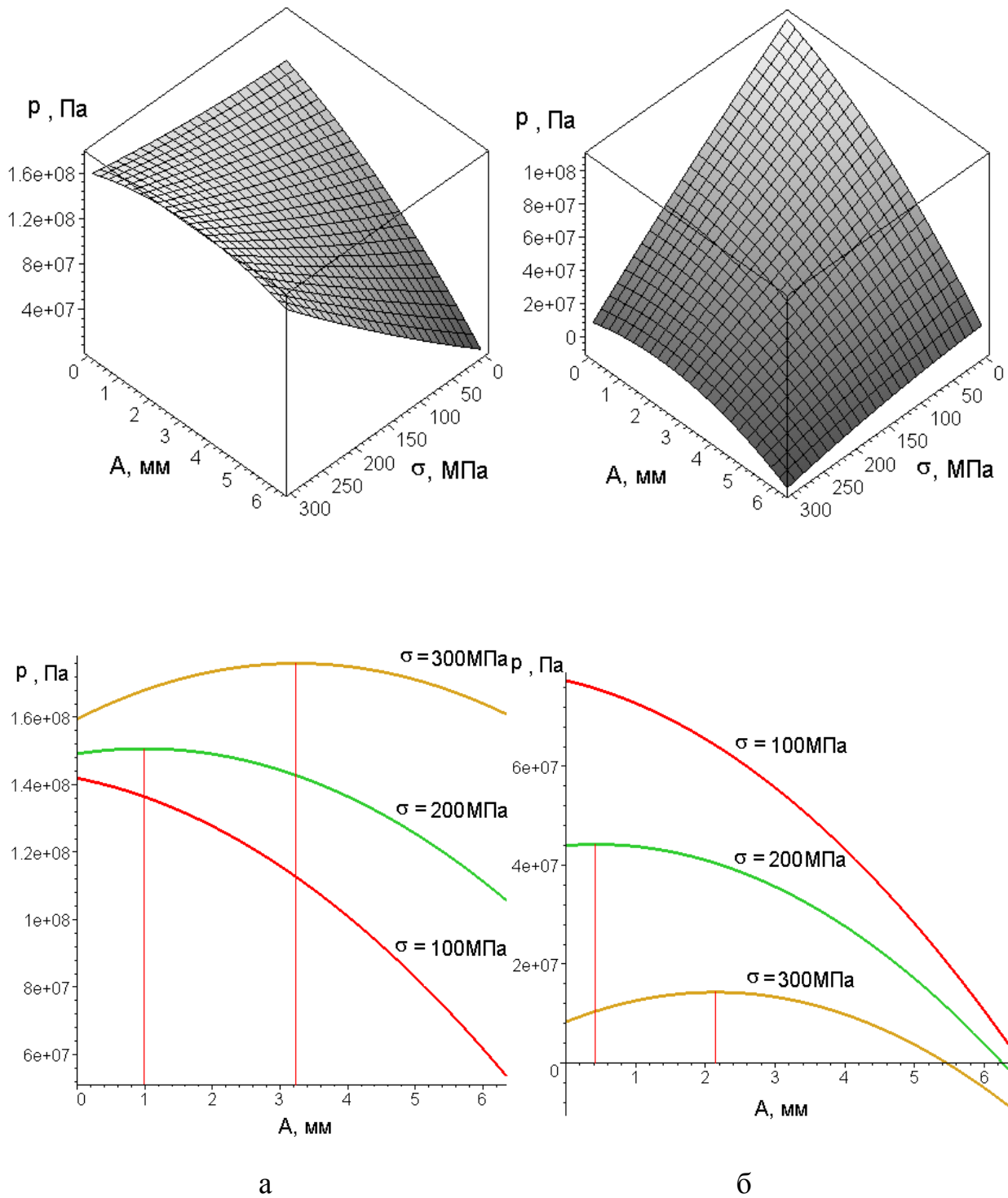
а



б

а) – робочі (навантажені) сторони; б) – неробочі
 \square, Δ – $\sigma_p=0$ МПа; $\diamond, \blacksquare, \blacktriangle$ – $\sigma_p=300$ МПа;
 \diamond – $A=6,35$ мм; \square, \blacksquare – $A=3,175$ мм; Δ, \blacktriangle – $A=0$ мм;

Рисунок 4.28 – Контактний тиск на сторонах витків різьби



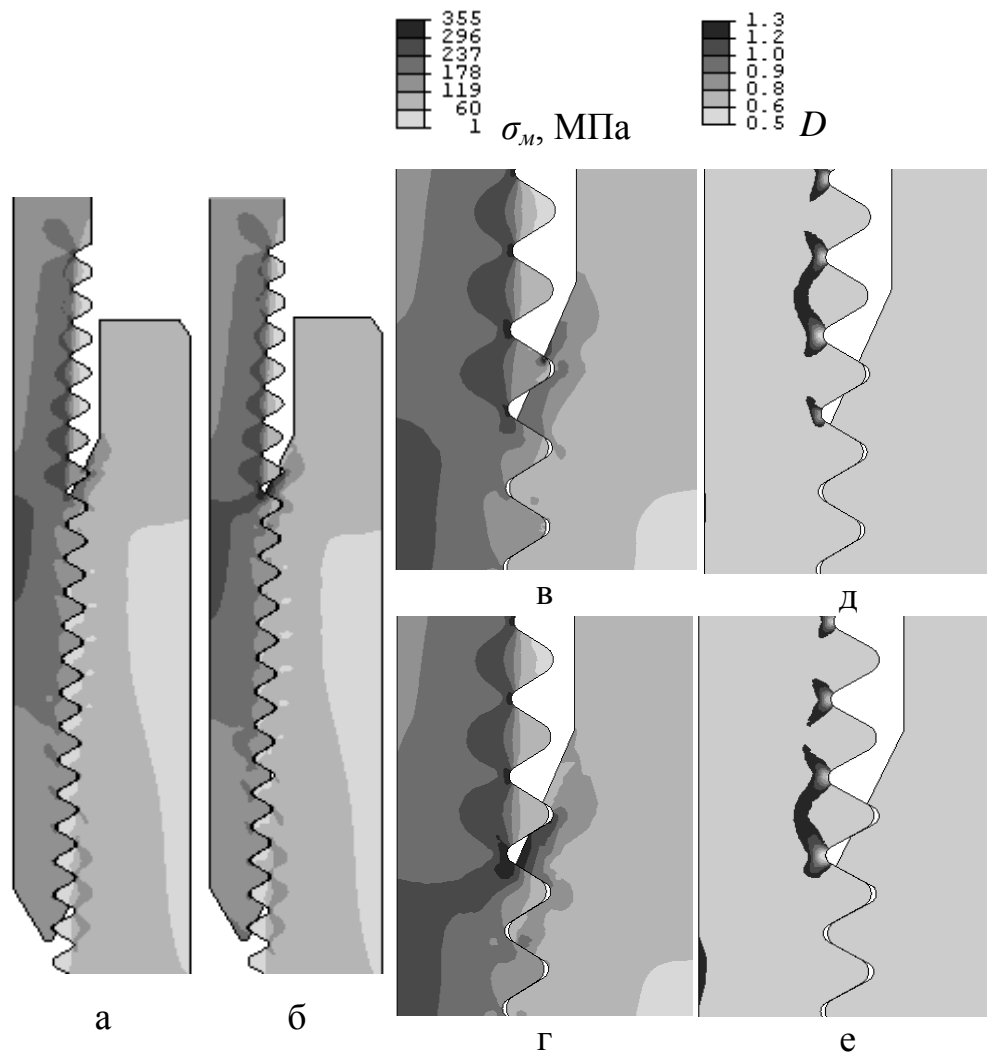
а) – робочі (навантажені) сторони; б) – неробочі сторони

Рисунок 4.29 – Теоретична залежність середнього контактного тиску на сторонах витків різьби ніпеля p від величин A і σ_p

4.10 Обґрунтування застосування різьб НКТ, які виготовлені некоригованими різцями з від'ємним переднім кутом

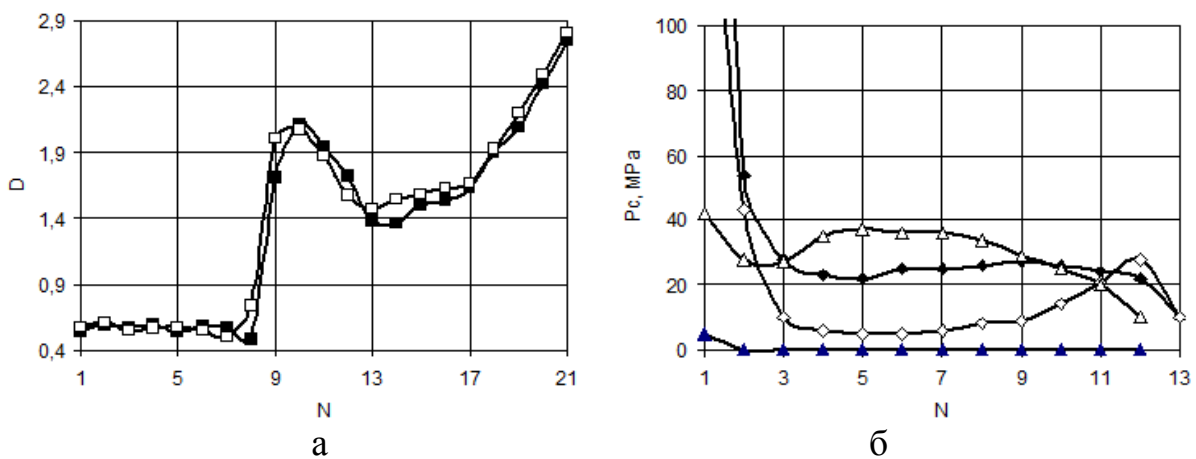
На основі геометричної моделі РЗ гладких НКТ з зовнішнім діаметром 114 мм, різьба ніпеля яких має відхилення кута профілю внаслідок нарізання некоригованим різцем з від'ємним переднім кутом γ , розроблена осесиметрична СЕМ в CalculiX [10]. Розглядали РЗ згвинчене вручну (з величиною натягу $A = 6,5$ мм), як найбільш небезпечне з точки зору герметичності. На нижньому кінці РЗ задані граничну умову нерухомості. В тілі НКТ створювали напруження $\sigma_p = 155,1$ МПа, яке моделює зовнішнє навантаження розтягу. Матеріалом деталей є сталь з $E = 210000$ МПа та $\nu = 0,28$. Використовували еластопластичну модель матеріалу з ізотропним зміцненням. Використовували контакт "поверхня-поверхня" з малим ковзанням. Коефіцієнт тертя 0,1. Розмір сітки в контактній зоні 0,3 мм. Для обчислення коефіцієнта запасу D використовували залежність Сайнса з границею витривалості 207 МПа.

Розглянемо СЕМ з'єднання НКТ діаметром 114 мм з номінальним профілем різьби на ніпелі та муфті ($\gamma = 0^\circ$, кут нахилу різальної кромки $\lambda = 0^\circ$). Також розглянемо модель де різьба ніпеля виготовлена некоригованим різцем (без модифікації профілю) з параметрами $\gamma = -10^\circ$, $\lambda = 0^\circ$, а муфта має номінальний профіль ($\gamma = 0^\circ$, $\lambda = 0^\circ$). Помітно, що розподіл еквівалентних напружень в РЗ різний (рис. 4.30). Різниця кутів профілю спричинює зростання напружень в западині профілю ніпеля і в вершині профілю муфти. Це не суттєво змінює значення D (рис. 4.30, 4.31а), але впливає на контактні тиски P_c на навантаженій стороні профілю в зоні мінімального і максимального діаметрів (рис. 4.31б). Контактні тиски на більшості навантажених сторін профілю в зоні мінімального діаметра рівні нулю. Це погіршує герметичність і створює умови до самовідгвинчування. Також виявлено, що зміна значення кута λ до $1,029^\circ$ майже не змінює значень D і P_c в цьому РЗ.



а, б, в, г) – σ_m (МПа); д, е – D ; а, в, д) – $\gamma = 0^\circ$; б, г, е) – $\gamma = -10^\circ$

Рисунок 4.30 – Розподіл σ_m і D в РЗ НКТ з ніпелем, нарізаним різцем з кутом γ



□, Δ, ◇ – $\gamma = 0^\circ$; ■, ▲, ◆ – $\gamma = -10^\circ$;

◇, ◆ – в зоні мінімального діаметра; Δ, ▲ – максимального діаметра

Рисунок 4.31 – Значення D в западині N різьби ніпеля (а) і значення контактного тиску P_c на навантаженій стороні профілю витка N ніпеля (б), якщо ніпель нарізано різцем з кутом γ

4.11 Моделювання гармонічного осьового навантажування різьбового з'єднання насосно-компресорних труб

В праці [326] за допомогою МСЕ знайдено власні частоти згвинченого замка ЗН-80: перша – 7480 Гц, друга – 22925 Гц, третя – 37191 Гц. Проведено гармонічний аналіз РЗ під час навантаження змінною зовнішньою осьовою силою величиною 150 кН в діапазоні частот 5-30 кГц. Виявлена можливість розкриття стику РЗ та втомного руйнування муфтової частини по останньому витку під час першої власної частоти. Друга власна частота більш сприятлива для втомного руйнування ніпеля по першому витку. На основі виконаних досліджень можна зробити висновок, що замкове РЗ під час значних осьових навантажень та дії вібрацій може входити в резонанс, що сприяє зменшенню напружень стиску в упорних торцях ніпеля і муфти та можливій втраті монолітності та герметичності різьби.

Спрощений гармонічний аналіз СЕМ РЗ гладких НКТ умовним діаметром 114 мм (ГОСТ 633-80) з пружною моделлю матеріалу виконано в праці [7]. За допомогою МСЕ знайдено власні частоти згвинченого вручну РЗ ($A=6,35$ мм) [7]: перша – 10627 Гц, друга – 14226 Гц, третя – 15156 Гц (11248 Гц, 14275 Гц, 15258 Гц – для згвинченого на верстаті з натягом $A=0$ мм). Значення переміщень під час власних коливань [7] свідчать про можливість роз'єднання контактних поверхонь ніпеля і муфти.

Проведено також гармонічний аналіз РЗ під час навантаження змінною зовнішньою силою, яка створює напруження розтягу в тілі НКТ 100 МПа в діапазоні частот 0-20 кГц [7]. Аналіз амплітуд деформацій частини РЗ в зоні торця муфти показав, що перша власна частота спричиняє значні амплітуди осьової деформації, а друга і третя – радіальної. Згвинчування РЗ з натягом майже не впливає на ці амплітуди. Проте амплітуда контактного тиску між витками змінюється суттєво. Згвинчування РЗ ($A=3,175$ мм) призводить до вирівнювання амплітуд контактного тиску на неробочих сторонах витків. З'єднання стає більш монолітним. Після згвинчування на лівій (робочій) стороні витків ніпеля

амплітуда контактного тиску значно зростає в зоні третьої власної частоти. Порівняно з замковим РЗ [325], найбільш небезпечною частотою з точки зору втомного руйнування РЗ в небезпечних перетинах (перша робоча впадина різьби ніпеля та остання робоча впадина різьби муфти) є тільки перша власна частота.

Більш ґрунтовний гармонічний аналіз РЗ виконано в праці [15]. В Abaqus/CAE 6.14 розроблена осесиметрична модель муфтового РЗ гладких НКТ умовним діаметром 114 мм за ГОСТ 633-80 (аналог 4-1/2 non-upset tubing coupling API Spec. 5CT). Розглянуто згвинчене вручну РЗ (з натягом $A = 6,5$ мм), як найбільш небезпечний випадок з точки зору герметичності. Спочатку були виконані два статичні загальні кроки аналізу (general analysis step) – згвинчування і осьове розтягування. Для імітації згвинчування зміщували ніпель відносно муфти на величину, кратну кроку різьби. В опціях "Interference Fit..." контактної взаємодії вибирали опцію "Gradually remove slave node overclosure during the step" (поступово зменшувати перекриття для підпорядкованих вузлів протягом кроку навантаження). На нижньому торці муфти (рис. 4.32) задавали граничну умову симетрії YSYMM. На верхній торець ніпеля діє постійний тиск $p = -155,1$ МПа, який імітує зовнішнє навантаження розтягу. Матеріал деталей – сталь 40 ($E=210$ ГПа, $\nu=0,28$, густина $7,8 \cdot 10^{-9}$ т/мм³). Застосовано пружно-пластичну модель матеріалу з ізотропним зміцненням (табл. 4.2). Між деталями моделювали контакт «поверхня-до-поверхні» (surface-to-surface contact) з малим ковзанням (small sliding). Коефіцієнт тертя між поверхнями f_m змінювали в межах 0,01..0,06. Розмір елементів сітки в зоні контакту 0,3 мм.

Таблиця 4.2 – Залежність напруження (σ) – пластична деформація (ε)

σ , МПа	314	349	384	419	454	489	525	560	595	813
ε	0	0,0013	0,0034	0,0067	0,0116	0,0188	0,0289	0,043	0,062	0,5978

В Abaqus/Standard найшвидше отримати реакції моделі на гармонічне збурення можна за допомогою процедур аналізу лінійного збурення (linear perturbation). В них реакція моделі є лінійною реакцією збурення відносно базового стану, який був досягнутий на попередньому загальному кроці аналізу.

Для врахування початкового напруженого стану, причиненого згинчунням і навантаженням p , потрібно включити опцію `nonlinear geometric effects` (`NLGEOM=YES`) в попередніх загальних кроках аналізу. Під час аналізу лінійного збурення будь-які нелінійності виключаються з моделі. Якщо задані умови контакту, то контактний стан поверхонь не змінюється протягом кроку такого аналізу. Тому цей аналіз не є точним, але потребує менше обчислювальних ресурсів. Спочатку були проведені такі види аналізу лінійного збурення Abaqus/Standard як розрахунок власних частот (`natural frequency extraction`) і стаціонарний динамічний аналіз прямим методом (`direct-solution steady-state dynamic analysis`). Ці аналізи передбачають, що вузли моделі отримують вільні або вимушені стаціонарні гармонічні коливання.

Для моделювання динаміки необхідно включити сили інерції та демпфування в рівняння рівноваги сил системи [184].

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F}, \quad (4.3)$$

де \mathbf{u} – вектор вузлових переміщень; $\dot{\mathbf{u}}$ – вектор вузлових швидкостей; $\ddot{\mathbf{u}}$ – вектор вузлових прискорень; \mathbf{M} , \mathbf{C} , \mathbf{K} – матриці мас, демпфування і жорсткості; \mathbf{F} – вектор зовнішніх сил.

Розрахунок власних частот (`natural frequency extraction`) – найбільш швидкий метод аналізу динаміки. Власні частоти можуть бути розраховані за рівнянням руху (4.3) без врахування зовнішніх навантажень ($\mathbf{F}=0$) і демпфування ($\mathbf{C}=0$). Якщо вектор переміщень подати у вигляді $\mathbf{u} = \boldsymbol{\varphi} \cos \omega t$, і підставити його у (4.3), то отримаємо проблему власних значень [184]

$$\omega_j^2 \mathbf{M} \boldsymbol{\varphi}_j = \mathbf{K} \boldsymbol{\varphi}_j,$$

де $\boldsymbol{\varphi}_j$ – власний вектор (форма коливань) для j -ї моди системи, ω_j – власна кутова частота системи (рад/с), t – час. Якщо СЕМ має n ступенів вільності, то вона має n власних значень ω_j^2 . Власні частоти можна обчислити і для конструкції з попередньо напруженим станом [184].

Для проведення аналізу потрібно ввести кількість перших власних частот або діапазон їх пошуку. В даному випадку був заданий діапазон 0-20 кГц. На рис. 4.32 показані результати цього аналізу – власні частоти (Гц) та відповідні форми вільних коливань (сумарне переміщення). Перша власна частота – 10957 Гц. Помітно, що багатьом цим частотам властиві форми коливань, за яких можлива розгерметизація різьби. Найбільш помітне значне взаємне радіальне переміщення ніпеля і муфти в верхній частині РЗ.

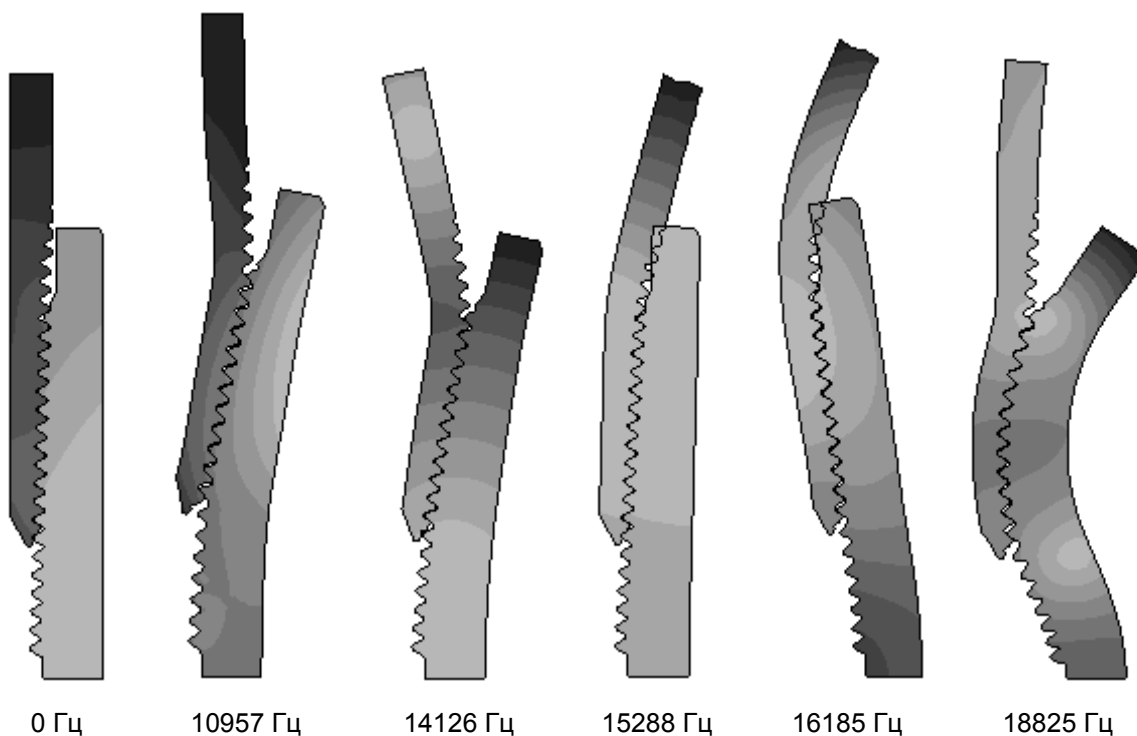


Рисунок 4.32 – Власні частоти згвинченого вручну РЗ, розраховані за допомогою аналізу Natural frequency extraction

Після цього проведено аналіз стаціонарного гармонічного відклику прямим методом (direct-solution steady-state harmonic response analysis). Для усталених гармонічних коливань усі вузли системи рухаються з однаковою частотою, але можуть мати зсув фази, що пояснюється наявністю демпфування. Таким чином вектор переміщень [184]

$$\mathbf{u} = \mathbf{u}_{\max} (\cos \varphi + i \sin \varphi) e^{i\omega t} = (\mathbf{u}_R + i\mathbf{u}_I) e^{i\omega t},$$

де \mathbf{u}_{\max} – вектор амплітуд переміщень, φ – зсув фаз для переміщень (рад), $\mathbf{u}_R = \mathbf{u}_{\max} \cos \varphi$ – вектор дійсних частин переміщень, $\mathbf{u}_I = \mathbf{u}_{\max} \sin \varphi$ – вектор уявних частин переміщень. Якщо аналогічно записати вектор зовнішніх гармонічних сил $\mathbf{F} = (\mathbf{F}_R + i\mathbf{F}_I)e^{i\omega t}$ і підставити \mathbf{u} і \mathbf{F} в (4.3), то отримаємо рівняння [184]

$$(\mathbf{K} - \omega^2 \mathbf{M} + i\omega \mathbf{C})(\mathbf{u}_R + i\mathbf{u}_I) = \mathbf{F}_R + i\mathbf{F}_I.$$

Це рівняння можна подати у вигляді $\mathbf{K}_c \mathbf{u}_c = \mathbf{F}_c$ і розв'язати за допомогою комплексної арифметики для заданого значення ω . Результатами гармонічного аналізу є амплітуди і фази відклику моделі на зовнішнє гармонічне збурення як функції частоти. Результати (переміщення, напруження, контактний тиск) повертаються в комплексній формі та можуть бути подані у вигляді дійсної X_R і уявної частини X_I , або фазового кута $\varphi = \arctg(X_I / X_R)$ і амплітуди $X_0 = \sqrt{X_R^2 + X_I^2}$. Значення величини в момент часу t може бути обчислене для заданої частоти ν за формулою $X = X_0 \cos(2\pi\nu \cdot t + \varphi)$ для кута $2\pi\nu \cdot t + \varphi$ від 0 до 2π рад. Цей тип аналізу є більш точний ніж попередній, але володіє більшою обчислювальною трудомісткістю. Аналіз дозволяє враховувати ефекти, які залежать від частоти (наприклад залежне від частоти демпфування). Для адекватної симуляції необхідні точні значення коефіцієнтів демпфування, які можна отримати тільки експериментально. Можна додати коефіцієнти демпфування Релея або структурного демпфування (structural damping factor), яке спричинене напруженим станом, та включити ефекти демпфування, які викликані тертям. В першому наближенні було ігноровано демпфування і розраховано тільки дійсний відклик (опція Compute real response only). Тобто в даному випадку $X_I = 0$ і $X_{t=0} = X_R$. Діапазон частот 10-20 кГц розбивався на 500 точок. Для кожного значення частоти проводився аналіз відклику моделі на зовнішнє гармонічне збурення. В якості зовнішнього гармонічного збурення був вибраний тиск, прикладений до верхнього торця ніпеля, амплітудою -10 МПа та середнім

значенням $-155,1$ МПа. В лістингу M.1 наведено частину Input-файлу Abaqus для відтворення гармонічного аналізу.

Незважаючи на неможливість розділення контактних поверхонь в такому аналізі, є можливість розрахунку значень контактного тиску. Результати (рис. 4.33) показують, що найбільш широкими діапазонами частоти з підвищеною амплітудою контактної тиску в середній частині РЗ є $10\text{-}12$ кГц і $18,5\text{-}19,2$ кГц. Форми коливань з екстремальними значеннями контактної тиску показані на рис. 4.34. Деформація на цих рис. збільшена для кращої візуалізації. Аналіз цих результатів дозволяє зробити висновки, що за певних частот можливе розкриття стиків РЗ і втрата ним герметичності. Згвинчене з натягом $A=3,175$ мм РЗ володіє іншими власними частотами ($11417, 14257, 15421, 16487, 19707$ Гц). Помітно, що більший момент згвинчування (менший натяг A) підвищує першу власну частоту (рис. 4.33). Ці результати можна використати для обґрунтувати границь допустимих експлуатаційних частот.

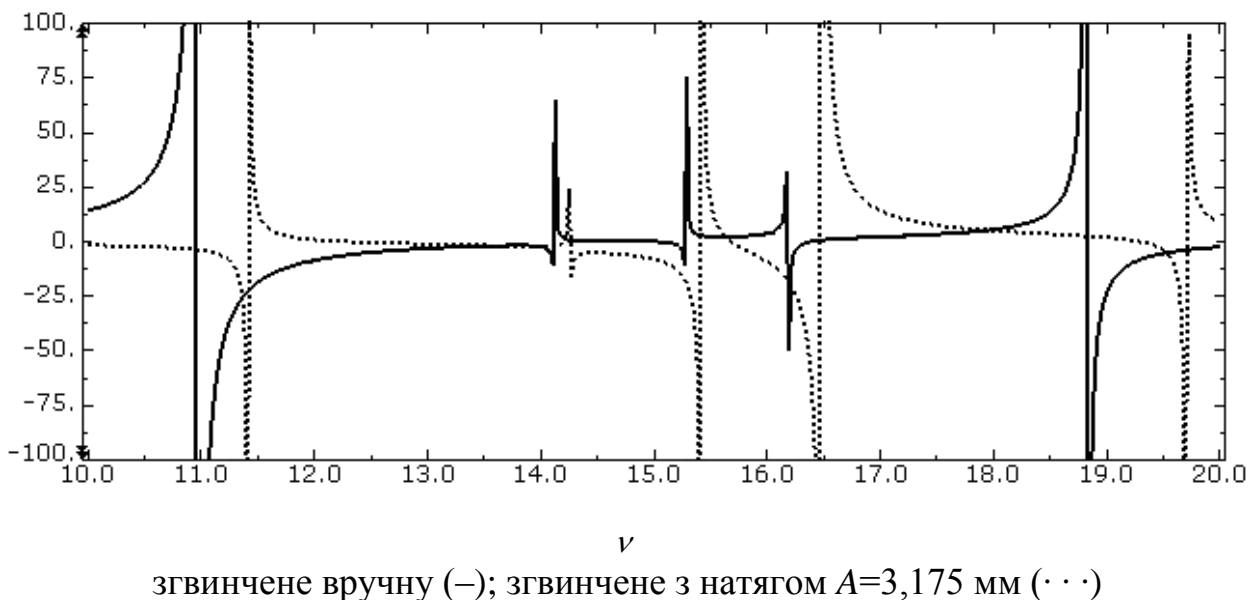


Рисунок 4.33 – Залежність дійсної частини контактної тиску (МПа) на робочій стороні профілю різьби в середній частині РЗ від частоти ν (кГц) зовнішнього гармонічного осьового навантаження

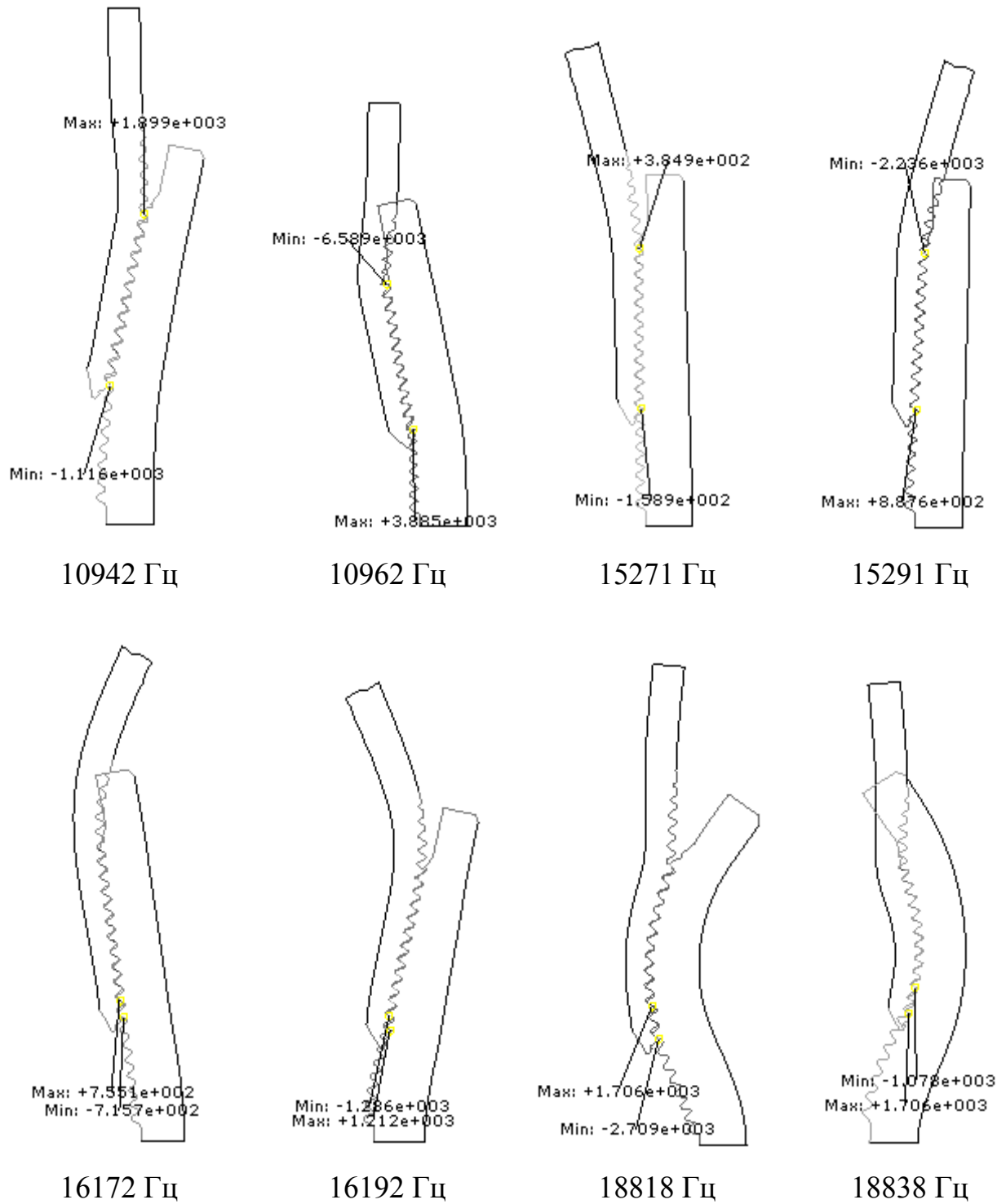


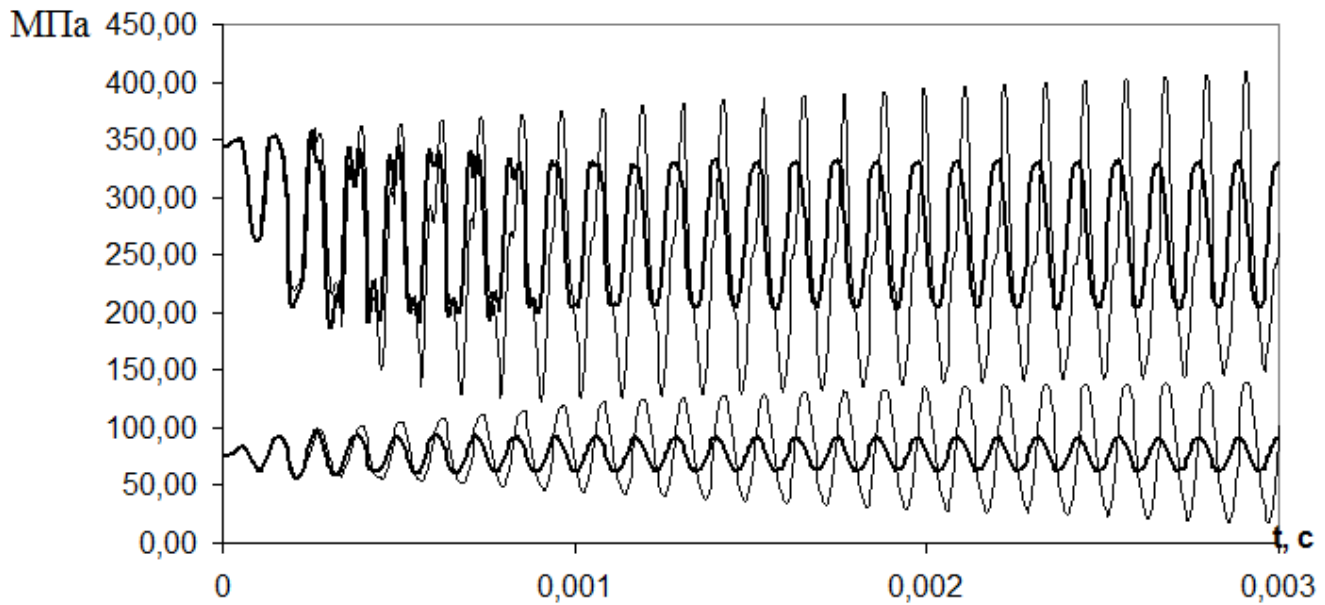
Рисунок 4.34 – Форми коливань та екстремальні значення дійсної частини контактної тиску (МПа), отримані за допомогою Direct-solution steady-state dynamic analysis

Наступним етапом досліджень є неявний динамічний аналіз РЗ з прямим інтегруванням (Implicit dynamic analysis using direct integration). Пряме інтегрування рівнянь руху виконується в Abaqus/Standard за допомогою процедури неявної динаміки. Ця процедура збирає матриці мас, демпфування та жорсткості та розв'язує рівняння (4.3) в кожній точці часу [329]. Цей аналіз дозволяє моделювати будь-які перехідні динамічні процеси, а не тільки стаціонарний гармонічний рух. Можна моделювати будь-які нелінійності та контакт поверхонь з можливістю їхнього розділення. Тому цей аналіз є найбільш точним, але потребує більше обчислювальних ресурсів. Тут дисипація енергії спричинюється тертям контактних поверхонь та пружно-пластичними властивостями матеріалу, які можуть призводити до його локального гістерезису. Тому не потрібно вводити яке-небудь додаткове демпфування. Зовнішнє навантаження являє собою тиск, який змінюється за законом $p = -155,1 - 10 \cdot \sin(2\pi vt)$ (МПа) і прикладений до верхнього торця ніпеля. Тут ν – частота (Гц), t – час гармонічного навантажування (с). Результати записувались з кроком часу $5 \cdot 10^{-6}$ с. Для уникнення небажаних початкових динамічних збурень слід відключити розрахунок прискорень на початку цього кроку опцією "Initial acceleration calculation at beginning of step: Bypass". В лістингу М.2 наведено частину Input-файлу Abaqus для відтворення неявного динамічного аналізу. Ця частина повинна бути розташована замість третього кроку в попередньому Input-файлі.

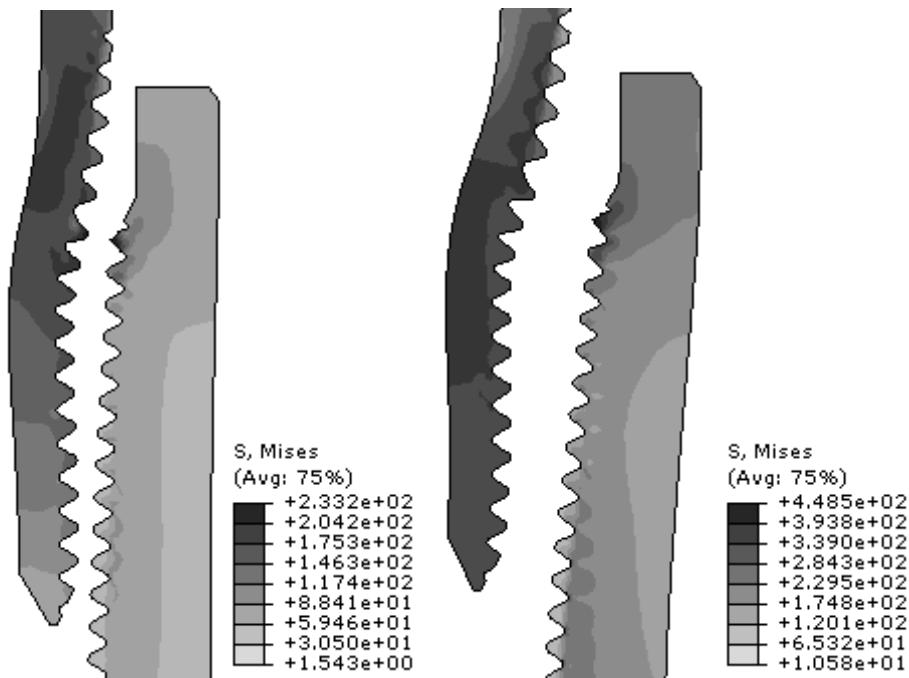
Результати цього аналізу можуть відрізнитись від попередніх. Зокрема під час навантажування частотою $\nu = 10957$ Гц в моделі не спостерігається резонансу, а є тільки незначне биття. Тому для пошуку резонансної частоти потрібно повторити цей аналіз для діапазону частот в околі власної частоти, отриманої попередніми типами аналізу. Таким методом була знайдена частота $\nu = 8754$ Гц, за

якої в моделі з $f_m=0,04$ відбувається резонанс (рис. 4.35). З залежності видно, що орієнтовно після 2..3 мс після початку гармонічного навантажування можливе розкриття контактних поверхонь в середній частині РЗ та повна втрата герметичності. Внаслідок розкриття контактних поверхонь імовірно і самовідгвинчування. Цікаво, що якщо коефіцієнт тертя збільшити до $f_m=0,06$, резонансні явища будуть слабше виражені (рис. 4.35). Зі зменшенням коефіцієнту тертя до 0,01 резонансні явища посилюються. Отже результати суттєво залежать від властивостей демпфування, зокрема коефіцієнта тертя в РЗ. Це підтверджує результати досліджень [330], в яких виявлено, що в РЗ демпфування вібрацій залежить від коефіцієнта тертя, пружних властивостей виробу, місця застосування змінного навантаження, характеру навантаження. Демпфування зростає зі збільшенням коефіцієнта тертя, але деколи сили тертя можуть збільшити вібрації, особливо під час моментального зняття навантаження [330].

Такий резонанс може призводити не тільки до втрати герметичності РЗ, але й до його втомного руйнування. Як свідчать промислові дані [98] і результати моделювання [7, 91-94], втомне руйнування такого РЗ найбільш імовірно в зоні першої робочої западини різьби ніпеля. На рис. 4.35а показано зміну еквівалентних напружень σ_m в цій зоні. Помітно, що під час резонансу вони швидко зростають і можуть призвести до втомного руйнування РЗ. На рис. 4.35б, в показані екстремальні деформації та напруження в РЗ впродовж циклу резонансного навантажування в моменти часу $t=2,955$ мс (початок циклу) і $t=3,03$ мс (кінець циклу). Деформації на рис. збільшені в 50 раз для кращої візуалізації. Помітно згин середньої частини ніпеля, який може спричинити розгерметизацію РЗ і втомне руйнування ніпеля.



а



б

в

а) – криві внизу – контактний тиск на робочій стороні профілю різьби в середній частині РЗ; криві вверху – значення σ_m в першій робочій западині різьби ніпеля;

$f_m=0,04$ (—); $f_m=0,06$ (---);

б, в) – розподіл σ_m (МПа) для $f_m=0,04$, $\nu=8754$ Гц

та часу симуляції $t=2,955$ мс (б), $t=3,03$ мс (в)

Рисунок 4.35 – Результати гармонічного осьового навантажування частотою 8754 Гц

Недорогим і ефективним засобом попередження відмов НКТ внаслідок резонансу може бути застосування систем автоматичного контролю за вібраційним станом НКТ. Найпростіша така система повинна містити сенсор амплітуди і частоти вібрацій НКТ та реле для виключення двигуна приводу. Як видно з рис. 4.35а, у такої системи буде запас часу на виключення двигуна приводу у разі значного зростання амплітуди.

Внаслідок складності адекватного опису властивостей демпфування отримані результати мають орієнтовний характер і можуть суттєво відрізнятись від поведінки реальних РЗ. Тому необхідні експериментальні дослідження на стендах з вібраційним навантаженням, які планується провести в майбутньому. Викладена методика скінченно-елементного аналізу може бути використана для визначення допустимих частот і амплітуд гармонічного навантаження за критеріями втомної міцності, герметичності та стійкості до самовідгвинчування. Параметричні моделі можуть використані для дослідження інших типорозмірів РЗ.

4.12 Висновки до розділу

1. Запропоновані принципи побудови та використання параметричних СЕМ РЗ на основі комерційних [196, 313-315] і вільних [187] CAD/FEA та розроблено параметричні СЕМ РЗ ШСНУ (муфтового РЗ ШН, замкового РЗ, двоопорного РЗ порожнистих ШН, муфтового РЗ НКТ), які є гнучкими для удосконалення і модифікації, володіють можливістю інтеграції в ІС, мають можливість автоматичної перебудови, симуляції та збереження результатів для різних значень параметрів. Принципи полягають у застосуванні розроблених автором програмних каркасів мовою Python для створення та використання елементів осесиметричних або тривимірних СЕМ РЗ та прикладних САПР РЗ. Моделі дозволяють виконувати ґрунтовний різносторонній аналіз і оптимізацію РЗ, підвищити продуктивність і якість проектування нових їхніх типів. Результати досліджень можуть бути використані для обґрунтування вибору геометричних

параметрів, матеріалів, моменту згвинчування цих РЗ за критеріями втомної міцності та герметичності.

2. Розроблена система на основі FreeCAD геометричного моделювання різьб з відхиленнями (у тому числі технологічними) [8, 13] може бути використана для побудови 3D та 2D геометричних моделей РЗ з відхиленнями, обґрунтування їхніх допусків і оптимізації геометричних параметрів з додатковим програмним забезпеченням для скінченно-елементного аналізу.

3. На основі СЕМ досліджено вплив геометричних параметрів РЗ ШН на розподіл еквівалентних напружень у западинах витків різьби ніпеля [196, 320]. Виявлені залежності еквівалентних напружень від величини радіусів округлення зарізьбової канавки РЗ ШН [320]. Вибір оптимальних геометричних параметрів муфтового РЗ ШН дозволяє зменшити величину еквівалентних напружень в небезпечних зонах на 10...20% [314, 315]. Зі збільшенням границі плинності сталі ніпеля напруження в небезпечних зонах РЗ зростають [12]. Розроблено методику та отримано залежності для визначення оптимальної величини згвинчування РЗ ШН [198, 323]. Для підвищення втомної міцності РЗ необхідним є збільшення довжини зарізьбової канавки мінімум на 20 мм, що вимагає збільшення оптимальної величини згвинчування [19, 20] з 0,15 до 0,25..0,3 мм [198, 323].

4. Виявлено, що від'ємні значення відхилень кута профілю різьби муфти ШН дещо зменшують значення напружень і збільшують значення D в першій навантаженій западині різьби ніпеля [8]. Контактні тиски зростають в зоні мінімального діаметра різьби муфти. Додатні значення, навпаки, різко зменшують значення D в першій западині різьби ніпеля, але майже не змінюють еквівалентні напруження в ній.

5. Додатні значення величини зменшення кроку різьби муфти ШН на кожному витку (починаючи з першого) є причиною різкого зменшення значень D внаслідок нерівномірного навантаження на витки [8]. Від'ємні значення, навпаки, вирівнюють навантаження в різьбі, дещо зменшуючи напруження в першій навантаженій западині ніпеля і дещо зменшують область з від'ємними значеннями D .

6. Виявлені регресійні залежності контактних тисків в різьбі ШН від максимального діаметра різьби ніпеля d та номера витка різьби N [11]. Залежності можуть бути використані для обґрунтування допустимих значень d . З залежностей для з'єднання 19 мм ШН виявлено, що менші за 26,2 мм значення d є причиною різкого збільшення контактних тисків і різкого зменшення коефіцієнта запасу втомної міцності D .

7. Отримано залежності для визначення оптимальної величини згвинчування Δ замкового РЗ, яке може бути використане як альтернатива стандартному для з'єднання порожнистих ШН. Збільшення Δ і застосування муфти з пластичнішого матеріалу дещо вирівнює напруження у впадинах різьби ніпеля, але й майже не змінює їх у перших впадинах [315]. Виявлено, що зменшення границі плинності сталі муфти з 500 до 300 МПа вимагає збільшення оптимальної величини згвинчування Δ з 0,16 до 0,23 мм та призводить до підвищення втомної міцності РЗ. Виявлено, що у порівнянні зі стандартним РЗ ШН замкове РЗ володіє меншою циклічною довговічністю внаслідок відсутності зарізьбової канавки.

8. Заснована на параметричному геометричному моделюванні та МСЕ ітеративна методика проектування дозволила підтвердити доцільність застосування двоопорних РЗ для порожнистих ШН та удосконалити їхню конструкцію [9, 14]. Завдяки двом опорам, модифікованим зарізьбовим канавкам з еліптичним профілем, корекції першого витка різьби ніпеля та оптимізації натягів удосконалене двоопорне РЗ володіє більшою міцністю під час згину, кручення та стиску, вищим опором до самовідгвинчування та герметичністю, більш рівномірним розподілом навантаження на витки та більшою втомною міцністю у порівнянні зі стандартним. Коефіцієнт запасу D збільшено в зоні першого витка ніпеля на 13..14, в зоні останнього витка муфти на 2,6..4, в зоні зарізьбової канавки ніпеля на 0..0,4. Застосування пружного елемента на додатковій опорі дозволило збільшити допуск натягу на ній до 0,2 мм, що робить непотрібним контроль натягу перед згвинчуванням. Навіть після пошкодження пружної частини та втраті натягу на додатковій опорі РЗ буде міцніше за стандартне. У

разі ремонту ШН удосконалені ніпелі можна легко впровадити шляхом приварювання їх до тіла ШН. Існує також можливість впровадження двоопорних циліндричних РЗ без значної модифікації стандартного ніпеля.

9. Виконано аналіз напружено-деформованого стану СЕМ РЗ НКТ, згвинченого з різними натягами A [7, 196]. Отримано залежності контактних тисків в різьбі муфтового РЗ НКТ від величини натягу для різних значень зовнішнього навантаження розтягу. Запропоновано методика аналізу і залежності можуть бути використані для визначення оптимального натягу за критерієм максимальної герметичності.

10. Обґрунтовано можливість застосування РЗ НКТ, різьба ніпеля яких має відхилення кута профілю внаслідок нарізання некоригованим різцем з від'ємним переднім кутом -10° [10]. Застосування такого кута підвищує технологічність обробки високоміцних сталей. Втомна міцність такого РЗ майже не змінилася, але для запобігання появи зазорів в різьбі різьба муфти повинна бути нарізана з врахуванням різниці кутів профілю ніпеля і муфти.

11. Запропонована методика аналізу і проаналізовано відклик СЕМ згвинченого вручну муфтового РЗ НКТ умовним діаметром 114 мм за ГОСТ 633-80 на гармонічне зовнішнє осьове навантаження частотами 0..20 кГц. Наближений аналіз РЗ за допомогою процедур лінійного збурення Abaqus/Standard дозволив отримати орієнтовні значення власних частот в діапазоні 0..20 кГц [7] та значення контактного тиску на робочій стороні профілю в середній частині РЗ як функцію частоти. Уточнений неявний динамічний аналіз РЗ з прямим інтегруванням дозволив врахувати нелінійну поведінку матеріалу і тертя контактних поверхонь під час гармонічного осьового навантажування [15]. Аналіз показав, що зменшення коефіцієнта тертя між поверхнями контакту до 0,01..0,04 може підвищити схильність РЗ до резонансу в околі частоти 8754 Гц. Результати показують можливість порушення герметичності та втомного руйнування РЗ внаслідок резонансу за певних умов та можуть бути використані для обґрунтування допустимих границь частот експлуатаційного гармонічного навантаження.

РОЗДІЛ 5

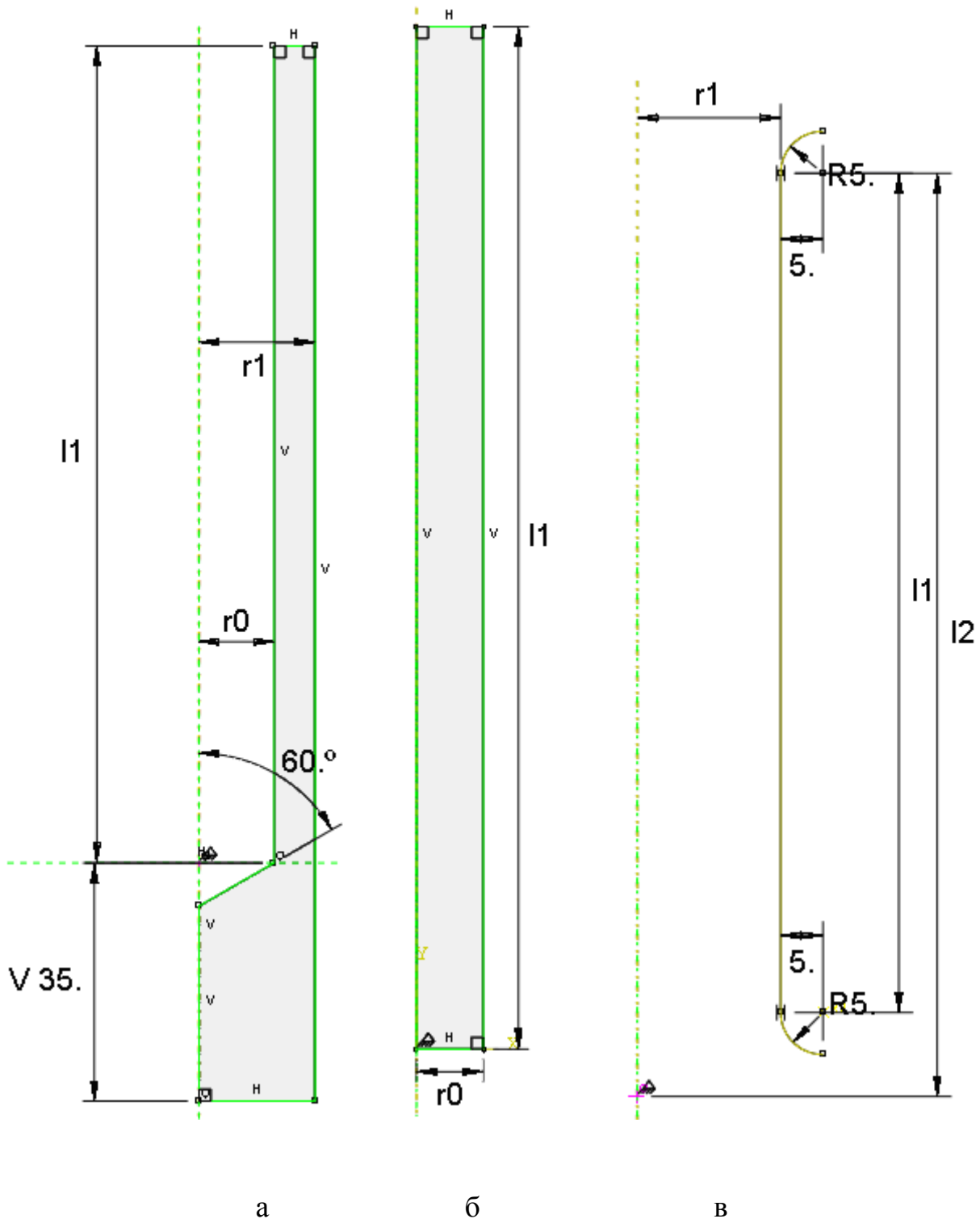
СКІНЧЕННО-ЕЛЕМЕНТНІ МОДЕЛІ ЕЛЕМЕНТІВ КОЛОНИ ШТАНГ ТА НКТ

5.1 Моделі пресового з'єднання тіла склопластикової насосної штанги зі сталевую головкою

5.1.1 Принципи побудови моделі на основі Abaqus/CAE

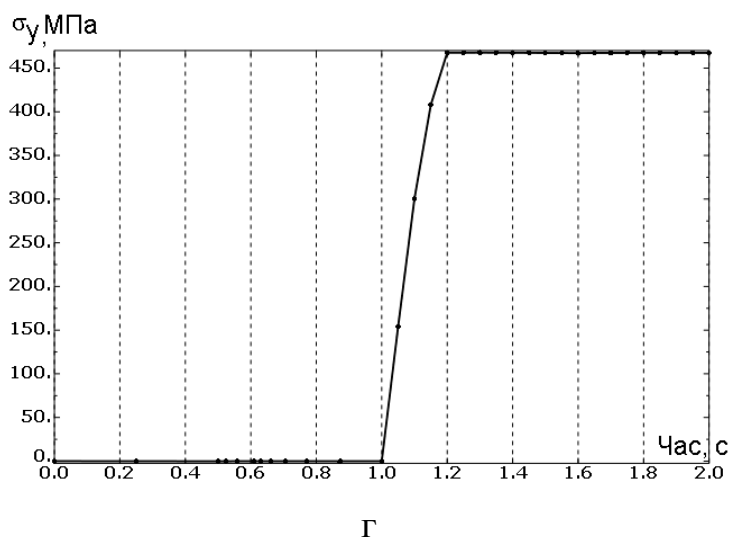
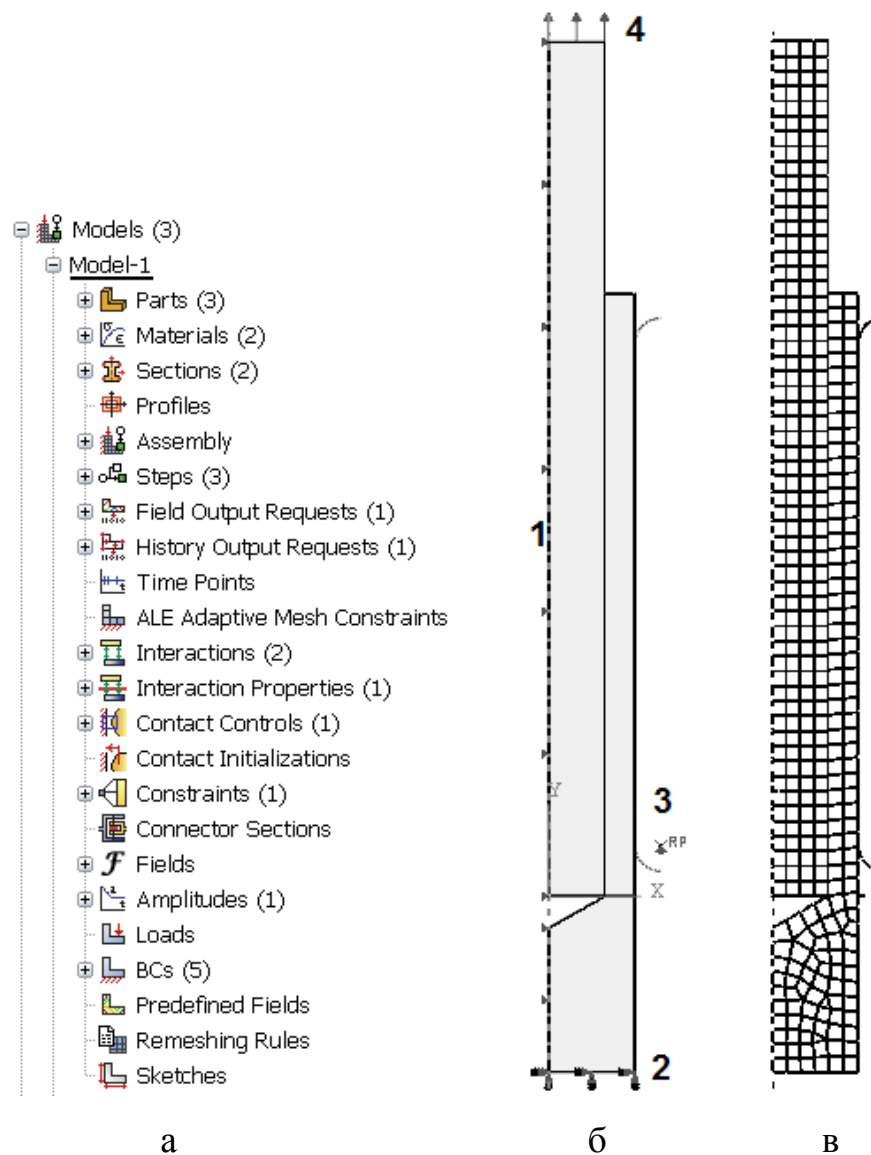
Для оптимізації різнотипних пресових з'єднань (рис. 0.1) автором розроблено параметричну осесиметричну СЕМ з'єднання в системі Abaqus/CAE 6.10, яка дозволяє моделювати пластичність сталі головки, ортотропію механічних характеристик склопластику, контакт між тілом і головкою, процес обтискання головки жорсткими штампами довільної форми (рис. 5.1) [16, 331]. Тип задачі – нелінійний статичний аналіз. Геометричні параметри з'єднання: діаметр склопластикового тіла – 22 мм, зовнішній діаметр головки – 34 мм, довжина обтискання (довжина штампа) $l_f=100...180$ мм. Матеріал головки ШН – сталь з такими механічними характеристиками: $E=2,1 \cdot 10^{11}$ Па, $\nu=0,28$. Характеристики пластичності (ГОСТ 13877-96) вводяться в Abaqus® у вигляді пластичної ділянки істинної діаграми деформування (ділянка $\sigma_m-\sigma_\epsilon$), яка задавалась у вигляді степеневої залежності. Характеристики матеріалу склопластикового тіла: модуль пружності в осьовому напрямку $E_y=0,5 \cdot 10^{11}$ Па, в радіальному напрямку $E_x=0,1 \cdot 10^{11}$ Па, коефіцієнт Пуассона $\nu_{xy}=0,22$. Коефіцієнт тертя між поверхнями контакту $f=0,1$.

Симуляція проводилась в два кроки навантаження. На першому задавалась глибина радіального переміщення штампів $\Delta = 0,1...0,3$ мм (або тиск обтискання головки $p=300...600$ МПа). На другому кроці дія штампів (тиску обтискання) усувалась, а на торці склопластикового тіла задавалась гранична умова осьового переміщення Δ_y . Другий крок розбивався на підкроки (фрейми), на кожному з яких переміщення Δ_y поступово збільшувалось. Фрейм з максимальним значенням осьового напруження σ_y в тілі ШН відповідає моменту руйнування з'єднання (рис. 5.2).



а) – сталева головка, б) – склопластикове тіло, в) – жорсткий штамп

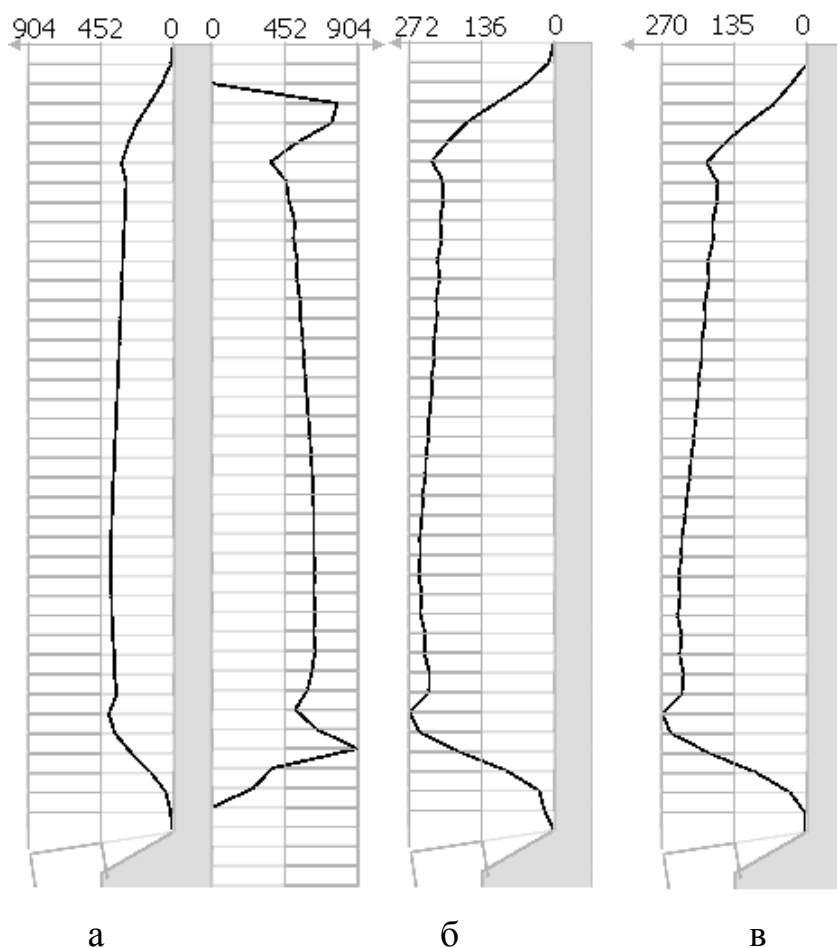
Рисунок 5.1 – Геометричні моделі деталей



а) – дерево побудови моделі; б) – граничні умови (1 – $U_x=0$; 2 – $U_x=0, U_y=0$; 3 – $U_x=\Delta$; 4 – $U_y=\Delta y$); в) – сітка елементів; г) – залежність σ_y від часу симуляції

Рисунок 5.2 – СЕМ пресового з'єднання та результати симуляції

На рис. 5.3 показано розподіл контактних тисків в з'єднанні ($\Delta=0,25$ мм, $l_I=100$ мм, $\sigma_m=400$ МПа) в різні моменти часу симуляції [331]. Ці дані відповідають результатам, отриманим автором раніше за допомогою програмного комплексу Ansys® [1]. Помітно, що контактний тиск на внутрішній поверхні головки не розподіляється рівномірно по довжині контакту. В місцях його максимуму можливе розтріскування поверхневого шару склопластикового стержня і руйнування по тілу [1]. Тому під час обтискання важливо не перевищити величину допустимого контактного тиску, яку слід визначати експериментально. Засобом боротьби з проникненням середовища в з'єднання може бути нанесення клею на поверхні контакту безпосередньо перед обтисканням.



а) – під час обтискання штампами; б) – після обтискання; в) – в момент руйнування

Рисунок 5.3 – Розподіл контактного тиску (МПа) на внутрішній (крива зліва) та зовнішній (крива справа) поверхнях сталеві головки для $\Delta=0,25$ мм

На рис. 5.4 показано розподіл контактних тисків в з'єднанні порожнистої ШН (зовнішній діаметр тіла 22 мм, внутрішній – 12 мм, $l_1=100$ мм) в різні моменти часу симуляції. Обтискання моделювали на першому кроці тиском 500 МПа на зовнішній поверхні головки. З'єднання з несучільною головкою (з окремою сталевую циліндричною вставкою в отворі ШН) витримує $\sigma_y=420$ МПа, а з'єднання з суцільною головкою – $\sigma_y=590$ МПа. Помітно, що контактний тиск на поверхні вставки більший ніж на поверхні отвору. Характер розподілу контактного тиску відповідає результатам моделювання з'єднання з суцільним тілом. В момент руйнування контактні тиски в верхній частині зменшуються.

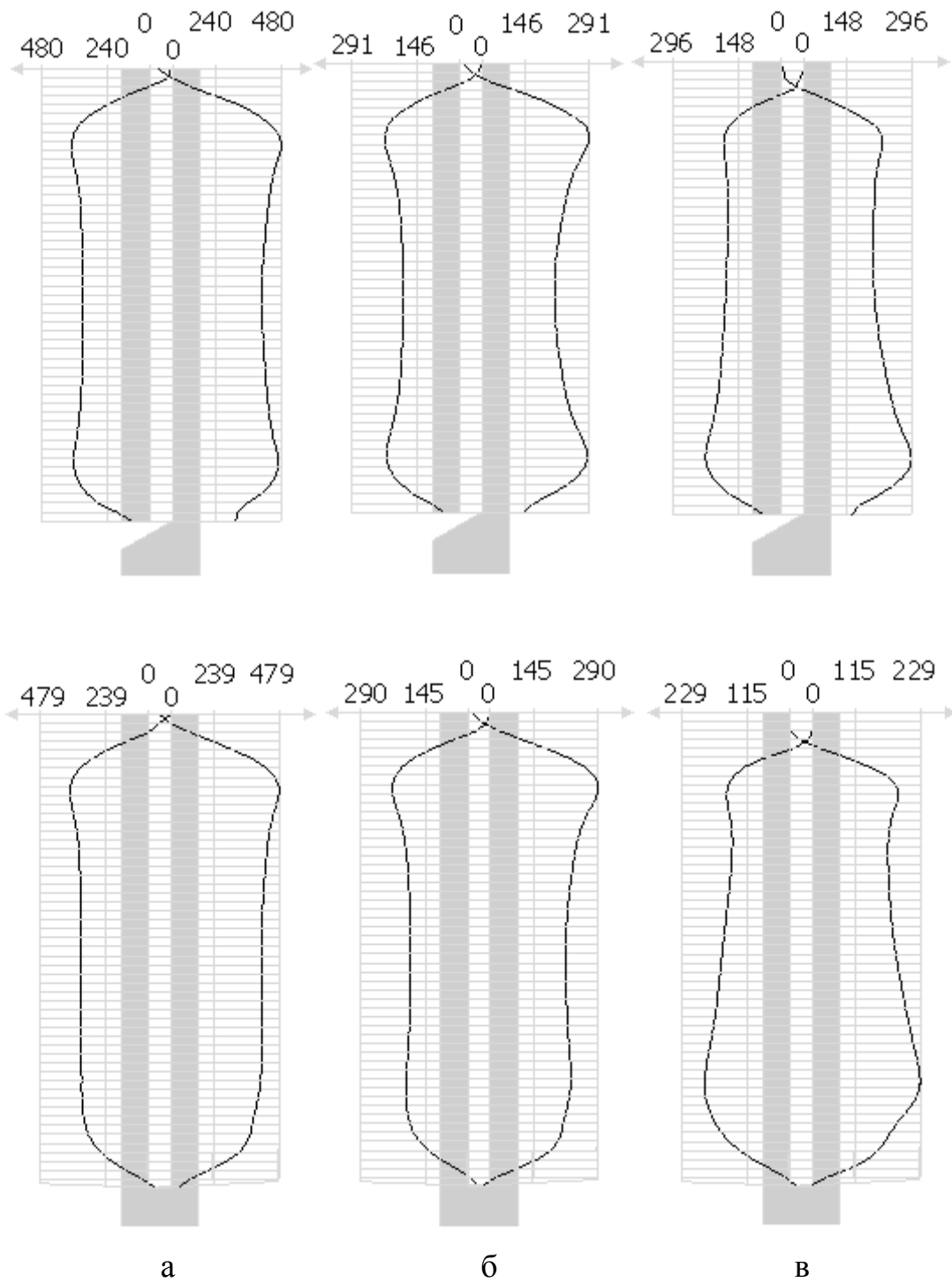
Для автоматизації перебудови моделі із заданими параметрами та отримання результатів розрахунку використовувалась програма-сценарій мовою Python (лістинг П.1). Параметрична СЕМ з'єднання разом з програмою-сценарієм являє собою прикладну САПР побудовану на базі Abaqus® CAE [331]. Алгоритм програми містить кілька вкладених циклів. Для прикладу, в зовнішньому циклі змінюється значення глибини переміщення штампів Δ (або тиску обтискання), у внутрішньому – границя плинності сталі σ_m , довжина обтискання l_1 або інший параметр з'єднання. В тілі внутрішнього циклу перебудовується модель, виконується розрахунок, визначається контактний тиск, осьове напруження руйнування з'єднання і дані записуються у файл. Таким чином автоматизовано отримуються залежності руйнуючого напруження $\sigma_{руйн}$ від глибини переміщення штампів Δ та інших параметрів з'єднання. Для спрощення програмування автором розроблено клас `Material` та функції `set_values`, `mesh_all`, `JobSubmit`, `readODB_set2`, `findmax`. Нижче наведено основну частину алгоритму на псевдокоді.

```
FOR EACH  $\Delta$  IN (0.1, 0.15, 0.2, 0.25, 0.3):
```

```
  FOR EACH  $l_1$  IN (100.0, 140.0, 180.0):
```

```
    FOR EACH  $\sigma_m$  IN (300.0, 400.0, 500.0):
```

```
      установити значення геометричних параметрів, оновити геометрію,
      змінити властивості матеріалу, створити сітку, змінити граничні
      умови, виконати задачу, відкрити базу даних результатів, прочитати
      вектор напружень і знайти максимальне, прочитати вектор контактних
      тисків і знайти максимальне, записати дані у файл, закрити базу
      даних результатів.
```



а) – під час обтискання; б) – після обтискання; в) – в момент руйнування;

верхній ряд – несучільна головка; нижній ряд – суцільна;

криві ліворуч – поверхня отвору; криві праворуч – поверхня вставки

Рисунок 5.4 – Розподіл контактного тиску (МПа) на контактних поверхнях сталевій головки з'єднання порожнистих ШН

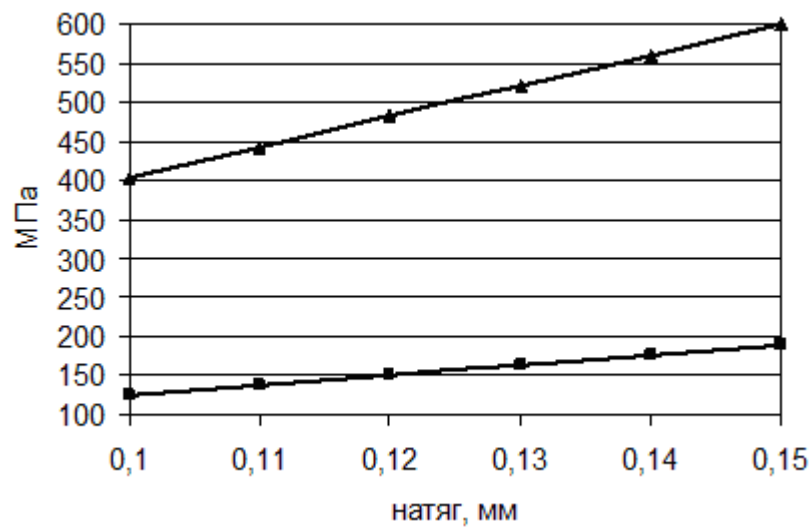
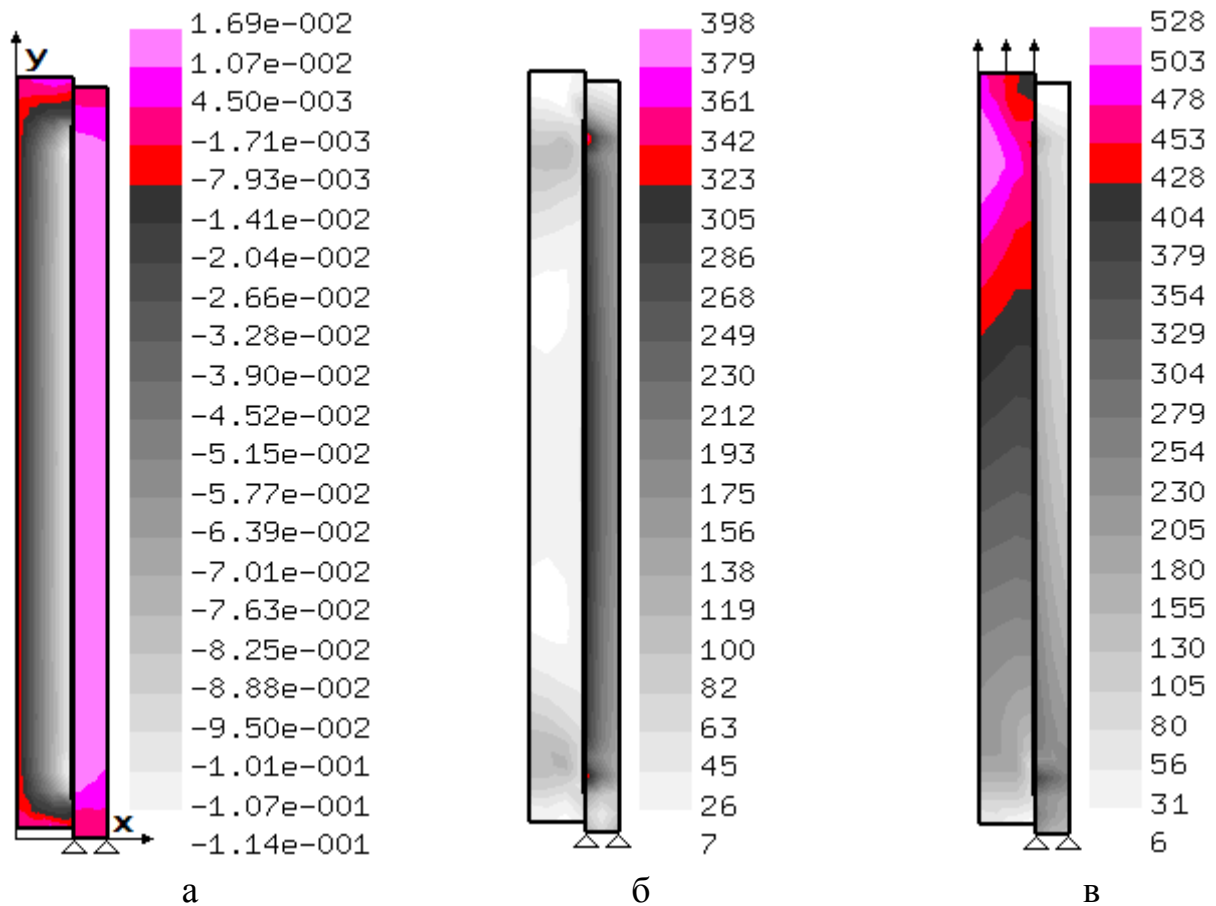
5.1.2 Принципи побудови моделі на основі вільного програмного забезпечення

Метою є створення САПР металополімерних з'єднань на основі вільного програмного забезпечення – мови програмування Python 2.7 та FEA-системи CalculiX 2.12 [186]. Розглянемо побудову осесиметричної СЕМ пресового з'єднання склопластикового тіла ШН зі сталевим ніпелем. Базовим компонентом розробленої САПР [18, 117] є програма мовою Python (лістинг Р.1), яка створює геометричні параметри з'єднання, будує регулярну сітку елементів на прямокутних областях з'єднання та формує вхідний файл для розв'язувача CalculiX. Для побудови сітки необхідно ввести кількість елементів по ширині та висоті прямокутних областей. Після створення вузлів і елементів програма створює контактну поверхню ніпеля. Її форма може бути задана за допомогою $dx(y)$ – залежності радіального зміщення вузла внутрішньої поверхні ніпеля від осьової координати точки y . Величина зміщення повинна бути меншою розміру елемента. Після цього програма створює вхідний файл для розв'язувача CalculiX. Структура такого файлу показана в табл. Р.1. У файлі описані два кроки навантаження – крок утворення з'єднання з натягом та крок поступового осьового переміщення верхнього торця тіла. Задача виконується в CalculiX шляхом створення окремого процесу. У випадку успішного завершення задачі програма читає результати з файлу `jobname.dat` шляхом його синтаксичного аналізу. Основні результати – максимальне радіальне напруження в тілі в кінці першого кроку та максимальне впродовж другого кроку осьове напруження на торці тіла. Перше повинно бути меншим максимального допустимого радіального напруження в тілі, а друге визначає максимальне осьове навантаження, яке витримує з'єднання.

Опишемо параметри моделі з'єднання. Діаметр стержня 22 мм, зовнішній діаметр ніпеля 36 мм, довжина ШН 150 мм, довжина ніпеля 150 мм, границі вертикальної зони деформування 10...140 мм. Пружна модель матеріалу ніпеля відповідає сталі 40 ($E=210$ ГПа, $\nu=0,28$). Пружна модель матеріалу стержня володіє наступними характеристиками: модуль пружності в осьовому напрямку $E_y=0,5 \cdot 10^5$ МПа, в радіальному напрямку $E_x=0,1 \cdot 10^5$ МПа, коефіцієнт Пуассона $\nu_{xy}=0,22$. Між деталями моделювався контакт «поверхня-до-поверхні» (Surface-to-surface) зі скінченним ковзанням (Finite Sliding) та коефіцієнтом тертя 0,2. Параметр контактної взаємодії ADJUST дозволяє моделювати натяг на першому кроці навантажування. Функція зміщень $dx(y)=0,12$ утворює постійний вздовж осі натяг величиною 0,12 мм. Переміщення і еквівалентні напруження в такому з'єднанні показані на рис. 5.5. Центральна частина з'єднання є послабленою (рис. 5.5б), що вимагає застосування хвилястих вздовж штампів. В момент руйнування найбільші напруження виникають вверху тіла і внизу головки (рис. 5.5в). Розраховані напруження на торці в момент руйнування з'єднання рівні 482 МПа.

Програма може автоматично виконувати послідовність задач для різних значень параметрів з'єднання шляхом циклічного повтору команд побудови моделі, розрахунку і запису результатів. Таким чином можна виконувати пошук оптимальних параметрів з'єднання. Так були отримані залежності осьового напруження руйнування з'єднання та радіального напруження в тілі в кінці першого кроку від величини натягу (рис. 5.5г). Вони дозволили знайти оптимальне значення натягу (0,12 мм), для якого значення радіальних напружень в тілі не перевищують допустимих (орієнтовно 150 МПа [1]).

Розроблена САПР дозволить проводити ґрунтовний різносторонній аналіз таких з'єднань та виконувати їхню оптимізацію. Наступним етапом досліджень є пошук оптимальної функції радіальних зміщень $dx(y)$ та експериментальна перевірка результатів моделювання.



г

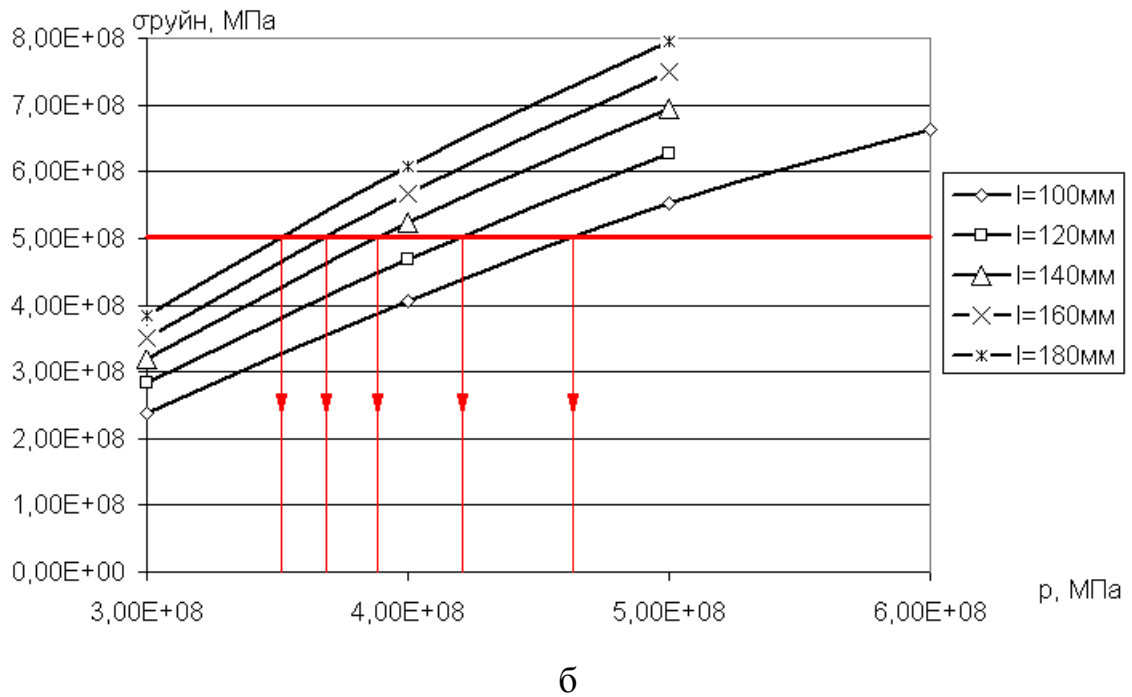
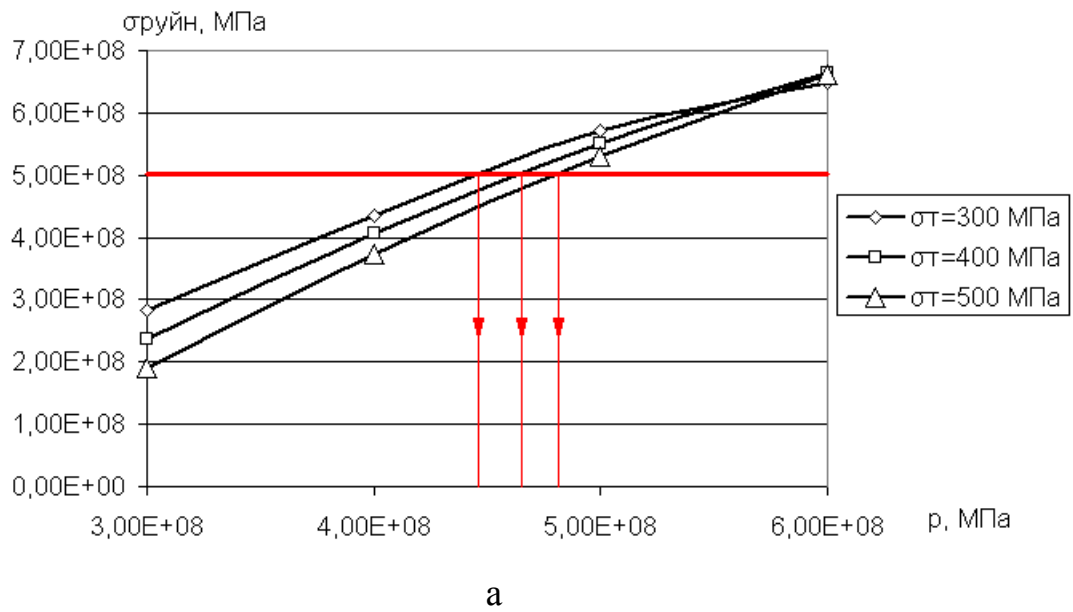
а) – радіальні (X) переміщення (мм); б, в) – напруження σ_m (МПа); а, б) – без зовнішнього навантаження, в) – в момент руйнування з'єднання; г) – залежність осьового напруження руйнування з'єднання (▲) та радіального напруження в тілі в кінці першого кроку (■) від величини натягу

Рисунок 5.5 – Результати симуляції пресового з'єднання в CalculiX

5.1.3 Залежності міцності з'єднання тіла склопластикової насосної штанги зі сталевією головкою від його параметрів

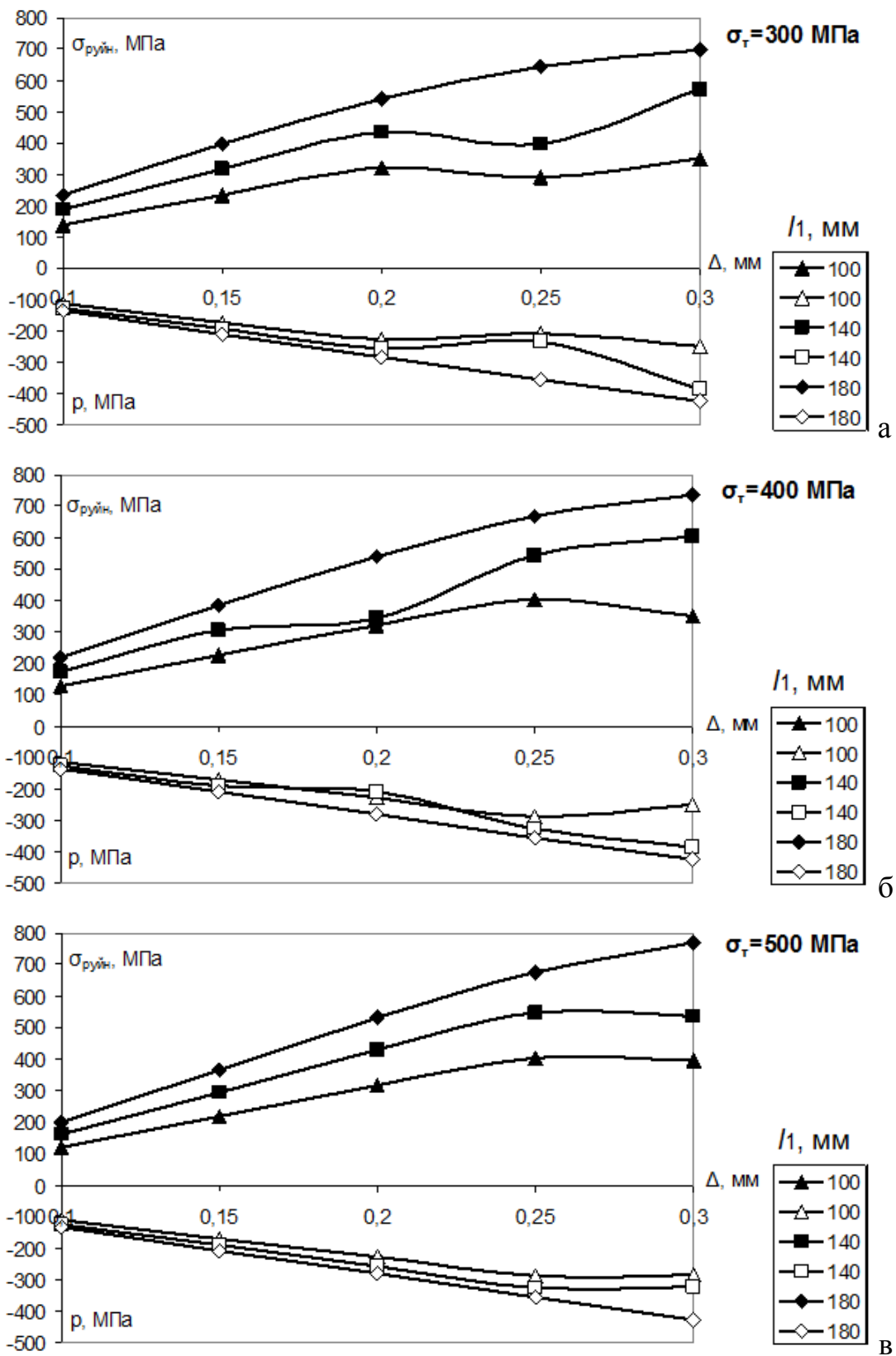
Нижче (рис. 5.6) наведено результати моделювання з'єднання, у якого обтискання моделюється радіальним тиском p , що діє на зовнішню циліндричну поверхню ніпеля. Геометричні параметри з'єднання: діаметр склопластикового тіла – 22 мм, зовнішній діаметр головки – 34 мм, довжина обтискання $l=100$ мм. Аналіз таких залежностей дозволяє вибрати оптимальні параметри з'єднання – тиск обтискання, границю плинності сталі, довжину обтискання. Зокрема помітно, що для границі міцності тіла $\sigma_e=500$ МПа тиски обтискання слід вибирати не меншими 445, 460, 475 МПа для σ_m 300, 400, 500 МПа відповідно (рис. 5.6а). Для з'єднання з $\sigma_m=400$ МПа тиски обтискання слід вибирати не меншими 350, 365, 387, 420, 460 МПа для довжини обтискання l 180, 160, 140, 120, 100 мм відповідно (рис. 5.6б). Верхня межа тиску обтискання обмежується міцністю склопластикового тіла під час обтискання в радіальному напрямку.

Нижче (рис. 5.7) наведено результати моделювання з'єднання, у якого обтискання моделюється радіальним переміщенням Δ жорстких штампів [331]. Цей спосіб моделювання більш реалістичний і дозволяє моделювання формованих штампів, але володіє вищою обчислювальною трудомісткістю. Аналіз таких залежностей дозволяє вибрати оптимальні параметри з'єднання – глибину переміщення штампів, границю плинності сталі, довжину обтискання. Наприклад, для заданої міцності з'єднання $\sigma_{руйн}=400$ МПа глибину переміщення штампів потрібно вибирати з табл. 5.1.



а) – змінна σ_m ($l=100$ мм); б) – змінна довжина обтискання l ($\sigma_m=400$ МПа)

Рисунок 5.6 – Залежність $\sigma_{руйн}$ від тиску обтискання головки p



а) – $\sigma_m=300$ МПа; б) – $\sigma_m=400$ МПа; в) – $\sigma_m=500$ МПа

Рисунок 5.7 – Залежність $\sigma_{руйн}$ (темні точки) і середнього контактного тиску в з'єднанні під час обтискання p (світлі точки) від Δ для різних значень l_1

Таблиця 5.1 – Глибина переміщення штампів Δ та контактний тиск в з'єднанні p під час обтискання для з'єднання з $\sigma_{руїн}=400$ МПа

Границя плинності сталі σ_m , МПа	Довжина обтискання l_l , мм					
	100		140		180	
	Δ , мм	p , МПа	Δ , мм	p , МПа	Δ , мм	p , МПа
300	-	-	0,18	230	0,15	160
400	0,25	290	0,22	230	0,16	180
500	0,25	300	0,19	250	0,16	210

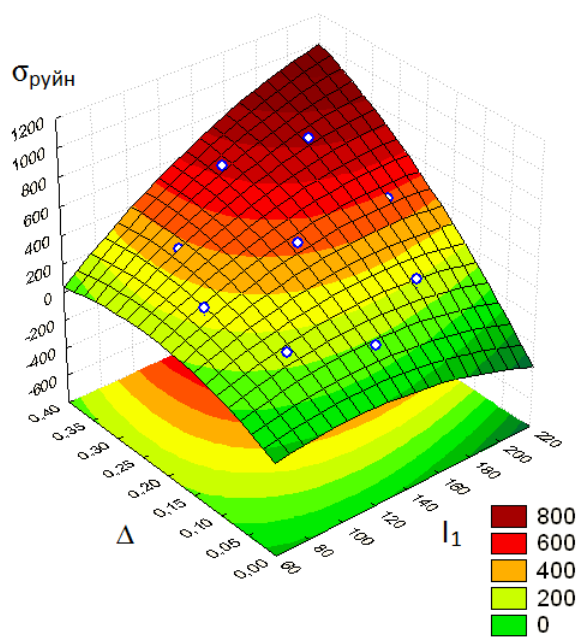
За результатами моделювання отримано відповідні регресійні залежності напруження руйнування $\sigma_{руїн}$ від Δ , довжини обтискання l_1 і зовнішнього радіуса сталеві головки r_1 (рис. 5.8):

$$\sigma_{руїн} = -341,792 + 4,619 \cdot l_1 - 0,0215 \cdot l_1^2 + 984,23 \cdot \Delta - 4151,397 \cdot \Delta^2 + 18,59 \cdot l_1 \Delta,$$

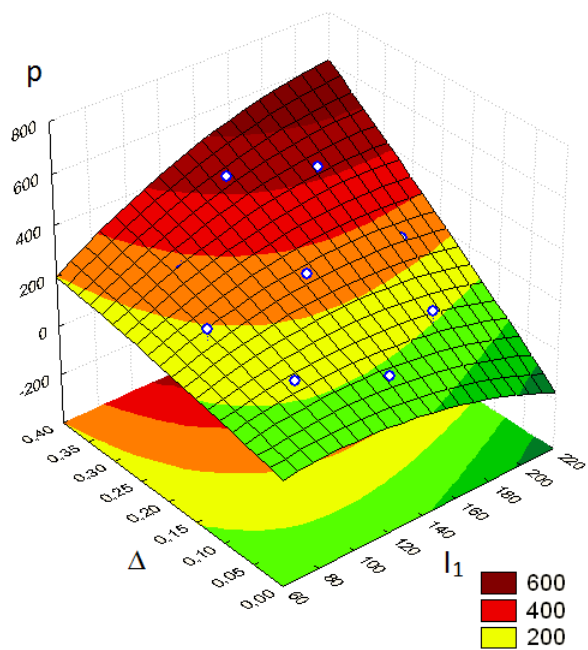
$$\sigma_{руїн} = -569,579 + 77,203 \cdot r_1 - 2,9285 \cdot r_1^2 - 358,657 \cdot \Delta - 3122,284 \cdot \Delta^2 + 213,258 \cdot r_1 \Delta.$$

Ці залежності можуть бути використані для вибору значень глибини переміщення штампів, довжини обтискання і зовнішнього радіуса сталеві головки.

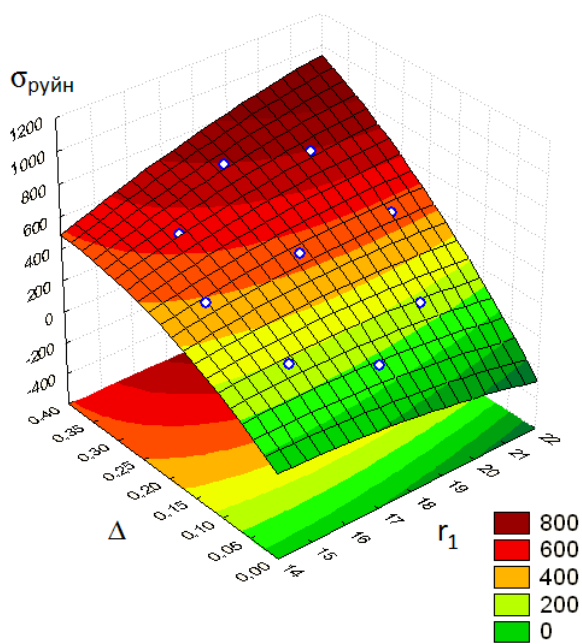
В дійсності обтискання виконуються не рівномірно, а шістьма парами плоских штампів з різних сторін головки (рис. 5.9). На рис. цифрами 1-6 позначені пари штампів. Тому для перевірки адекватності осесиметричних моделей була виконана симуляція тривимірних моделі. Моделювали одночасне радіальне переміщення шести пар плоских штампів (рис. 5.9а) і послідовне обтискання парою штампів в шести різних точках кола (рис. 5.9в). Помітно, що в моделях з одночасним і послідовним обтисканням в центральній частині з'єднання утворюються зазори, що негативно відбиваються на міцності з'єднання. Тому пропонується обтискати з'єднання формованими штампами (хвилястими вздовж), або виконувати хвилястою вздовж внутрішню поверхню головки. Крім того видно, що послідовне обтискання не забезпечує однаковість контактних тисків в коловому напрямку внаслідок пружної деформації – головка набуває еліптичної форми. У зв'язку з цим слід віддавати перевагу одночасному обтисканню.



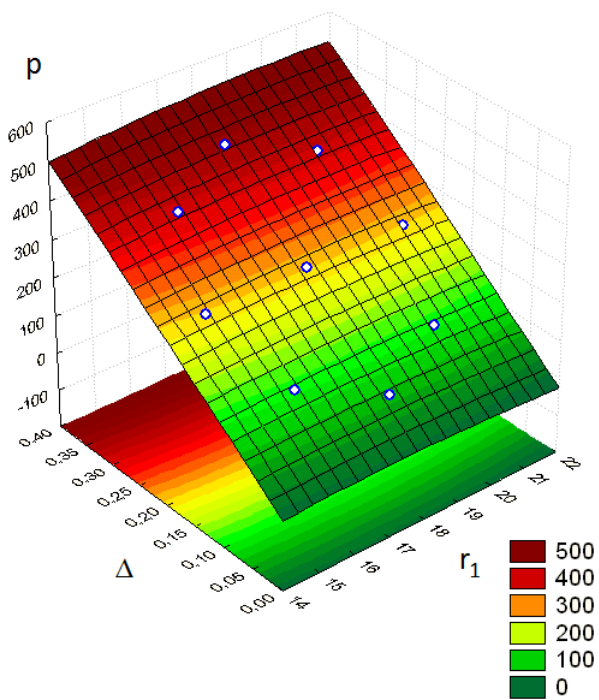
а



б



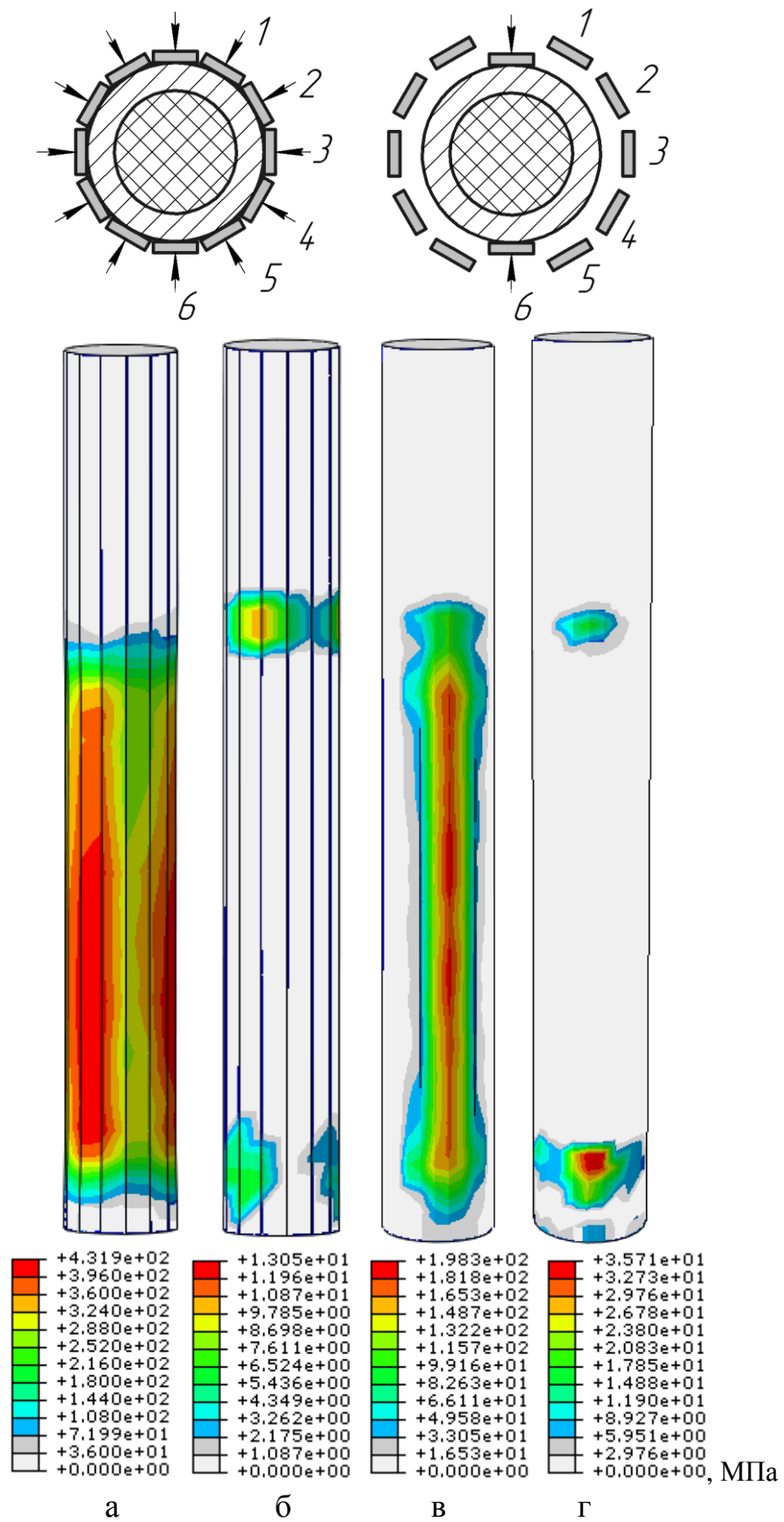
в



г

а, в) – $\sigma_{руйн}$ (МПа); б, г) – p (МПа); а, б) – від l_1 (мм); в, г) – від r_1 (мм)

Рисунок 5.8 – Регресійні залежності $\sigma_{руйн}$ і середнього контактного тиску в з'єднанні під час обтискання p від Δ (мм), l_1 і r_1



а, б) – одночасне обтискання усіма штампами; в, г) – послідовне;

а, в) – під час обтискання; б, г) – після обтискання

Рисунок 5.9 – Контактний тиск p на поверхні стержня тривимірної моделі

Розроблені СЕМ і САПР може бути використана для ґрунтового різностороннього аналізу з'єднань такого типу (рис. О.1). Зокрема можна оптимізувати інші параметри з'єднання, такі як зовнішній діаметр головки, форму штампів, нерівномірність тиску обтискання та попередній натяг, розраховувати з'єднання на втомну міцність, удар, аналізувати міцність з'єднання для різних механічних характеристик склопластику.

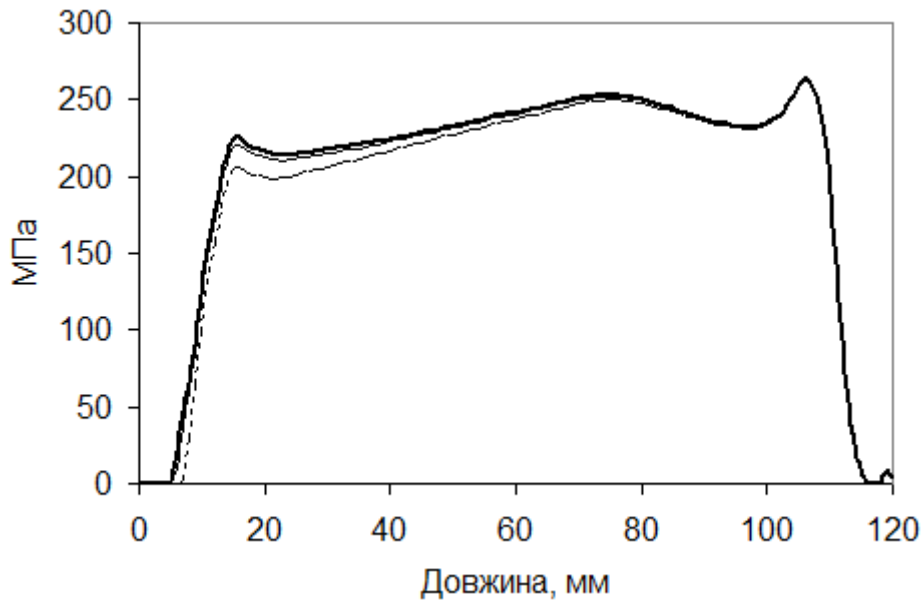
5.1.4 Моделювання гармонічного осьового навантажування пресового з'єднання склопластикової насосної штанги

Виконано аналіз відклику СЕМ пресового з'єднання склопластикового стержня зі сталевим ніпелем на зовнішнє гармонічне осьове навантаження частотами 0-20 кГц [332, 333]. Високочастотні вібрації можуть виникати після ударних навантажень на головку штанги та у разі різноманітних порушень нормальної роботи установки. В Abaqus/CAE 6.14 розроблена осесиметрична модель пресового з'єднання склопластикового стержня зі сталевим ніпелем. Так як розміри моделі задані в міліметрах, то використовували систему одиниць міліметр-тонна-секунда. Діаметр стержня 22 мм, зовнішній діаметр ніпеля 34 мм, довжина з'єднання 120 мм. На торці ніпеля задана гранична умова симетрії YSYMM. Пружно-пластична модель матеріалу ніпеля відповідає сталі 40 ($E=210$ ГПа, $\nu=0,28$, густина $7,8 \cdot 10^{-9}$ т/мм³). Пружна модель матеріалу стержня володіє наступними характеристиками: модуль пружності в осьовому напрямку $E_y=0,5 \cdot 10^5$ МПа, в радіальному напрямку $E_x=0,1 \cdot 10^5$ МПа, коефіцієнт Пуассона $\nu_{xy}=0,22$,

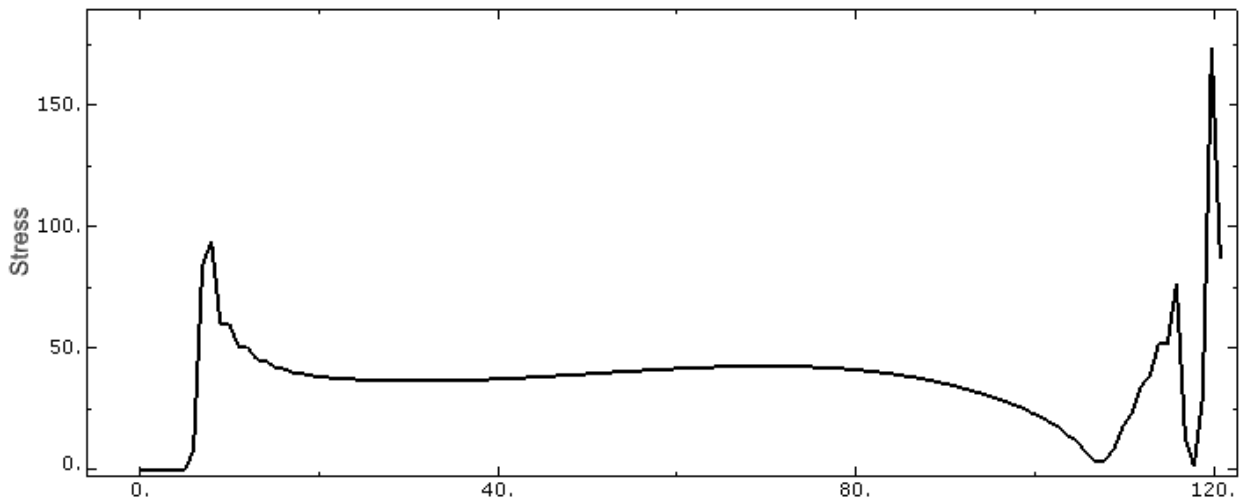
густина $2,1 \cdot 10^{-9}$ т/мм³. Між деталями моделювався контакт «поверхня-доповерхні» (Surface-to-surface contact) з скінченним ковзанням (Finite Sliding) та коефіцієнтом тертя 0,1. Середній розмір елементів – 1 мм.

На першому статичному кроці (Static, General) моделювали радіальне обтискання ніпеля жорсткими штампами довжиною 100 мм і радіусом заокруглення 5 мм. Глибина переміщення штампів – 0,25 мм. На наступних статичних кроках моделювали навантажування з'єднання тиском p , який діє на верхній торець стержня та імітує зовнішнє навантаження розтягу. Розраховували з'єднання для мінімально допустимого значення p (-47 МПа) та максимально допустимого (-185 МПа) згідно діаграми Гудмена [1]. На рис. 5.10а показано розподіл контактного тиску по довжині контакту стержня і ніпеля для наступних значень p : 0 МПа, -47 МПа, -185 МПа. Значення довжини 0 відповідає верхній частині з'єднання. Помітно, що після обтискання прямими штампами розподіл контактного тиску по довжині не рівномірний. Зі збільшенням навантаження розтягу контактний тиск в верхній частині з'єднання зменшується.

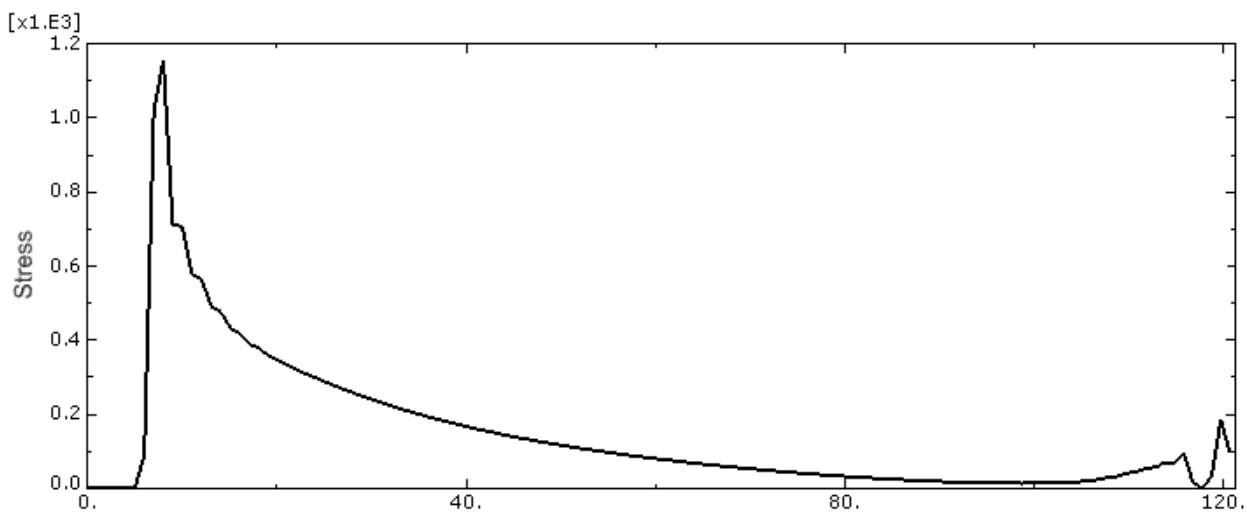
Наступний крок – це обчислення власних частот (Natural frequency extraction). В діапазоні частот 0-20 кГц були знайдені власні частоти зі значеннями 8039,6 Гц та 18944 Гц. За формами вільних коливань (рис. 5.11) та амплітудами контактного тиску (рис. 5.10б,в) помітно, що радіальні переміщення більш характерні для другої частоти. В цьому випадку можливе роз'єднання верхньої частини з'єднання (рис. 5.11б). Значення осьових переміщень більші для першої частоти. У цьому випадку помітне звуження (рис. 5.11а) нижньої частини ніпеля.



а



б



в

а) – контактний тиск для значень p : 0 МПа (—), 47 МПа (---), 185 МПа (- · -);
 б, в) – амплітуда контактного тиску (МПа) для частоти 8041 Гц (б), 18946 Гц (в)

Рисунок 5.10 – Розподіл по довжині контакту стержня і ніпеля з'єднання контактного тиску та його амплітуди

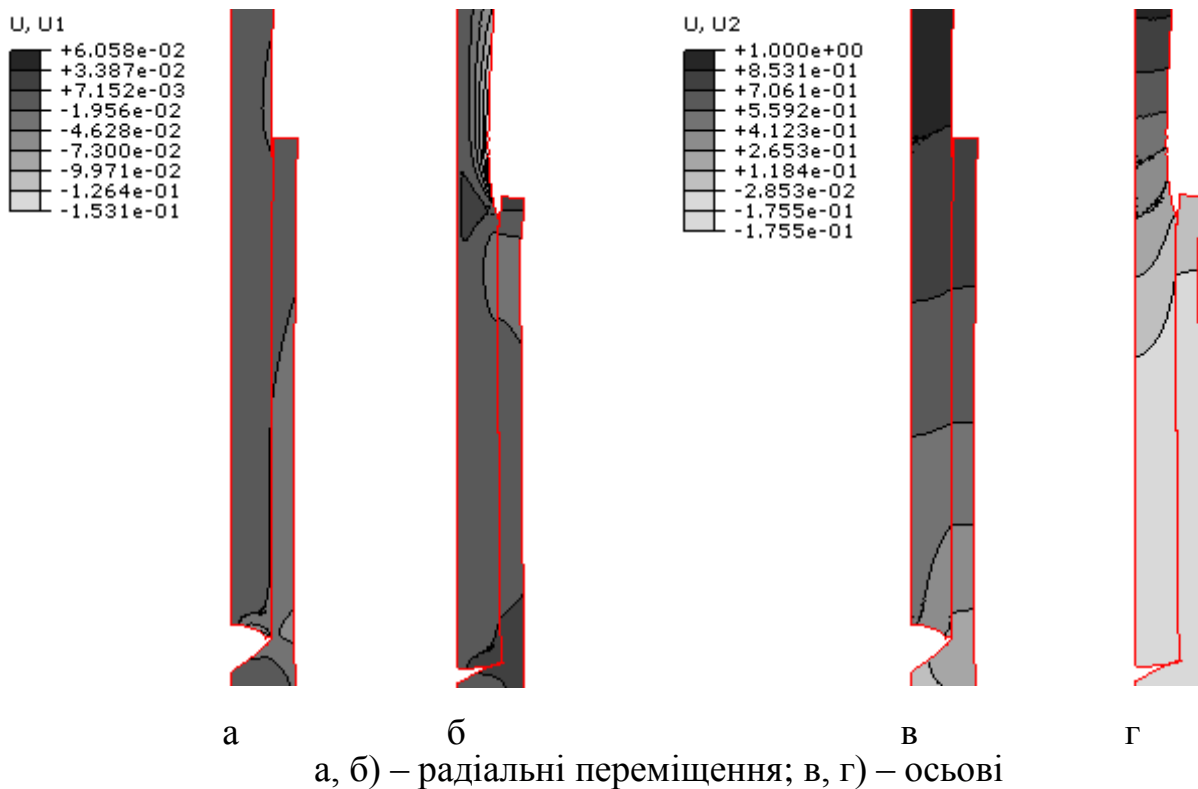
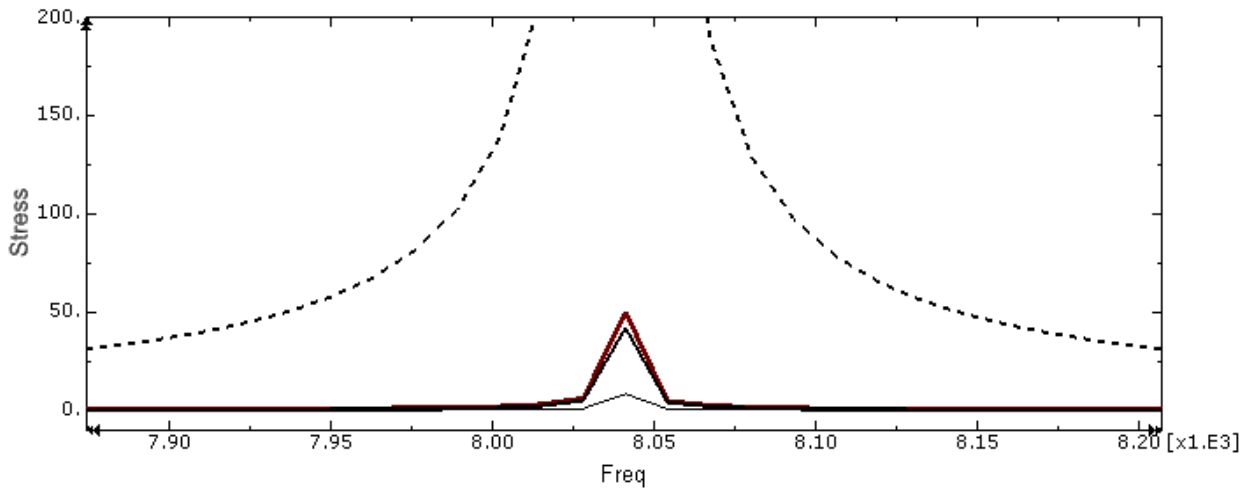
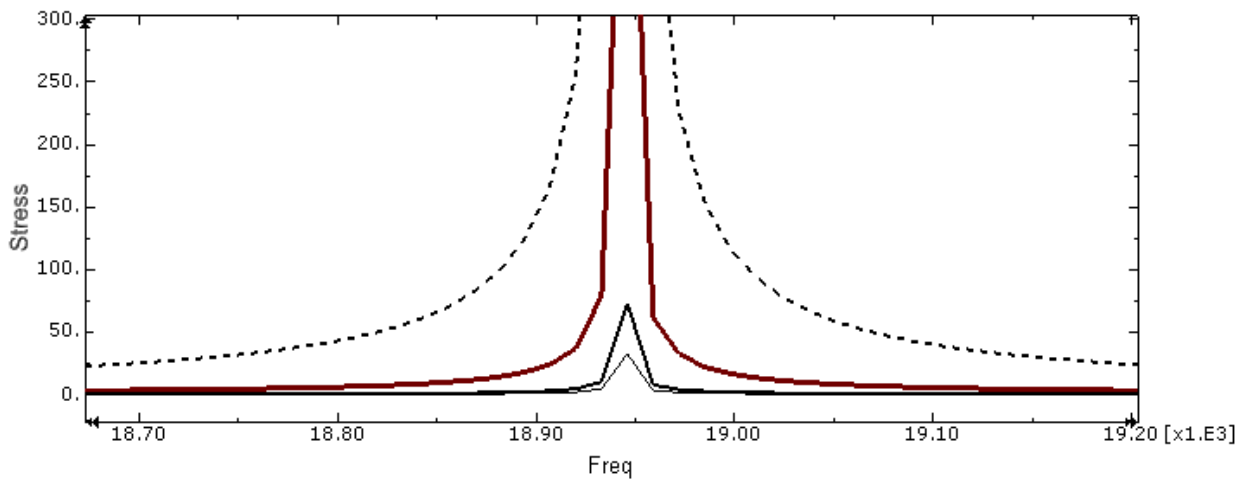


Рисунок 5.11 – Форми вільних коливань для частот 8039,6 Гц (а, в), 18944 Гц (б, г)

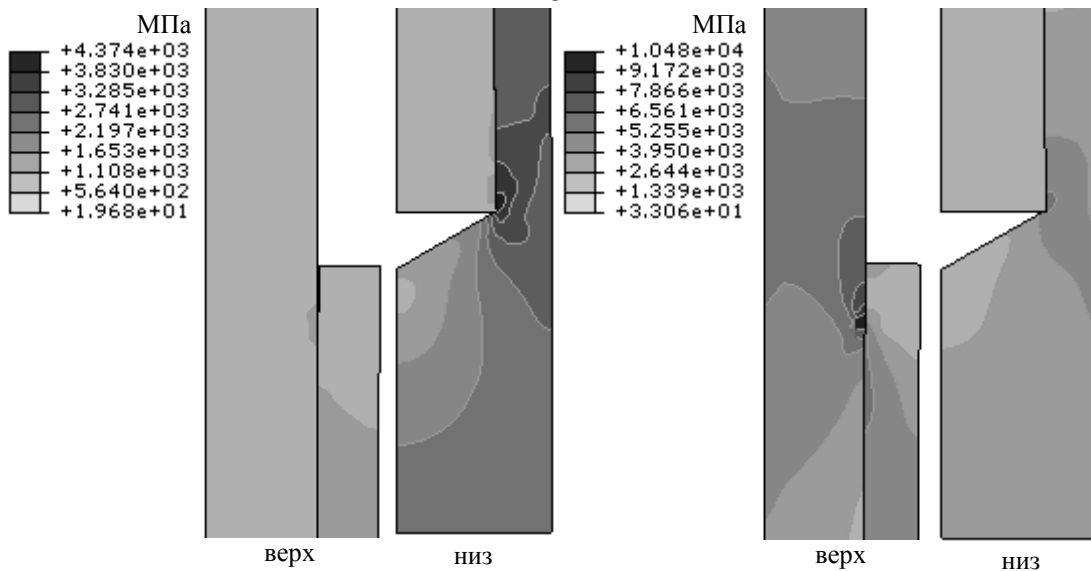
Наступний крок – це стаціонарний динамічний аналіз прямим методом (Direct-solution steady-state dynamic analysis). Цей тип аналізу більш точний ніж попередній. Було вибрано опції Compute real response only (розраховувати тільки дійсний відклик) та Include friction-induced damping effects (включити ефекти демпфування, викликані тертям). Діапазон частот 7-20 кГц розбивався на 1000 точок. В кожній точці проводився аналіз відклику моделі на зовнішнє гармонічне збудження зі значенням частоти в точці. В якості зовнішнього гармонічного збудження був вибраний тиск, прикладений до верхнього торця ніпеля амплітудою -1 МПа та середнім значенням -116 МПа. Незважаючи на неможливість розділення контактних поверхонь в такому аналізі, є можливість обчислення значень контактного тиску. Помітно (рис. 5.12а,б), що найбільші амплітуди контактного тиску спостерігаються в верхній частині з'єднання, а найменші – в нижній.



а



б



верх

низ

в

верх

низ

г

(—) – контактний тиск (МПа), верх з'єднання; (—) – середина; (—) – низ; (- -) – σ_M (МПа);
 а) – в околі 8039,6 Гц; б) – в околі 18944 Гц; в) – 8041 Гц; г) – 18946 Гц

Рисунок 5.12 – Амплітуди контактного тиску в різних зонах з'єднання, амплітуда напружень σ_M в нижній зоні концентрації напружень ніпеля (а, б) та розподіл амплітуди напружень σ_M (в, г)

З рис. 5.12 також видно, що якщо прийняти граничну допустиму амплітуду напружень в нижній зоні ніпеля 100 МПа, то експлуатаційні частоти не повинні лежати в межах $8039,6 \pm 50$ Гц та 18944 ± 50 Гц. В іншому випадку можливе швидке втомне руйнування з'єднання. На рис. 5.12в,г показано розподіл амплітуди напружень σ_m в зонах концентрації напружень з'єднання. Помітно, що частота 8041 Гц може спричинити втомне руйнування сталевого ніпеля в нижній частині з'єднання, а частота 18946 Гц може стати причиною втомного руйнування склопластикового стержня вверху з'єднання. Напруження в зонах концентрації напружень можна зменшити шляхом збільшення радіуса заокруглення в перехідній ділянці ніпеля та радіуса заокруглення штампа.

5.1.5 Моделювання втомної міцності ніпеля склопластикової насосної штанги

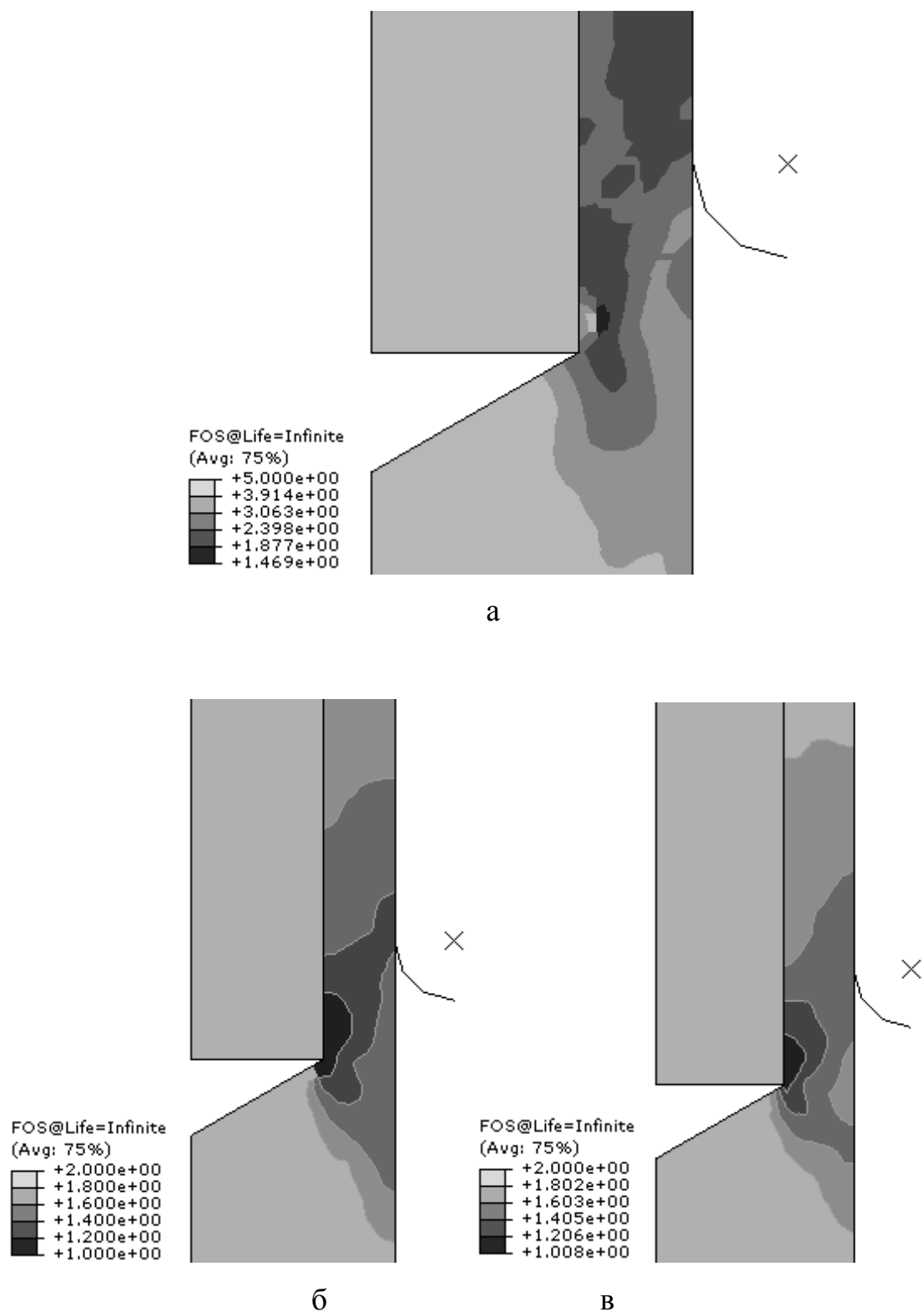
Проведено також аналіз втомної міцності сталевого ніпеля з'єднання на основі СЕМ пресового з'єднання в Abaqus/CAE 6.14 [334]. Параметри моделі з'єднання ШН з діаметром тіла 22 мм описані в праці [331]. Глибина переміщення штампів – 0,25 мм. Для динамічного аналізу додатково потрібно ввести значення густини сталі ($7,8 \cdot 10^{-9}$ т/мм³) і склопластику ($2,1 \cdot 10^{-9}$ т/мм³). Середній розмір скінченних елементів – 1 мм. На статичних кроках моделювали навантажування з'єднання тиском p , який діє на верхній торець стержня та імітує зовнішнє навантаження розтягу. Впродовж циклу втомного навантажування діаграма Гудмена [1] рекомендує мінімальне (-47 МПа) та максимальне (-185 МПа) допустиме значення p . На кроці стаціонарного динамічного аналізу прямим методом (Direct-solution steady-state dynamic analysis) в якості зовнішнього гармонічного збудження був вибраний цей тиск p амплітудою -1 МПа та середнім значенням -116 МПа. Обчислення втомної міцності виконували в fe-safe 6 [202, 208]. Обчислювали коефіцієнт запасу втомної міцності D за критерієм Брауна-Міллера (1.11). В базі даних fe-safe був вибраний аналог сталі 40 – сталь SAE1040. В даному випадку D розраховували для 10^7 циклів навантажування.

Спочатку розрахуємо значення D за результатами статичного аналізу. На рис. 5.13а показано розподіл D в нижній частині ніпеля, якщо впродовж циклу зовнішнє навантаження p змінюється від $p_{min}=-47$ МПа до $p_{max}=-185$ МПа. Помітно, що найменші значення D (1,469) локалізуються біля кута перехідної зони ніпеля.

Розрахуємо граничні значення допустимої частоти навантажування в околі першої власної частоти (8039,6 Гц). Для цього передамо в fe-safe результати стаціонарного динамічного аналізу – дійсні та уявні складові напружень для заданих частот. Створимо блок циклічного навантажування Modal Block, який потрібно заповнити, наприклад, наступними даними: номер набору даних з дійсними значеннями напружень (Real Dataset=273), номер набору даних з уявними значеннями напружень (Imaginary Dataset=274), частота в герцах (Frequency=8002), кількість повторів (Repeats=1), довжина повтору в секундах (Length=1.26e-04). Тут Length=1/Frequency. За цими даними fe-safe створить файл визначення навантаження (ldf-файл), який містить блок:

```
BLOCK n=1, dt=1.26e-04, modal=steady
rds=273, ids=274, freq=8002
END
```

Такий розрахунок в околі власної частоти 8039,6 Гц дозволив виявити частоти, які забезпечують мінімальне допустиме значення D ($D=1$) – 8002 Гц та 8080 Гц (рис. 5.13б,в) [333]. Іншими словами експлуатація з частотами в діапазоні від 8002 Гц до 8080 Гц не забезпечує циклічної довговічності $N=10^7$ циклів за даних умов та її потрібно уникати.



а) – квазістатичний цикл втомного навантажування $p_{min}=-47$ МПа, $p_{max}=-185$ МПа;
 б, в) – навантажування частотою 8002 Гц (б) та 8080 Гц (в)

Рисунок 5.13 – Коефіцієнт запасу D в нижній сталевій частині ніпеля

5.2 Моделі штанг і НКТ з дефектами та склопластиковими бандажами

5.2.1 Геометричні моделі дефектів штанг і НКТ

Під час експлуатації ШН і НКТ спостерігаються різноманітні їхні дефекти: тріщини, корозійні дефекти, дефекти від механічного зношування та деформації. Нерідко необхідно обґрунтовано прийняти рішення про можливість експлуатації ШН чи НКТ з тим чи іншим дефектом. Для цього з успіхом можуть бути застосовані сучасні CAD/CAE-системи, які реалізують МСЕ для задач теорії пружності та пластичності. Основною проблемою є моделювання геометрії того чи іншого дефекту на НКТ. Нижче проведено аналіз можливостей САПР SOLIDWORKS побудови різноманітних дефектів [17].

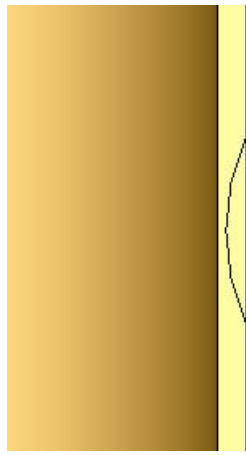
В залежності від видів дефектів і цілей моделювання для побудови дефектів у SOLIDWORKS 2011 можна застосувати наступні елементи: різні способи вирізу "Виріз" ("Витягнути", "Повернути", "За траєкторією", "За перетинами"), елементи "Купол", "Вільна форма", "Деформувати", "Згин", різні способи побудови поверхонь з наступним "Виріз/Поверхнею" або "Розділити".

Наведемо приклади деяких способів побудови дефектів труб у SOLIDWORKS 2011 (рис. 5.14). Тріщина може бути змодельована елементом "Лінія роз'єму", який створює проекцію ескізу на грань (рис. 5.14 а). В даному випадку грань ділиться на дві частини: поверхню осьової тріщини та поверхню осьового перетину труби, на яку можуть бути задані відповідні граничні умови під час моделювання МСЕ [124, 179]. Дефект правильної форми від зношування НКТ штанговими муфтами доцільно створювати за допомогою елемента "Виріз/Витягнути" (рис. 5.14 б). В даному випадку коло вирізає в тілі НКТ зношену поверхню в напрямку непаралельному осі труби. Корозійний дефект правильної форми можна створити елементами "Поверхня/Повернути" і "Розділити" (рис. 5.14 в). Такий спосіб дозволяє отримати окремо тіло труби з дефектом і тіло дефекту, що необхідно, наприклад, для розрахунку міцності НКТ

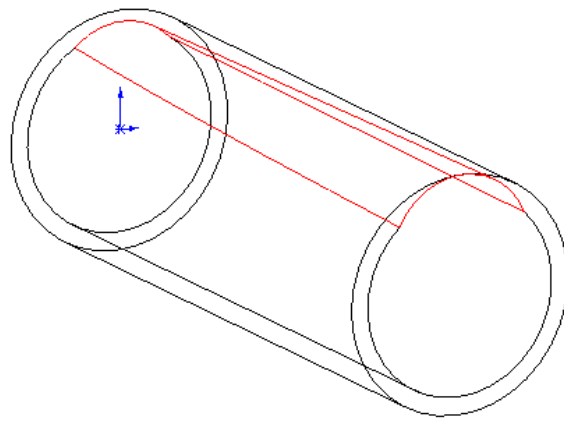
з виправленим епоксидним клеєм дефектом. Корозійний дефект довільної форми легко моделювати елементами "Сплайн на поверхні" (для побудови контуру дефекту на поверхні), "Лінія раз'єму" (для отримання поверхні, обмеженої контуром дефекту) і "Поверхня/Вільна форма" або "Поверхня/Заповнити" (для створення поверхні дефекту). Для розділення тіла НКТ і дефекту слід використовувати елемент "Розділити" (рис. 5.14 г). Під час моделювання таких дефектів важко отримати потрібні розміри, проте в SOLIDWORKS є інструмент "Аналіз товщини", який візуалізує товщину дефекту (рис. 5.14г). Точні поверхні дефектів можна отримувати за допомогою елементу "Гранична поверхня", а додатковий модуль ScanTo3D дозволяє отримувати моделі поверхонь, наприклад шляхом сканування чи дефектоскопії реальних дефектів. Деформації НКТ типу локальних вм'ятин просто моделювати елементом "Деформувати" (рис. 5.14 д). Можна вибрати різні види деформації, наприклад "точка деформації" або "витіснення поверхні". Згин НКТ моделюється елементами "Згин" ("згинання" або "поворот") або "За траєкторією", якщо необхідна точність розмірів деформації (рис. 5.14 е).

На рис. 5.15 показано спосіб побудови дефекту довільної форми за допомогою інструментів поверхня-лофт (Surface-Loft) та розділення тіл поверхнею (Split), який теж дозволяє отримати тіла труби і дефекту. Для побудови поверхні дефекту, в перпендикулярних до осі труби площинах Plane1, Plane2, Plane3 будуються ескізи Sketch2, Sketch3, Sketch4, які містять замкнуті сплайни. Координати точок цих сплайнів можна отримати, наприклад, шляхом вимірювання відхилення від округлості в рівновіддалених перетинах труби.

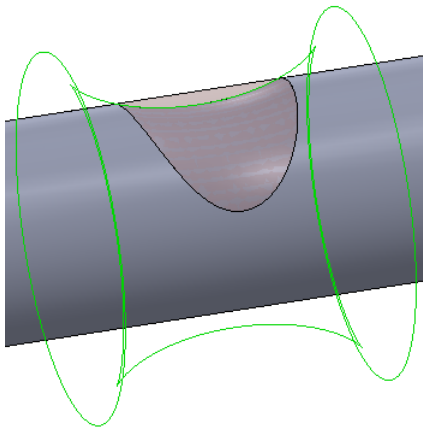
Таким чином SOLIDWORKS дозволяє легко створювати тривимірні параметричні моделі НКТ з різноманітними окремими дефектами або їхніми комбінаціями, і за допомогою FEA-модуля Simulation обґрунтовувати можливість їхньої експлуатації, а також досліджувати вплив розмірів того чи іншого дефекту на напруження в НКТ.



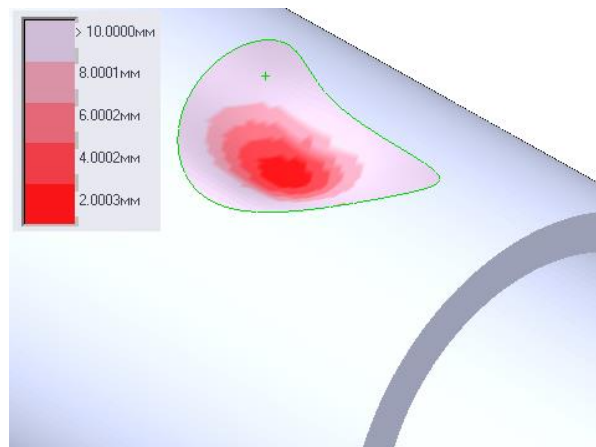
а



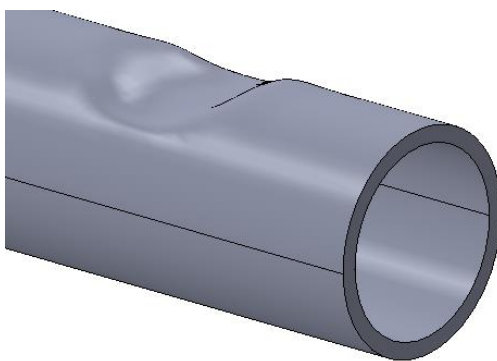
б



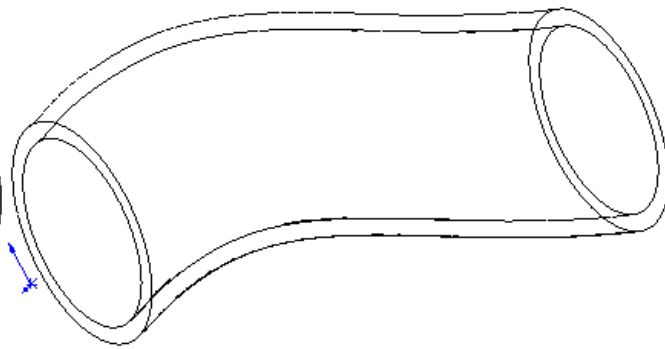
в



г



д



е

Рисунок 5.14 – Приклади моделювання дефектів НКТ у SOLIDWORKS 2011

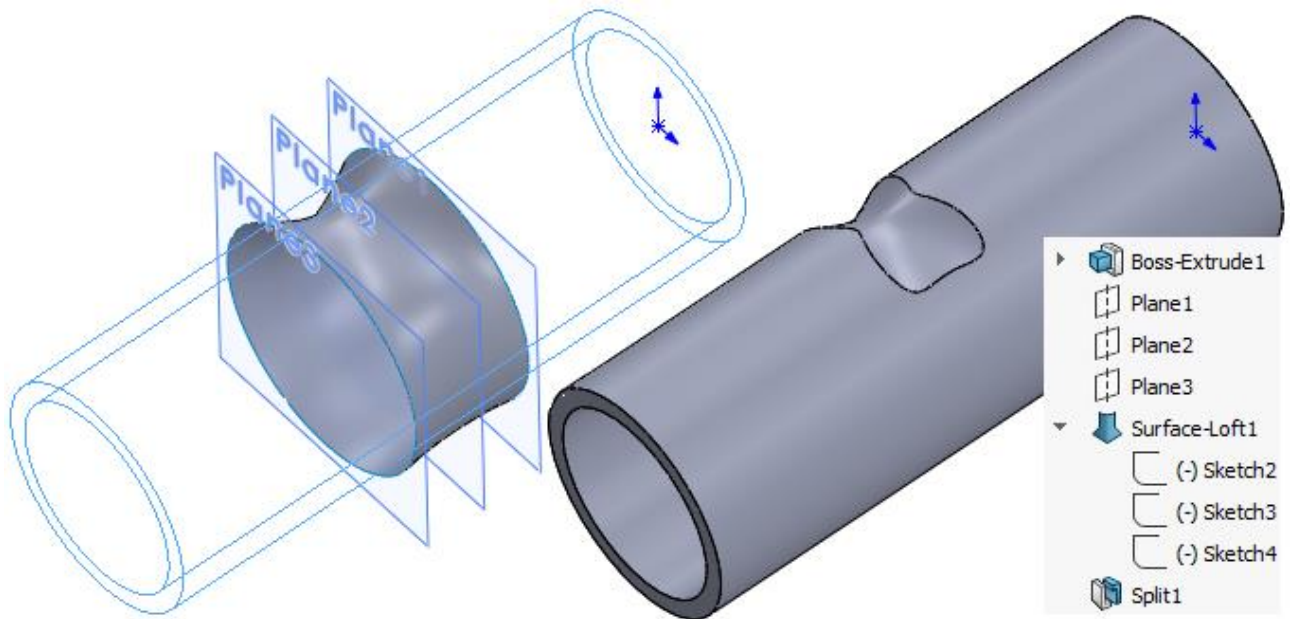


Рисунок 5.15– Спосіб побудови дефекту довільної форми за допомогою інструментів SOLIDWORKS Surface-Loft та Split

5.2.2 Вплив склопластикового бандажа на напруження в НКТ

Проведені дослідження [118, 124, 127] та досвід ремонту трубопроводів ПКМ [119, 120] дозволяє сформулювати наступну послідовність операцій для ремонту ШН та НКТ з ненаскрізними дефектами [335]:

- піскоструминна обробка поверхні або обробка металічними щітками;
- вимірювання дефекту і прийняття рішення про допустимість ремонту, визначення типу ремонту (ремонт заливним ПКМ, муфтовим, або комбінований), оптимізація БС;
- обезжирювання поверхні ацетоном;
- осушування поверхні за 40...60°C;
- пошарове нанесення заливного ПКМ;
- нанесення клею (рекомендуються клеї з високою адгезією до поверхні, зокрема епоксидні клеї з міцністю до сталі під час зсуву 10-15 МПа) на поверхню труби товщиною 0,5... 0,7 мм, полімеризація;
- просочування в епоксидному зв'язуючому стрічки з скловолокна, намотування з контролем натягу і фіксація;

- нанесення клею з меншим модулем пружності на торці бандажа з ціллю утворення фасок і зменшення концентрації напружень;
- полімеризація 24 год за 15-35°C (5..6 год за 40-60°C), контроль затвердіння;
- механічна обробка фасок для надання їм еліптичної форми;
- за потреби відбувається нанесення на БС водостійкого поліуретанового покриття або формування на БС поліуретанового або поліамідного протектора-центратора. Останнє дозволяє додатково захистити БС від водопоглинання, ударів і зношування.

Технологічний процес ремонту чи зміцнення ШН та НКТ, у порівняння з ремонтом трубопроводів, відбувається в цеху, що дозволяє досягти набагато вищого рівня якості ремонту, продуктивності та знизити його собівартість [335]. Під час транспортування деталей з БС необхідно захистити поверхню БС від пошкодження, наприклад за допомогою тимчасових захисних муфт. Така технологія також може бути застосована і для ремонту викидного трубопроводу (шлейфу від свердловини) та порожнистих ШН. Перед нанесенням бандажа ШН можуть додатково проходити дифузійну металізацію [118, 336].

Результати експериментальних досліджень БС наведено в працях [3, 124]. Для спрощення експерименту випробовування проводилось на стенді для гідравлічного випробовування балонів. Як фізичні моделі труб з БС використовували кисневий і газовий вуглекислотний балони з БС на зовнішній циліндричній поверхні (табл. 5.2). Для вимірювання кільцевого і осевого напружень на поверхню балона під БС і за БС монтували тензорезистори. Використовували конструкцію БС на основі наповнювача (склотканина Т-10-80 ГОСТ 19170-73) та зв'язуючого (ЭДТ-10 ОСТ 92-0957-74). Адгезію БС до тіла балона здійснювали на основі епоксидного клею К-153 (ОСТ 92-0948-74, ОСТ 92-0949-74). Ширина БС – 300 мм.

Порівнювались результати експерименту, аналітичних і чисельних обчислень. Для аналітичних обчислень напружень на поверхні труби використовували формули 5.1-5.2, для чисельних – осесиметричну СЕМ з ортотропною моделлю матеріалу БС.

В трубі (балоні) без БС, на яку діє тільки внутрішній тиск, значення

кільцевих σ_θ і осьових σ_z напружень можуть орієнтовно бути обчислені за відомими формулами:

$$\sigma_\theta = Pr_1/t_1, \quad \sigma_z = \sigma_\theta/2,$$

де P – внутрішній тиск,

r_1 – внутрішній радіус труби,

t_1 – товщина стінки труби.

На основі задачі Ляме Щербиною Н. М. отримано формули для обчислення радіальних і кільцевих напружень в трубi з БС [3, 4]:

$$\sigma_r = A - \frac{B}{r^2}, \quad \sigma_\theta = A + \frac{B}{r^2} \quad (5.1)$$

де r – змінний радіус,

A, B – коефіцієнти

$$A = \frac{Pr_1^2 - qr_2^2}{r_2^2 - r_1^2}, \quad B = \frac{(P - q)r_1^2 r_2^2}{r_2^2 - r_1^2},$$

де r_2 – зовнішній радіус труби,

r_3 – зовнішній радіус БС,

q – контактний тиск між трубою і БС

$$q = -P \left[1 + C \left(\frac{r_2^2}{r_1^2} - 1 \right) \right],$$

де C – константа

$$C = \frac{E_f - (1 - \nu_1) \left(1 - \frac{r_2^2}{r_3^2} \right)}{\left[\frac{r_2^2}{r_1^2} (1 - \nu_1) + (1 + \nu_2) \right] \left(1 - \frac{r_2^2}{r_3^2} \right) - E_f \left(\frac{r_2^2}{r_1^2} - 1 \right)},$$

де ν_1, ν_2 – коефіцієнти Пуассона матеріалу труби і БС,

E_f – константа

$$E_f = \frac{E_1}{E_2} \left[(1 - \nu_2) \frac{r_2^2}{r_3^2} + (1 + \nu_2) \right],$$

де E_1, E_2 – модулі пружності матеріалу труби і БС.

Осьове напруження σ_z в трубi з БС, яке виникає від дії осьової сили F , автором запропоновано обчислювати з умови рівності осьових деформацій труби (індекс 1) і БС (індекс 2):

$$\frac{F_1}{A_1 E_1} = \frac{F_2}{A_2 E_2},$$

де F_1, F_2 – осьові сили, що діють на трубу і БС,

A_1, A_2 – площі поперечного перетину тіла труби і БС.

$$\sigma_z = F / (A_1 + A_2 E_2 / E_1), \quad (5.2)$$

де $F = F_1 + F_2$ – сумарна осьова сила, що діє на трубу і БС. Якщо діє тільки внутрішній тиск P , то $F = P \pi r_1^2$. В формулі (5.2) значення E_2 – це осьовий модуль пружності БС (E_z).

Для МСЕ задано такі механічні характеристики матеріалів:

– сталеві труби: $E=210$ ГПа, $\nu =0,28$;

– БС: – $E_\theta=40$ ГПа, $E_z=10$ ГПа, $E_r=10$ ГПа, $\nu =0,28$ (для МСЕ). Для

формули (5.1) використовували $E_2=E_\theta$, для формули (5.2) – $E_2=E_z$.

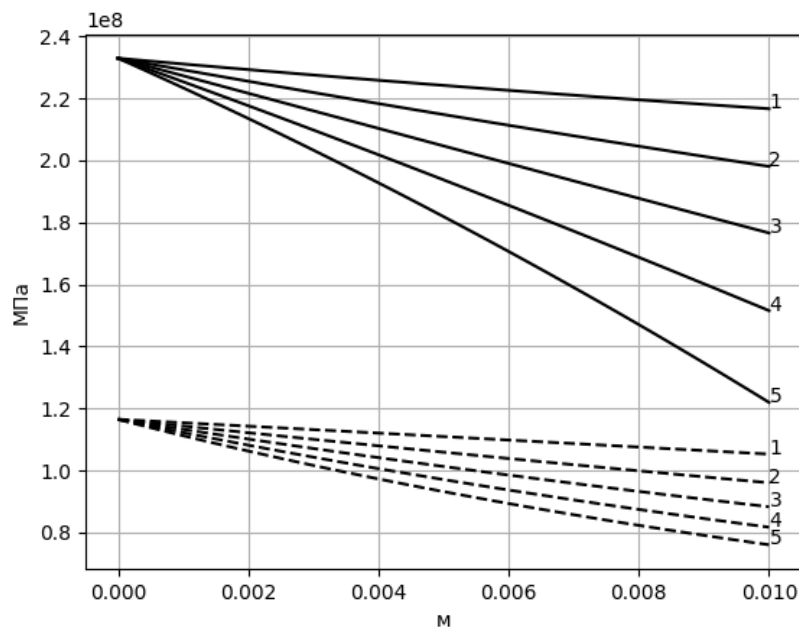
Результати експерименту і моделювання показані в табл. 5.2. Табл. також містить дані моделювання НКТ діаметром 73 мм з БС. Помітно, що у порівнянні з формулами 5.1, 5.2 моделювання МСЕ дає дещо занижені значення $\sigma_{\theta 1}/\sigma_{\theta 2}$ та дещо завищені значення $\sigma_{z 1}/\sigma_{z 2}$. В цілому результати показують, що БС зменшує кільцеві напруження в 1,18-1,54 рази, а осьові – в 1,05-1,21 рази. Для оптимізації товщини БС можуть бути використані залежності (рис. 5.16), отримані автором за формулами 5.1, 5.2. Наприклад, для НКТ 73x5,5 з БС ($t=5,5$ мм, $E_\theta=40$ ГПа, $E_z=10$ ГПа) та $P=45$ МПа еквівалентні напруження на поверхні труби з БС:

$$\sigma_{\text{екв}} = \sqrt{\sigma_\theta^2 + \sigma_z^2 - \sigma_\theta \sigma_z} = \sqrt{195^2 + 115^2 - 195 \cdot 115} = 170 \text{ МПа.}$$

Таблиця 5.2 – Експериментальні та теоретичні значення кільцевих σ_θ і осьових σ_z напружень (МПа) на поверхні труби з БС

Діаметр d_1 і товщина труби t_1 , товщина БС t (мм), тиск P (МПа).	Зона пов. труби*	σ_θ експ.	σ_θ ф-ла 5.1	σ_θ МСЕ	σ_z експ.	σ_z ф-ла 5.2	σ_z МСЕ
$d_1=219, t_1=7, t=10, P=16$ (кисневий балон)	1	238	226	-	119	113	-
	2	155	153	175	101	105	93
	3	167	-	179	106	-	93
	σ_1/σ_2	1,53	1,48	1,29	1,18	1,08	1,21
	1	137	129	-	68	65	-
$d_1=150, t_1=5,2, t=8, P=10$ (газовий балон)	2	113	84	100	62	60	55
	3	128	-	103	58	-	55
	σ_1/σ_2	1,21	1,54	1,29	1,1	1,08	1,18
	1	-	233	-	-	116	-
	2	-	190	198	-	110	105
$d_1=73, t_1=5,5, t=5,5, P=45$ (НКТ 73x5,5 ГОСТ 633-80)	3	-	-	205	-	-	105
	σ_1/σ_2	-	1,23	1,18	-	1,05	1,10

Примітка: *1 – без БС, 2 – під БС (в центрі БС), 3 – під БС (скраю БС), σ_1/σ_2 – відношення напружень без БС до напружень з БС.



(—) – $\sigma_\theta(t, E_\theta)$, формула 5.1; (- -) – $\sigma_z(t, E_z)$, формула 5.2;

значення E_θ або E_z : 1 – 10 ГПа; 2 – 20 ГПа; 3 – 30 ГПа; 4 – 40 ГПа; 5 – 50 ГПа

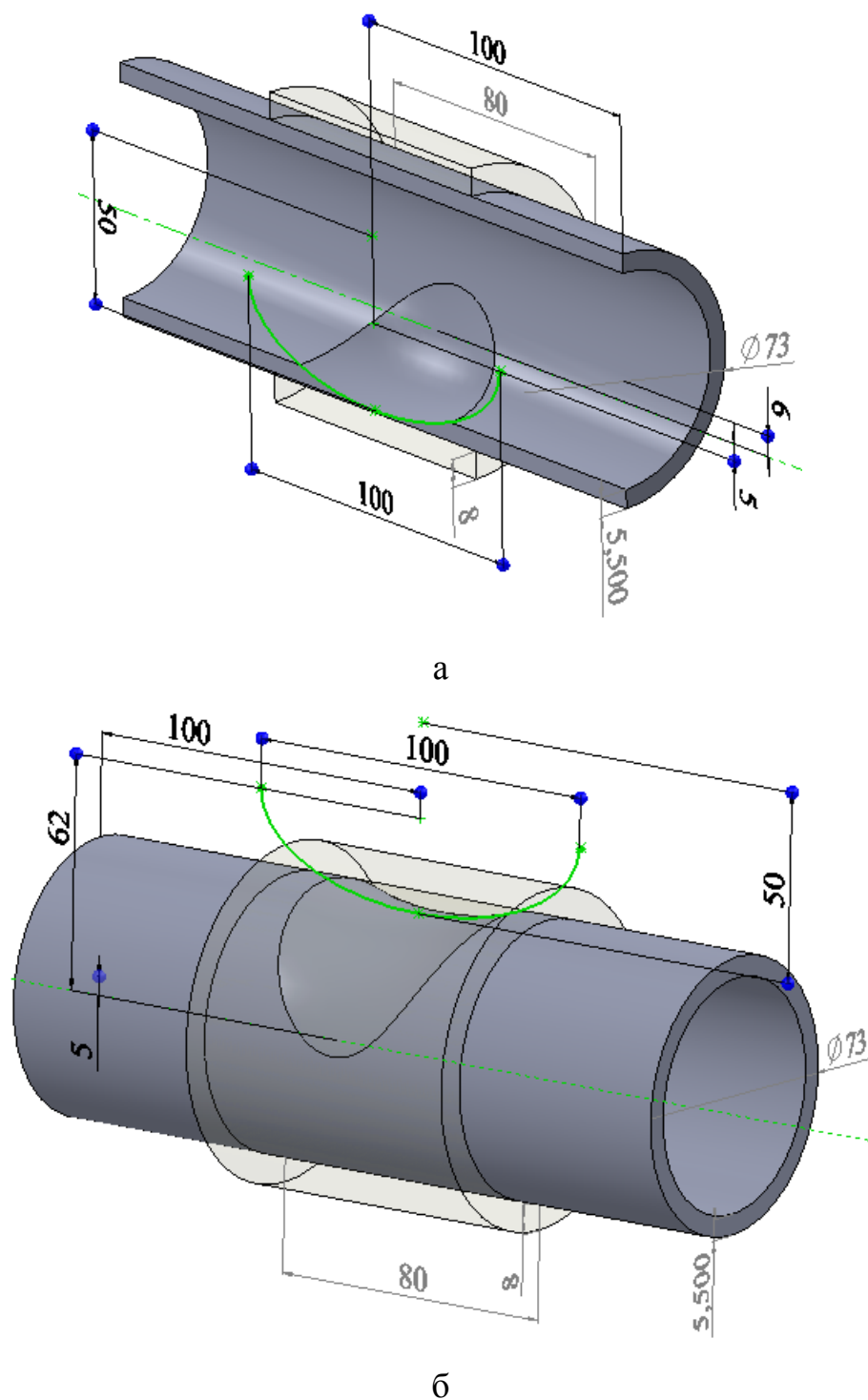
Рисунок 5.16 – Залежності кільцевих σ_θ і осьових σ_z напружень на поверхні НКТ 73x5,5 від товщини БС t (м) для $P=45$ МПа для різних модулів пружності БС E_θ та E_z

Виконано також експериментальне випробування цих БС на кисневому і газовому балонах на циклічну міцність [337]. Тиск змінювали від 0 до 20 МПа. Після 10^4 циклів ніяких розшарувань та порушення міцності БС виявлено не було. Також проводилось аналогічне випробування балона з дефектами і БС [124]. На зовнішню циліндричну поверхню балона було нанесено два дефекти глибиною 0,5 мм, шириною 5 мм, довжиною 40 мм і розміщених діаметрально. Після 10^4 циклів ніяких розшарувань та порушення міцності БС виявлено не було.

В праці [338] розроблена СЕМ труби діаметром 273 мм з корозійною виразкою і БС з товщиною 10 мм. Внутрішній тиск – 7 МПа, контактний тиск між БС і трубою – 10 МПа. Модель показала зменшення σ_m в трубі з 742 МПа до 418 МПа навіть без виправлення дефекту шпаклівкою з заливного ПКМ (епоксидний клей).

Для SOLIDWORKS автором розроблена параметрична модель НКТ з корозійним дефектом правильної форми і БС [5]. Дефект являє собою поверхню обертання, яка отримується обертанням еліпса навколо осі, перпендикулярній осі НКТ (рис. 5.17). Перевагами такого способу побудови є можливість отримання великої кількості різних форм внутрішніх і зовнішніх дефектів, які описуються невеликою кількістю параметрів.

Розглянемо результати моделювання НКТ 73x5,5 з такими дефектами. Параметри БС: довжина 80 мм, товщина $t=0\dots 8$ мм. Параметри дефектів: велика вісь еліпса 100 мм, мала – 50 мм, відстань від осі обертання поверхні до центра еліпса – 5 мм (62 мм для зовнішнього дефекту), відстань від осі НКТ до осі обертання поверхні $h=5$ мм. Для сталі НКТ $E=210$ ГПа, $\nu=0,28$. Модуль пружності БС зменшено у два рази для врахування можливої деградації матеріалу: $E_r=4,1\cdot 10^9$ Па, $E_\theta=11\cdot 10^9$ Па, $E_z=2,1\cdot 10^9$ Па, $\nu=0,28$. Для епоксидного клею: $E=2,4\cdot 10^9$ Па, $\nu=0,35$.



а) – внутрішній дефект; б) – зовнішній

Рисунок 5.17 – Тривимірні параметричні моделі НКТ з дефектом та БС

Результати симуляції за допомогою модуля Simulation для внутрішнього тиску в НКТ 10 МПа наведено в таблиці 5.3.

Таблиця 5.3 – Залежності максимального значення σ_m в НКТ від товщини БС

Товщина БС, мм	Максимальне еквівалентне напруження σ_m , МПа			
	Внутрішній дефект		Зовнішній дефект	
	$h=5$ мм	$h=6$ мм	без вирівнювання дефекту	вирівнювання дефекту заливним ПКМ
0	208	524	316	288
2	197	462	320	268
4	190	426	320	256
6	184	404	320	248
8	180	368	320	243

Помітно, що збільшення товщини БС є більш ефективним для ремонту глибоких внутрішніх дефектів, а ремонт зовнішніх дефектів вимагає попереднього вирівнювання дефекту заливним ПКМ. Даний спосіб моделювання дозволяє обґрунтовувати ефективність ремонту НКТ з різноманітними дефектами БС та оптимізувати параметри БС.

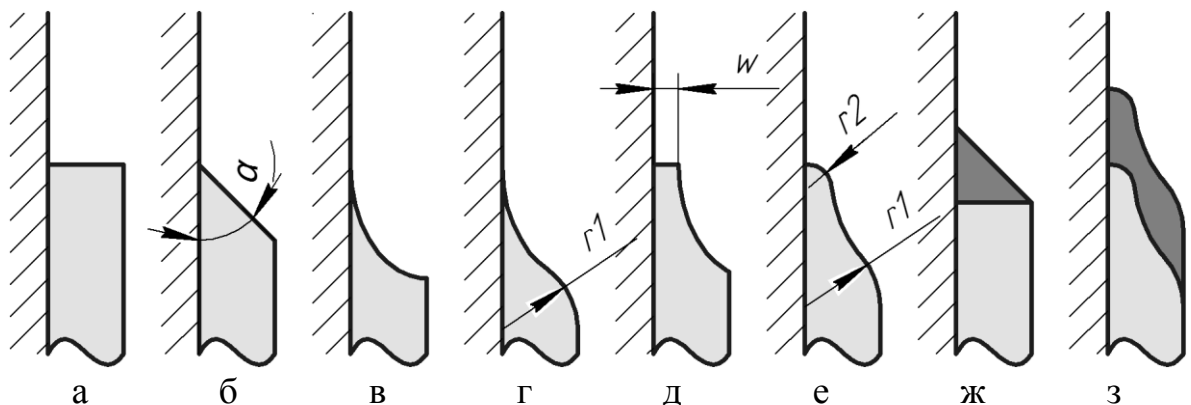
5.2.3 Оптимізація конструкції фаски склопластикового бандажа

БС без фаски (рис. 5.18а) може бути причиною значної концентрації напружень в перехідній зоні труби та її втомного руйнування. Особливо ця проблема актуальна, якщо БС виготовлено з матеріалу з високим модулем пружності. В праці [339] запропоновано способи зменшення концентрації напружень в перехідній зоні, але їхнє теоретичне чи експериментальне обґрунтування виконано не було. В праці [338] виявлено, що прямолінійна фаска під кутом $\alpha=20^\circ$ до поверхні труби може зменшити концентрацію напружень, особливо в тому випадку, коли між БС і трубою є натяг з контактним тиском 10 МПа. Проте обчислення втомної міцності та оптимізації конструкції таких БС виконано не було.

Найпростішою з технологічної точки зору є прямолінійна фаска (рис. 5.18б). Для зменшення концентрації напружень необхідно зменшувати її кут α . Інші

варіанти форми фаски – округлена або прямолінійна фаска з заокругленням. Такі фаски не забезпечують поступового зменшення напружень в місці концентрації напружень навіть після значного зменшення кута α . Але можна застосувати фаски з круговою [339] або еліптичною формою (рис. 5.18в) [340]. Постає задача оптимізації розмірів цієї еліптичної фаски. Для спрощення задачі приймемо, що горизонтальна вісь еліпса (рис. 5.19) є постійною і дорівнює подвійній товщині БС (8 мм). Потрібно знайти оптимальне значення довжини вертикальної осі еліпса за критерієм втомної міцності труби.

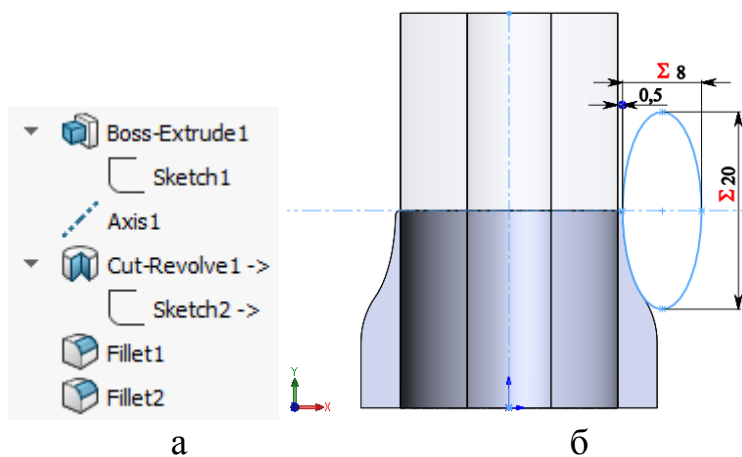
Під час вертикального переміщення ШН і НКТ бандаж може бути пошкоджений внаслідок зіткнення з торцями труб в місцях їхнього з'єднання. Щоб цьому запобігти потрібно виконувати закруглення радіусом r_1 на зовнішньому діаметрі БС (рис. 5.18г). Потрібно також враховувати, що в реальних конструкціях еліпс фаски повинен бути віддалений від поверхні труби на величину w (5.18д, е). Зменшити концентрацію напружень можна шляхом виготовлення фаски з матеріалу, у якого модуль пружності є меншим модуля пружності БС (рис. 5.18ж). Це може бути епоксидна смола або ПКМ РЭМ-Сталь [120]. Інструкція [119] рекомендує застосовувати фаски з клею з кутом $30...45^\circ$. Поєднання способів "е" і "ж" (рис. 5.18з) може максимально зменшити концентрацію напружень.



а) – без фаски; б) – прямолінійна фаска під кутом α , в – еліптична фаска; г) – еліптична фаска з радіусом r_1 ; д) – еліптична фаска з відступом w , е) – еліптична фаска з відступом і радіусами r_1 та r_2 ; ж) – фаска з менш пружного матеріалу; з) – поєднання варіантів "е" і "ж"

Рисунок 5.18 – Осьові перетини варіантів конструкції фаски

На рис. 5.19 показано параметричну геометричну тривимірну модель БС з еліптичною фаскою за варіантом "е" (рис. 5.18) для SOLIDWORKS. Модель будується шляхом кругового вирізу за допомогою елементу Cut-Revolve1. Ескіз Sketch2 містить еліпс з довжиною горизонтальної осі 8 мм та довжиною вертикальної осі $a=20$ мм. Еліпс віддалений від тіла ШН на $w=0,5$ мм. Елементи Fillet1 та Fillet2 призначені для побудови радіусів закруглення $r1$ та $r2$ на БС. Цю модель використано для оптимізації довжини a . Діаметр розглянутої порожнистої ШН 22 мм, діаметр отвору 8,5 мм, товщина БС t 4 мм, половина довжини БС 50мм, $a=20$ мм.



а) – дерево побудови; б) – ескіз фаски

Рисунок 5.19 – SOLIDWORKS-модель БС з еліптичною фаскою

На основі геометричної моделі розроблено осесиметричну СЕМ ШН, на яку діє циклічне зовнішнє осьове навантаження, що створює в тілі напруження розтягу $\sigma_p = 0 \dots 150$ МПа та відповідає гранично допустимому циклу для сталі 40 відповідно діаграми Гудмена. Значення еквівалентних напружень σ_m обчислювали для максимального навантаження циклу ($\sigma_p = 150$ МПа). Коефіцієнт запасу D обчислювали за залежністю Сайнса (1.8).

Спочатку було проаналізовано вплив модулів пружності БС на розподіл коефіцієнта D на поверхні ШН вздовж її осі. Розглядали конструкцію без фаски. Характеристики вихідного матеріалу БС: $E_r=5$ ГПа, $E_\theta=5$ ГПа, $E_z=5$ ГПа, $\nu=0,28$. Розглянуто шість видів матеріалів БС, в яких змінювали значення E_r або E_z або E_θ вихідного матеріалу (рис. 5.20). Виявлено, що для такого навантажування найбільше значення D в небезпечній зоні (кінець БС, $y \approx 20$ мм) зменшують матеріали з $E_z=20$ ГПа ($D_{\min}=2,03$) та $E_z=10$ ГПа ($D_{\min}=2,18$). Зі збільшенням E_z втомна міцність різко знижується (дві нижні криві на рис. 5.20). Найменше значення D зменшують матеріали з $E_\theta=20$ ГПа ($D_{\min}=2,27$) та $E_\theta=10$ ГПа

($D_{\min}=2,27$). Зміна значення E_{θ} майже не впливає на D (на рис. криві майже збігаються). Незначний вплив на значення D мають також матеріали з $E_r=20$ ГПа ($D_{\min}=2,22$) та $E_r=10$ ГПа ($D_{\min}=2,24$).

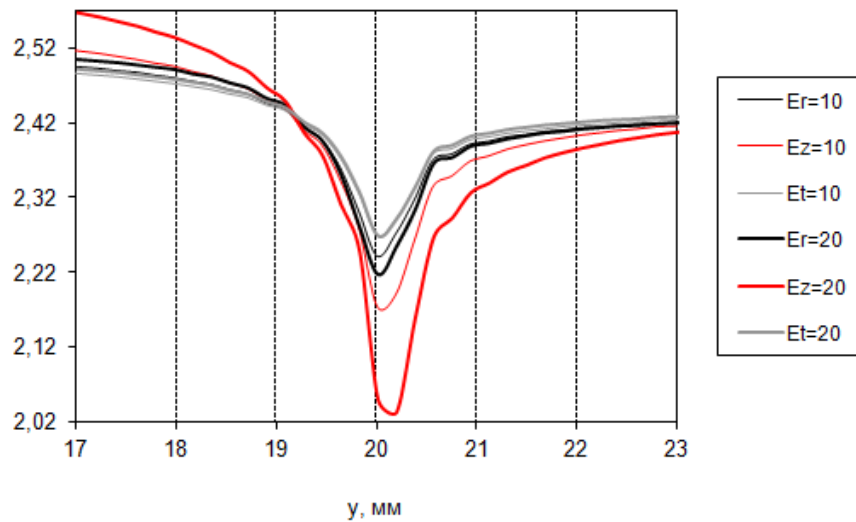
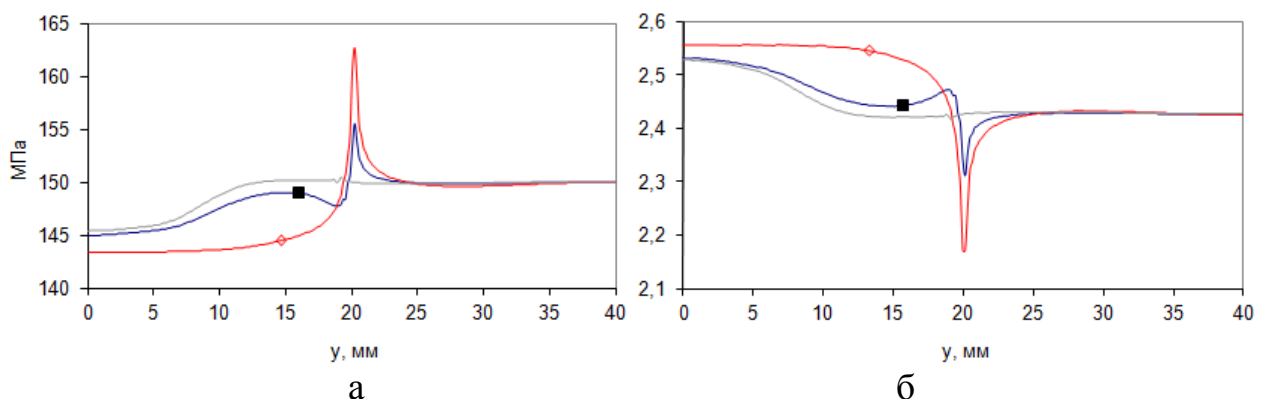


Рисунок 5.20 – Розподіл D на поверхні ШН по координаті y для різних значень модулів пружності БС E_r , E_z , E_{θ} (ГПа)

Наступні результати були отримані для БС з такими механічними характеристиками: $E_r=4,1$ ГПа, $E_{\theta}=2,1$ ГПа, $E_z=11$ ГПа, $\nu=0,28$. Виявлено, що відсутність фаски спричиняє напруження $\sigma_m=170,8$ МПа в перехідній зоні (що відповідає теоретичному ККН $170,8/150=1,139$) та $D=2,05$. Еліптична фаска (варіант конструкції "е") з $a=32$ мм спричиняє напруження $\sigma_m=159,5$ МПа (ККН= $159,5/150=1,063$) та $D=2,25$. На рис. 5.21 показані розподіли σ_m та D на поверхні сталеві деталі для варіантів "а", "г", "е" (рис. 5.18). Залежності для варіантів конструкції "а" та "е" мають математичну сингулярність в перехідній зоні, в якій різко погіршується втомна міцність. Варіант конструкції "г" майже її немає.



(\diamond) – варіант конструкції "а"; (\blacksquare) – "е" з $a=32$ мм; (—) – "г"

Рисунок 5.21 – Розподіл σ_m (а) та D (б) на поверхні ШН по координаті y

Оптимальним значенням може бути $a=20$ мм, оскільки для $a>20$ мм значення σ_m і D майже не змінюються (рис. 5.22).

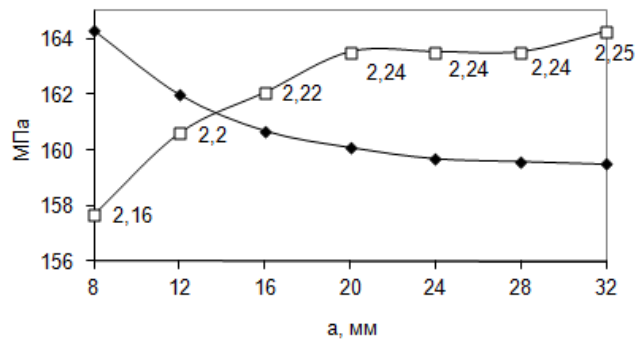
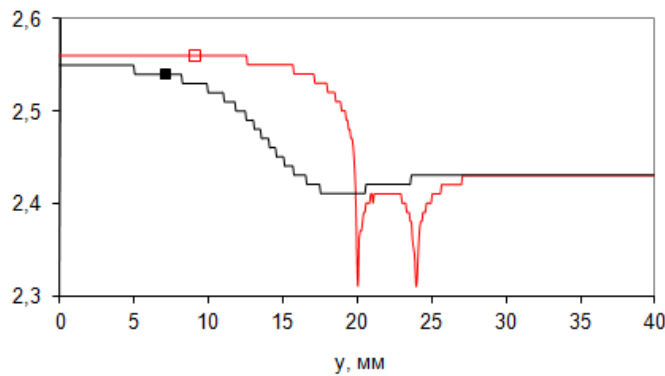


Рисунок 5.22 – Залежність σ_m і D від a для варіанта конструкції "е"

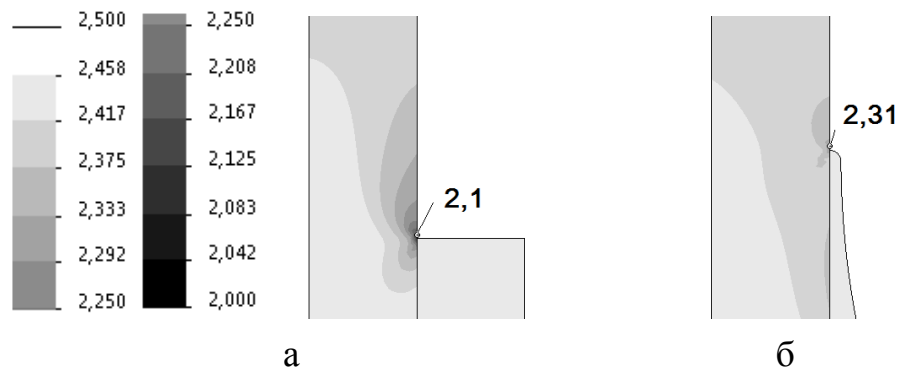
Варіант "г" володіє можливістю пошкодження БС в вузькому місці. Це місце можна захистити, для прикладу, шляхом нанесення матеріалу з меншим значенням E . Варіант "ж" містить фаску з ПКМ (епоксидна смола з $E=2,415$ ГПа, $\nu=0,35$) і забезпечує задовільні значення D (рис. 5.23). Значення D можна зменшити шляхом поєднання варіантів "е" ($a=20$ мм) та "ж". Значення D для такого варіанту "з" (2,41) майже рівне значенню D тіла ШН без бандажа (2,43) (рис. 5.23).



(□) – варіант конструкції "ж"; (■) – "з"

Рисунок 5.23 – Розподіл D на поверхні ШН по координаті y

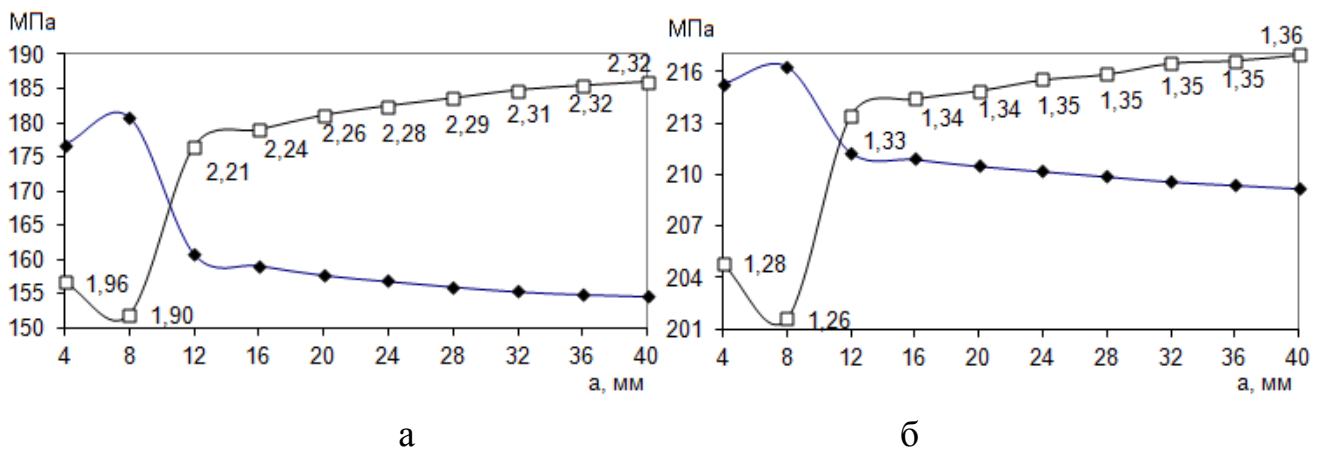
Для НКТ 73x5,5 з БС без фаски 5,5 мм, циклом $\sigma_p = 0 \dots 150$ МПа та внутрішнім тиском $P=0$ МПа значення D рівне 2,11 (рис. 5.24а). Наявність еліптичної фаски (варіант "е" з $a=32$ мм) збільшує це значення до 2,31 (рис. 5.24б).



а) – без фаски; б) – варіант фаски "е" з $a=32$ мм

Рисунок 5.24 – Розподіл D для циклу $\sigma_p=0\dots150$ МПа та тиску $P=0$ МПа

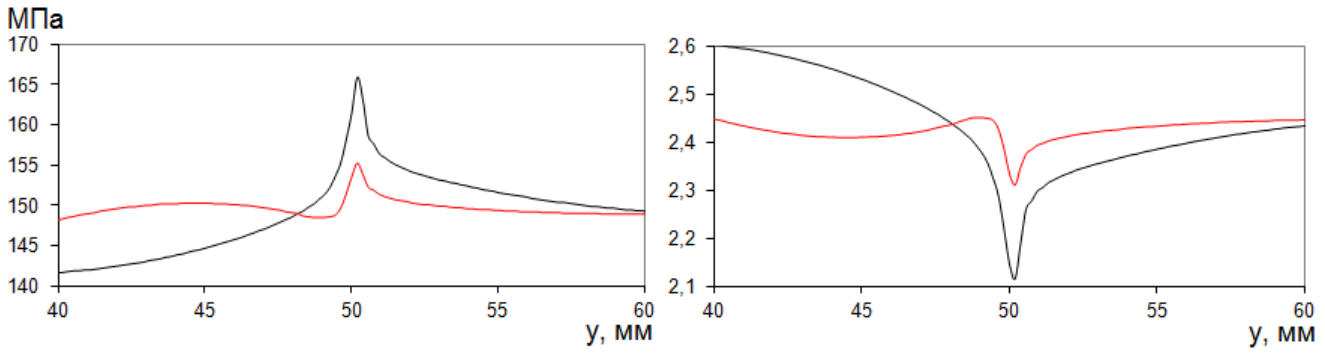
На рис. 5.25 показані залежності σ_m та D у зоні концентрації напружень на зовнішній поверхні НКТ від довжини a для циклу $\sigma_p=0\dots150$ МПа та значень $P=0$ МПа і $P=45$ МПа. Якщо $P=45$ МПа, то значення D майже удвічі зменшується. Оптимальні значення a знаходяться в інтервалі 28...36 мм.



а) – $P=0$ МПа; б) – $P=45$ МПа; (♦) – σ_m (МПа); (□) – D

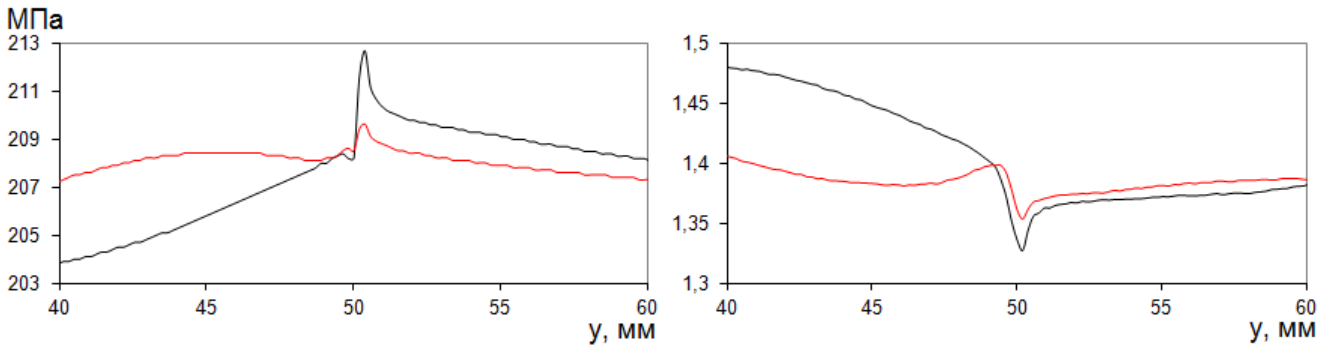
Рисунок 5.25 – Залежність σ_m та D на зовнішній поверхні НКТ від a для циклу $\sigma_p=0\dots150$ МПа

На рис. 5.26 показано розподіл напружень σ_m та D в зоні концентрації напружень на зовнішній поверхні НКТ для конструкції без фаски "а" та з фаскою "е" ($a=32$ мм). Така фаска значно збільшує значення D . Проте щоб згладити напруження повністю потрібно застосовувати варіант "з". Циклічна зміна внутрішнього тиску $P=0\dots45$ МПа незначно впливає на значення D в НКТ з БС (рис. 5.26 д). Наявність фаски збільшує значення D з 1,41 до 1,42.



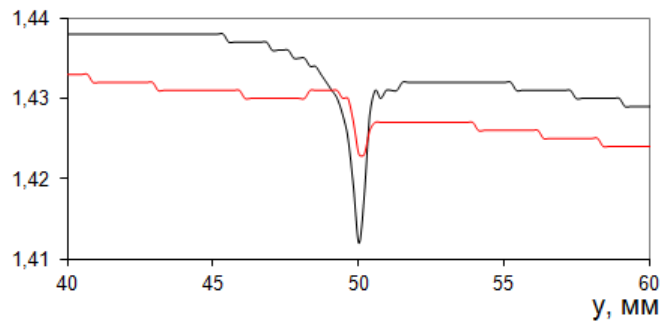
а

б



в

г



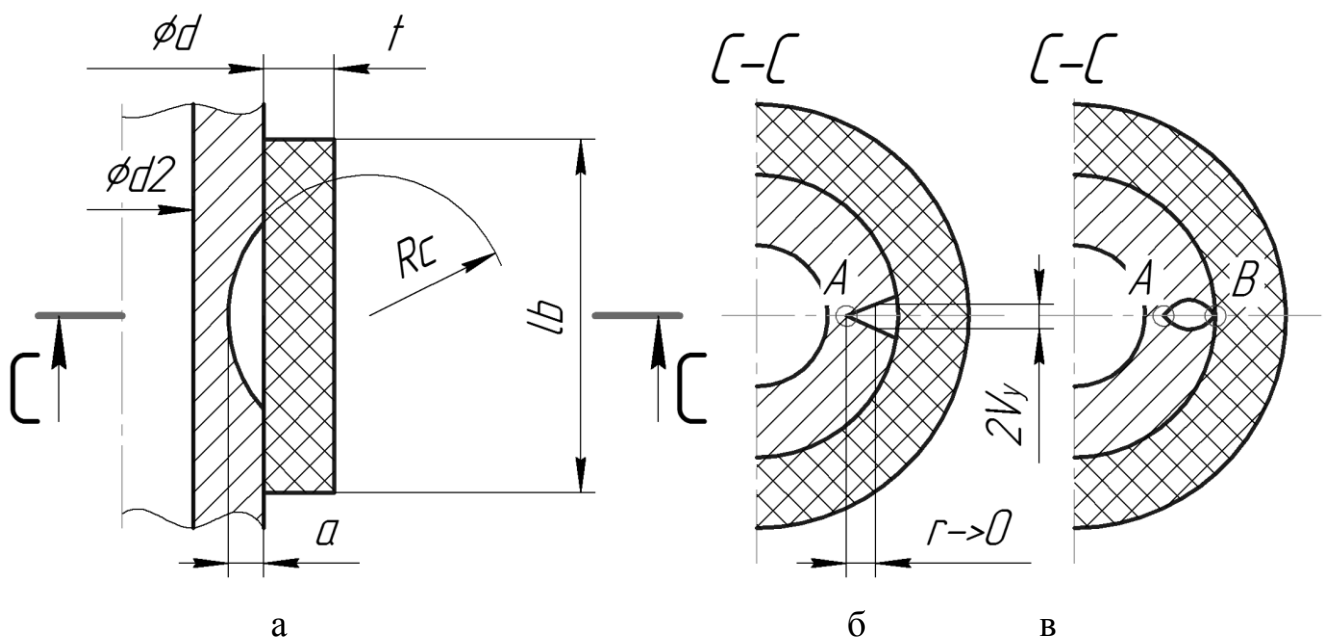
д

а, б) – $P=0$ МПа; в, г) – $P=45$ МПа;б, г) – для циклу $\sigma_p=0 \dots 150$ МПа; д) – для циклу $P=0 \dots 45$ МПа;(—) – без фаски; (—) – варіант фаски "е" з $a=32$ ммРисунок 5.26 – Розподіл σ_m (а, в) та D (б, г, д) на поверхні НКТ по координаті y

5.2.4 Прогнозування ресурсу НКТ з тріщиною і бандажем за допомогою коефіцієнта інтенсивності напружень

В працях [4, 179] виконано скінченно-елементний аналіз ремонту труб діаметром 273 мм з зовнішньою осьовою тріщиною БС. Виявлено, що збільшення натягу БС значно зменшує значення КІН в трубі. Збільшення товщини БС не суттєво зменшує значення КІН [4]. Проте обчислення циклічної довговічності та оптимізації БС виконано не було.

З метою опису методики аналізу розглянемо параметричну геометричну модель і СЕМ НКТ з зовнішньою осьовою тріщиною і БС в SOLIDWORKS (рис. 5.27). Зовнішній діаметр НКТ $d=73$ мм, внутрішній діаметр $d_2=63$ мм, товщина стінки $t_2=5,5$ мм. За замовчуванням товщина БС $t=5,5$ мм, довжина $lb=100$ мм. Для прикладу розглянуто тріщину, яка має круговий фронт з постійним радіусом $Rc=50$ мм та змінну глибину a . Геометричну модель тріщини будували за допомогою засобу SOLIDWORKS *SplitLine*. Характеристики матеріалу НКТ: $E=2,1 \cdot 10^{11}$ Па, $\nu=0,28$, $G=7,9 \cdot 10^{10}$ Па. Для БС: $E_1=4,1 \cdot 10^9$ Па, $E_0=11 \cdot 10^9$ Па, $E_z=2,1 \cdot 10^9$ Па, $\nu=0,28$.



б) – без адгезії ("no penetration"); в) – з адгезією ("bonded")

Рисунок 5.27 – Параметри НКТ з зовнішньою тріщиною і БС (а) та види моделювання контакту поверхонь труби і БС (б, в)

З метою зменшення обчислювальної трудомісткості задачі побудовано тільки одну четверту частину тривимірної СЕМ і створено граничні умови симетрії. В зоні вершини тріщини розмір елементів сітки близький до 0,2 мм. На НКТ діє циклічний внутрішній тиск $P=0...10$ МПа.

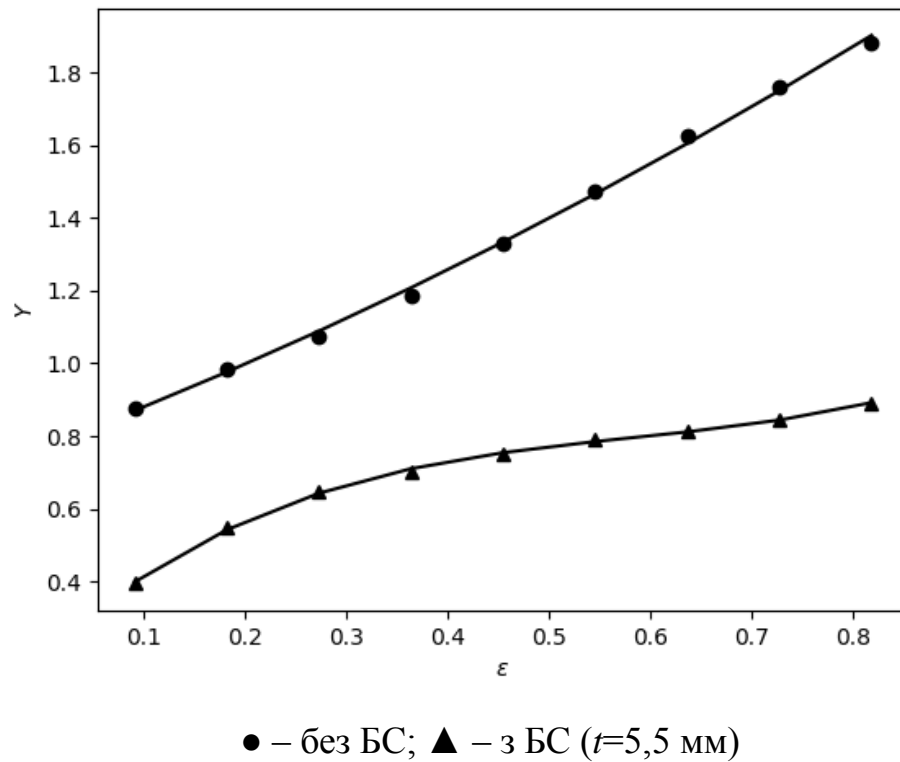
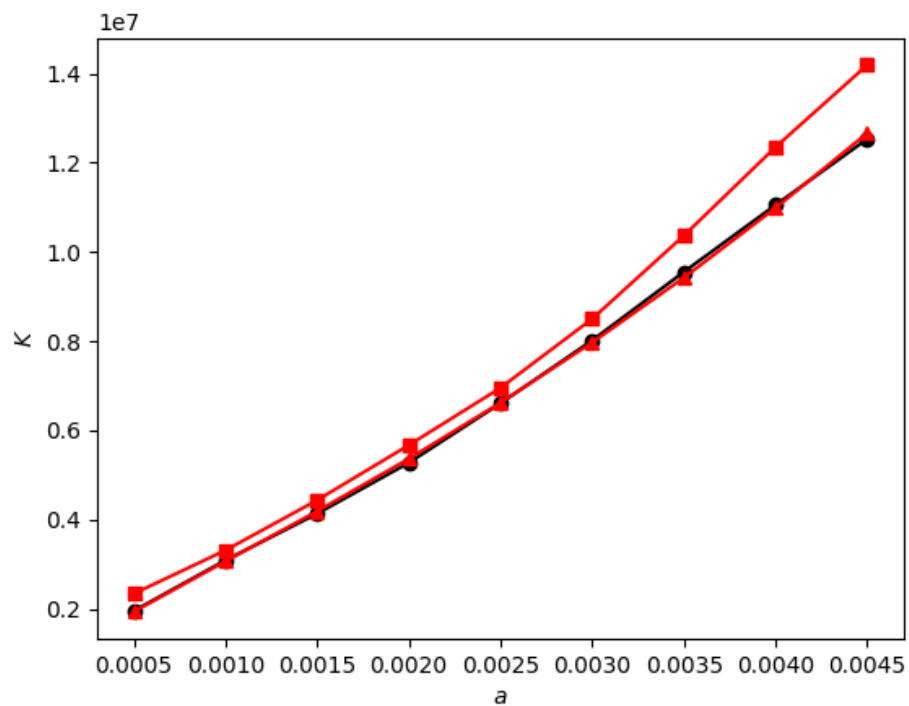
Використовували два види моделювання контакту поверхонь сталевого тіла і БС в SOLIDWORKS Simulation – без їхньої адгезії (вид контакту "no penetration") (рис. 5.27б) та з їхньою адгезією (вид контакту "bonded") (рис. 5.27в). На рисунку символом А позначено вершину тріщини в сталевій трубі, а символом В – вершину тріщини в БС. Якщо моделювати адгезію труби і бандажа (що більше відповідає реальній конструкції), то в точці В виникатиме концентрація напружень, яка може стати причиною росту втомної тріщини в БС. Отримано розподіли σ_m в зоні тріщини для цих видів контакту та значень $t=1$ мм і $t=5,5$ мм. У випадку контакту без адгезії в точці В не виникає сингулярності напружень, а збільшення t не суттєво зменшує значення σ_m . Для контакту з адгезією в точці В значення σ_m різко зростають. В цьому випадку значення КІН в БС можна розрахувати за допомогою прямого методу напружень (1.12) з екстраполяцією [209, 210].

За допомогою FEA SOLIDWORKS Simulation та її засобу автоматизації задач Design Study отримано залежності переміщення V_y (рис. 5.27) біля вершини тріщини в сталевому тілі від a і t . Значення КІН в сталевому тілі обчислювали прямим методом переміщень [211] за формулою (1.13). Після цього за допомогою регресійного аналізу знаходили відповідні значення функції Y (1.15), яка враховує вплив геометрії та залежить від параметра $\varepsilon=a/t_2$ (рис. 5.28). Функції $Y(\varepsilon)$ для НКТ без БС і з БС ($t=5,5$ мм):

$$Y = 0,395842 \varepsilon^2 + 1,05833632 \varepsilon + 0,77045447; R^2 = 0,9982.$$

$$Y = 2,15618837 \varepsilon^3 - 3,76444884 \varepsilon^2 + 2,47821731 \varepsilon + 0,20332406; R^2 = 0,9989.$$

На рис. 5.29 показані залежності КІН від глибини тріщини, отримані за формулою (1.14) МСЕ, емпіричною формулою (1.16), та емпіричною формулою з джерела [341].

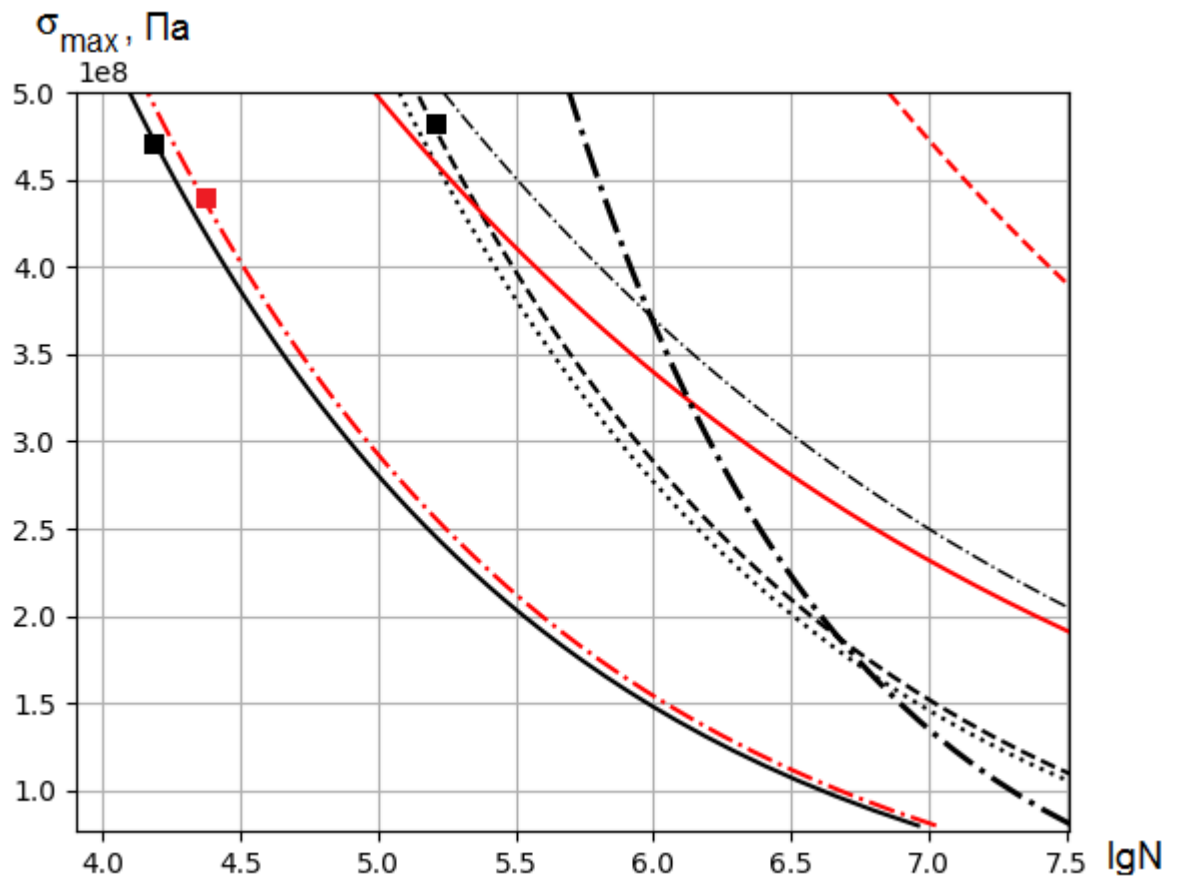
Рисунок 5.28 – Графіки функцій $Y(\epsilon)$ і відповідні емпіричні точкиРисунок 5.29 – Залежність КІН (Па·м^{0,5}) від a (м)

Остання дає незначне відхилення, що дещо збільшується зі збільшенням a . Це можна пояснити відмінністю фронтів тріщини – в праці [341] розглядалась еліптична тріщина.

Довговічність НКТ з тріщиною N обчислювали за (1.18) методом числового інтегрування. Значення $V(\Delta K)$ отримували з рівняння Періса (1.17). Використовували кінетичні діаграми втомного руйнування сталі 20Н2М. Параметри рівняння Періса для умов середовища 3% розчин NaCl: $C=9,14 \cdot 10^{-13}$ м/(МПа·м^{0,5}), $n=3,6$ [59]. Для повітря: $C=6,0 \cdot 10^{-17}$ м/(МПа·м^{0,5}), $n=6,04$ [59]. Для обчислення КІН та побудови кривих втоми (рис. 5.30) за результатами моделювання МСЕ створено програму (лістинг С.1).

На рис. 5.30 показані криві втоми $\sigma_{\max}(\lg N)$ гладких зразків зі сталі 20Н2М та криві росту тріщини від $a=0,5$ до $a=4,5$ мм для $R=0$. Для побудови кривих втоми гладких зразків використані рівняння кривих втоми $N = \sigma_R^m N_G / \sigma^m$ та дані їхніх випробувань [59] для $R = -1$. Для випробування в 3% NaCl $\sigma_R = \sigma_{-1} = 60$ МПа, $N_G = 2 \cdot 10^7$ циклів, $m=2,3$. Для випробування на повітрі $\sigma_R = \sigma_{-1} = 190$ МПа, $N_G = 2,5 \cdot 10^6$ циклів, $m=5,86$. Для обчислення границі витривалості σ_0 для віднульового циклу ($R=0$) використовували залежність (1.3), де $\psi_\sigma = 0,2$. Так як σ_0 – це максимальне напруження циклу та $\sigma_a = \sigma_m = \sigma_0 / 2$ то з $\sigma_0 / 2 = \sigma_{-1} - \psi_\sigma \sigma_0 / 2$ отримуємо $\sigma_0 = 2\sigma_{-1} / (\psi_\sigma + 1)$.

За рис. 5.30 циклічна довговічність в 3% NaCl, що відповідає пульсаціям внутрішнього тиску 0-10МПа ($\sigma_{\theta\max} = 56$ МПа), для НКТ без БС $N(\sigma_{\theta\max}) = 40$ млн. циклів, для НКТ з БС $t=5,5$ мм $N(\sigma_{\theta\max}) = 557$ млн. циклів. Порівняння двох нижніх кривих показує, що крива для труби з тріщиною, отримана без моделювання адгезії, незначно відрізняється від кривої для труби з тріщиною без БС. Отже адгезія поверхонь БС і труби важлива для забезпечення циклічної довговічності НКТ з тріщиною. Додатково, адгезія поверхонь забезпечує ізоляцію тріщини від середовища і реальна довговічність НКТ з БС буде значно вищою (див. криву з БС $t=5,5$ мм на повітрі). Завдяки ефекту зміцнення, що створює БС, ця довговічність навіть вища довговічності гладких зразків, які випробовувались на повітрі.



- (■—) – без БС в %3 водному розчині NaCl;
- (■- -) – без БС на повітрі (—); з БС $t=5,5$ мм в %3 NaCl;
 - (- -) – з БС $t=5,5$ мм на повітрі;
 - (· · ·) – з БС $t=0,5$ мм в %3 NaCl;
- (■- · -) – з БС $t=5,5$ мм в %3 NaCl, без моделювання адгезії;
- (- · -, жирна) – гладкий зразок в %3 NaCl [59];
- (- · -, тонка) – гладкий зразок на повітрі [59]

Рисунок 5.30 – Криві втоми гладких зразків зі сталі 20H2M та криві росту тріщини від $a=0,5$ до $a=4,5$ мм в НКТ

Необхідно також забезпечити тріщиностійкість БС в зоні В. Значення в'язкості руйнування (критичного КІН) K_{Ic} склопластику знаходяться в межах 10-50 МПа·м^{0,5}, що дещо менше значень для сталей [342]. За джерелом [343] це 17 МПа·м^{0,5}. Порогове значення КІН [344] знаходяться в межах 7-10 МПа·м^{0,5}. З рис. 5.31 видно, що тріщина з $a=4,5$ мм в НКТ не створюватиме в БС значення КІН, які більші ніж 0,5 МПа·м^{0,5}, за умови $t > 1$ мм. Отже збільшення t незначно підвищує циклічну довговічність сталевого тіла НКТ з тріщиною, але суттєво зменшує КІН в БС.

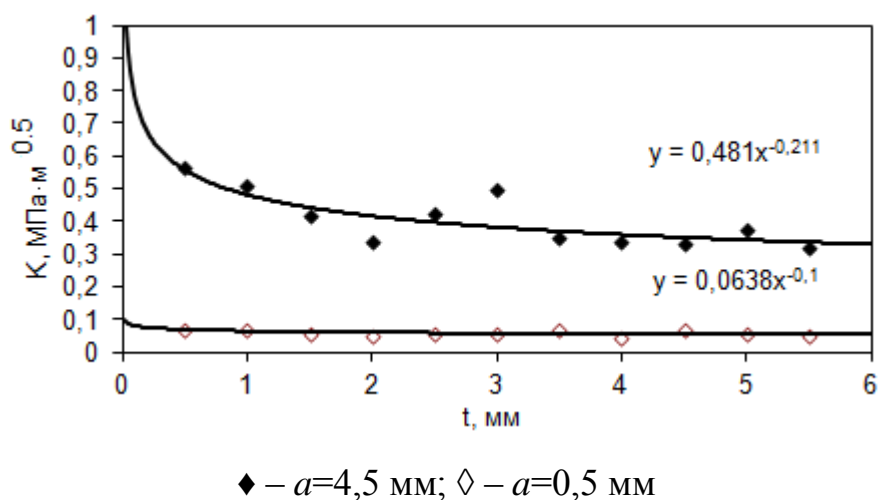
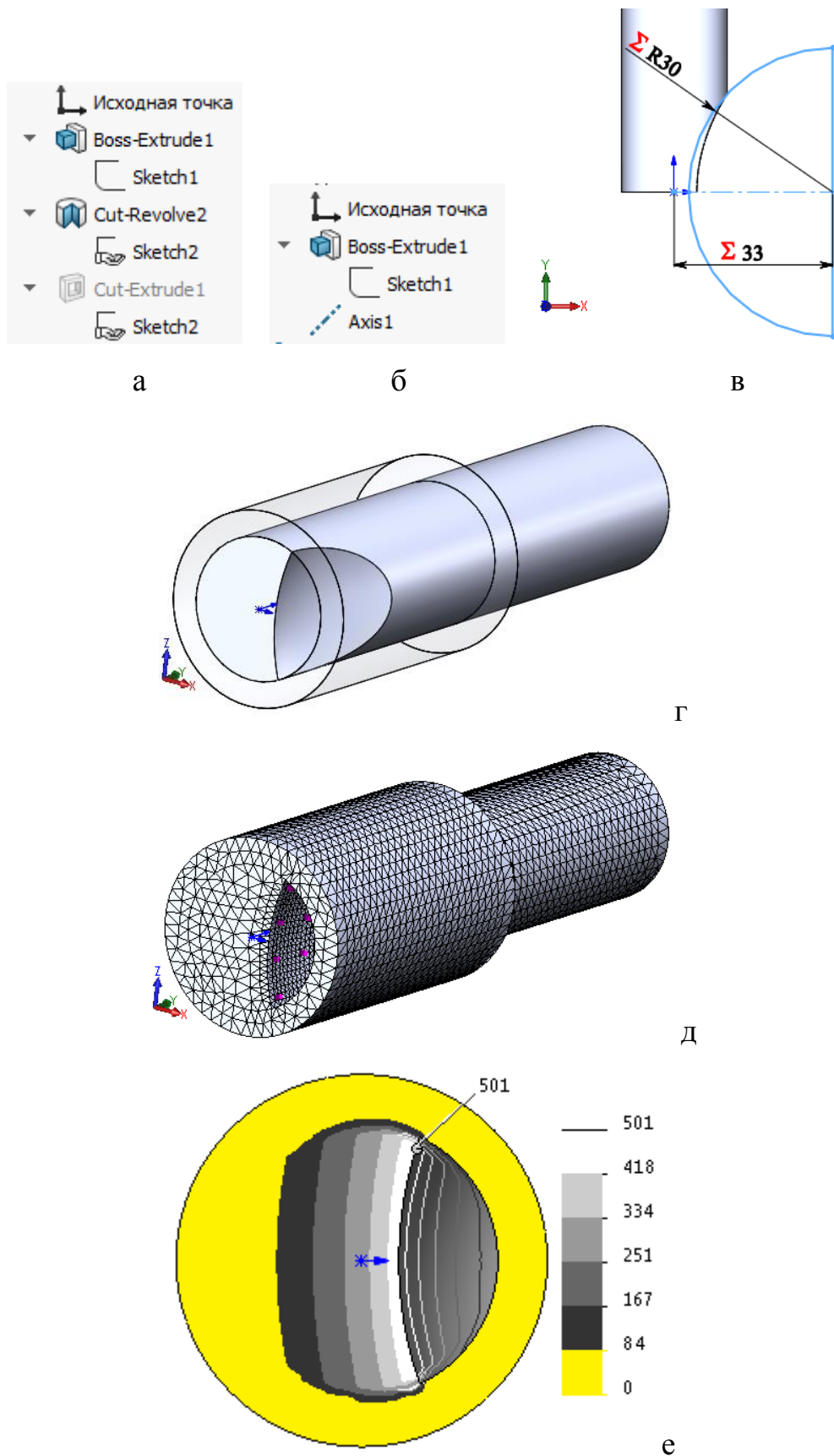


Рисунок 5.31 – Залежність КІН в БС від t

5.2.5 Вплив радіуса і глибини сферичного дефекту на напруження в штанзі

Геометричну параметричну модель ШН з дефектом будували в САПР SOLIDWORKS. Для побудови моделі корозійного сферичного дефекту будували ескіз з півколом в площині XY (рис. 5.32) і виконували ним круговий виріз Cut-Revolve. Модель передбачає також дефекти циліндричного типу, які моделюються за допомогою операції вирізу (Cut-Extrude) циліндричним тілом, вісь якого перпендикулярна до осі ШН. Для зменшення обчислювальної трудомісткості МСЕ будували половину ШН з дефектом і створювали граничні умови симетрії. Дерево побудови і модель показані на рис. 5.32, а параметри моделі – в табл. 5.4.



а, б) – дерева побудови моделі тіла з дефектом (а) та БС (б); в) – ескіз дефекту; г) – геометрична модель; д) – сітка СЕМ; е) – напруження σ_m (МПа) в зоні дефекту

Рисунок 5.32 – Модель ШН з круговим дефектом та БС

Таблиця 5.4 – Параметри SOLIDWORKS-моделі ШН з БС

Назва параметра	Рівняння	Значення	Пояснення
d	22	22 мм	діаметр ШН
db	30	30 мм	діаметр БС
lb	50	50 мм	півдовжини БС
h	8	8 мм	глибина дефекту
r	30	30 мм	радіус дефекту
$D1@Sketch1@Rod<1>.Part$	$= "d"$	22 мм	діаметр ШН
$D1@Sketch2@Rod<1>.Part$	$= "r"$	30 мм	радіус дефекту
$D2@Sketch2@Rod<1>.Part$	$= "d"/2 + "r" - "h"$	33 мм	міжосьова відстань
$D1@Boss-Extrude1@Band<1>.Part$	$= "lb"$	50 мм	півдовжини БС
$D1@Boss-Extrude1@Rod<1>.Part$	$= 2 * "lb"$	100 мм	півдовжини ШН
$D1@Sketch1@Band<1>.Part$	$= "d"$	22 мм	внутр. діаметр БС
$D2@Sketch1@Band<1>.Part$	$= "db"$	30 мм	діаметр БС

Для дослідження впливу радіуса R і глибини дефекту h на напруження в штанзі застосовано засіб автоматизації SOLIDWORKS Design Study, який використовує статичні задачі, що розв'язуються MCE в модулі SOLIDWORKS Simulation. Характеристики матеріалу ШН: $E=210$ ГПа, $\nu=0,28$, $G=79$ ГПа. Для БС: $E_r=4,1$ ГПа, $E_0=11$ ГПа, $E_z=2,1$ ГПа, $\nu=0,28$. На вільному торці ШН створювали навантаження розтягу, що відповідає $\sigma_p=150$ МПа. Результати обчислень для ШН з БС і ШН без БС наведено в табл. 5.5. В ШН з БС помітне зменшення напруження.

За цими даними будували регресійні залежності σ_m від параметрів h і R (рис. 5.33). За допомогою методу найменших квадратів та функції `curve_fit` з пакету SciPy залежність шукали у вигляді:

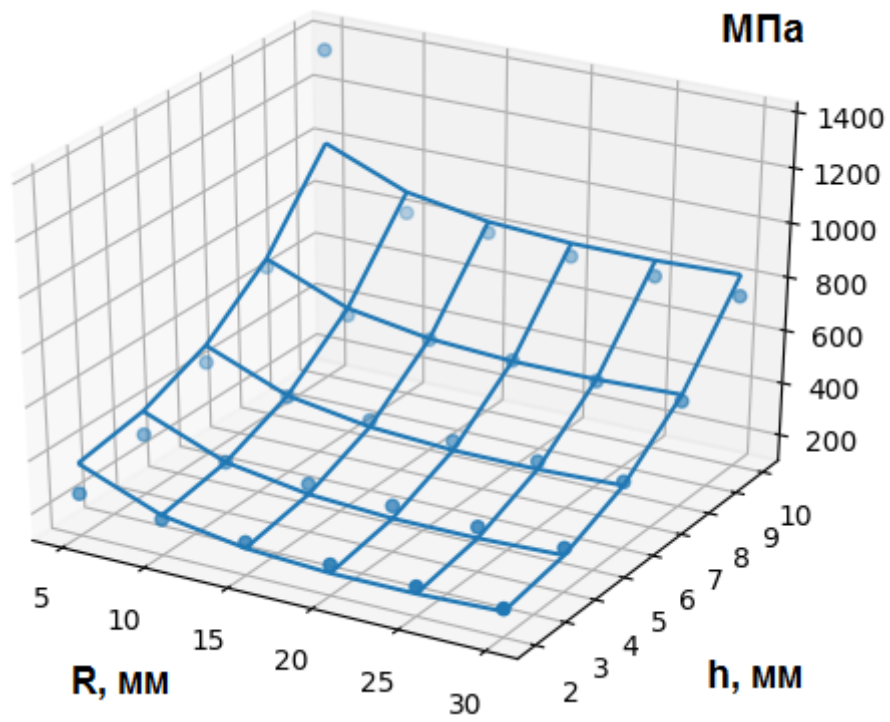
$$\sigma_M = a + bR + cR^{0,1} + dh + eh^3.$$

Таблиця 5.5 – Залежність максимальних значень σ_M в зоні дефекту (МПа) від h та R (мм) в ШН 22 мм для $\sigma_p=150$ МПа

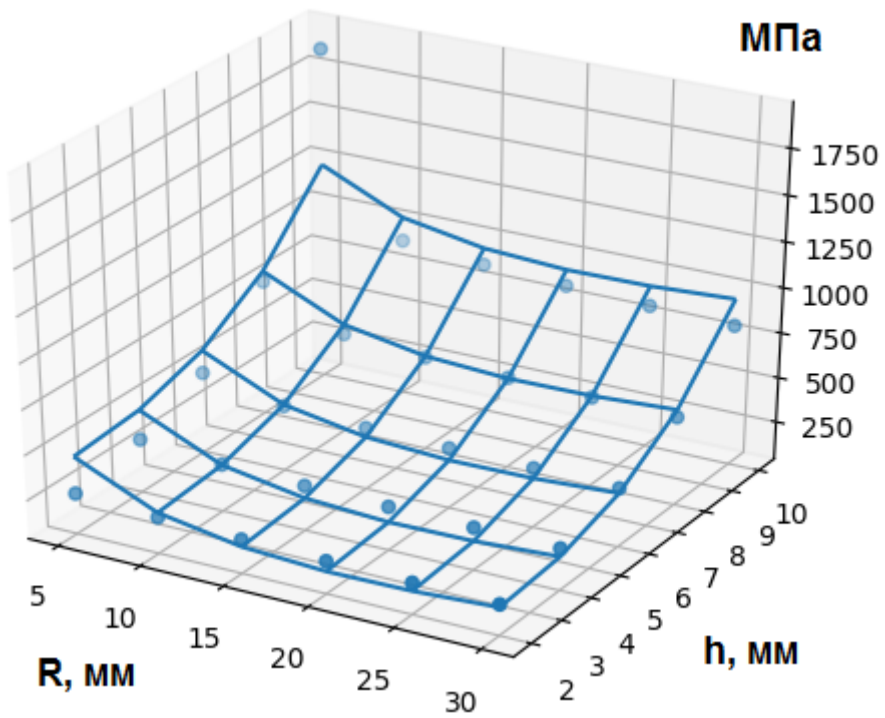
		З БС $db=30$ мм і $2lb=100$ мм						Без БС					
h	R	5	10	15	20	25	30	5	10	15	20	25	30
	2	259	236	224	216	211	207	263	240	227	219	214	210
	4	327	293	283	275	270	266	334	300	288	280	275	271
	6	451	389	369	362	357	353	486	401	382	373	368	365
	8	670	553	527	516	507	501	786	588	557	543	536	530
	10	1349	802	790	765	755	746	1870	905	865	843	828	817

Значення степенів членів полінома підбирали таким чином, щоб забезпечити високі значення коефіцієнта детермінації R^2 . Знайдено значення параметрів для моделі з БС (рис. 5.33а): $a=2974$, $b=14$, $c=-2285$, $d=4,393$, $e=0,599$, $R^2=0,94$. Для моделі без БС (рис. 5.33б): $a=4897$, $b=24,69$, $c=-3868$, $d=-7,506$, $e=0,8364$, $R^2=0,88$. Знайдені залежності можуть бути використані для обчислення $ККН$ (σ_{\max} / σ_p , де $\sigma_p=150$ МПа) в ШН з БС і без нього.

Для значень параметрів $R=20$ мм, $h=6$ мм, $db=30$ мм, $lb=50$ обчислено значення коефіцієнта запасу втомної міцності D в штанзі з БС і без нього (рис. 5.34). Найменші значення D (світла лінія) в ШН без БС спостерігаються в середній частині дефекту (на рис. внизу), тоді як в ШН з БС вони також спостерігаються на кромці дефекту і в зоні закінчення БС. В найбільш небезпечній середній частині дефекту значення D у ШН з БС є більшими (рис. 5.34 в).



а

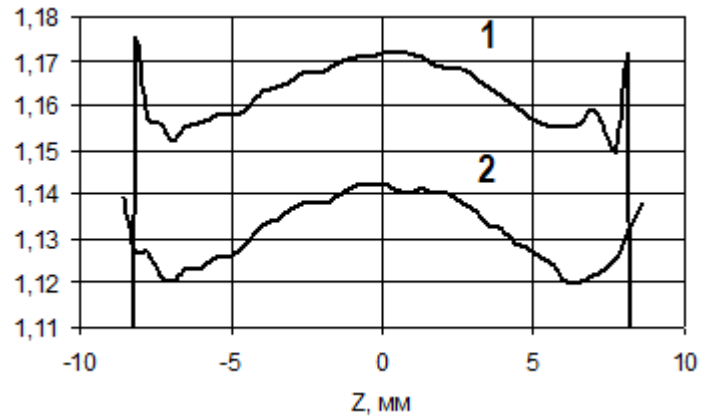
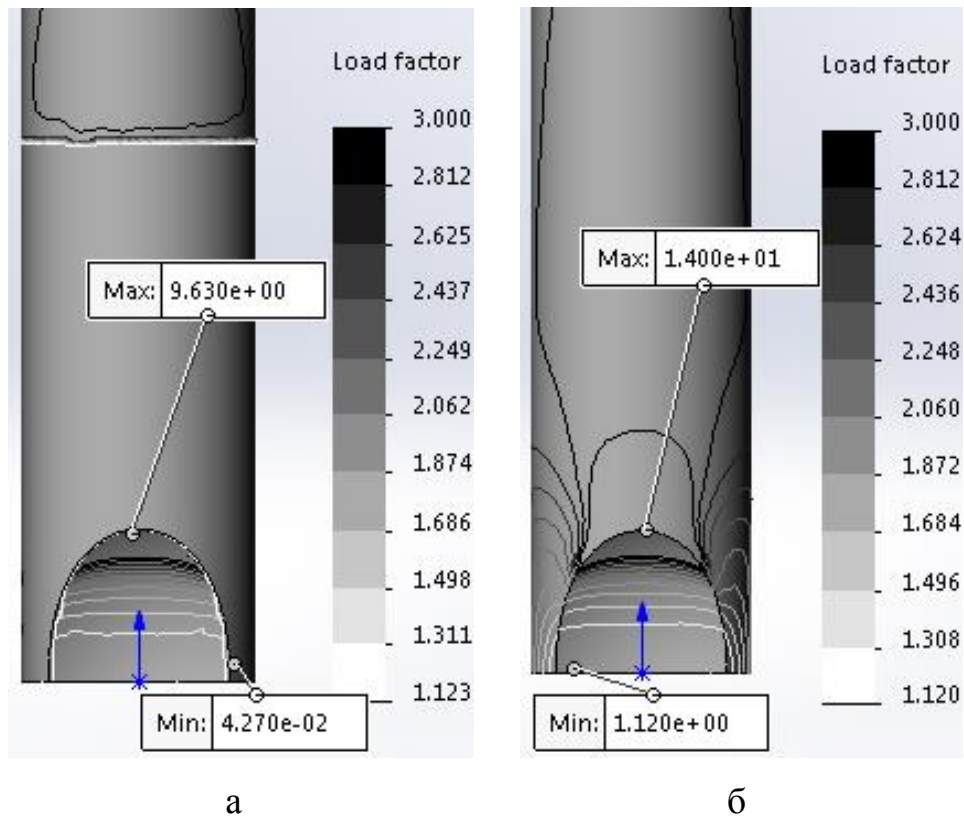


б

а) – з БС; б) – без БС

Рисунок 5.33 – Регресійні залежності максимальних σ_m від R та h в зоні дефекту

ШН



а) – з БС; б) – без БС

в) – з БС (1) та без БС (2)

Рисунок 5.34 – Значення коефіцієнта запасу D в штанзі (а, б)
та вздовж Z -координати в середній частині дефекту (в)

5.2.6 Вплив діаметра і довжини бандажа на напруження в штанзі зі сферичним дефектом

За допомогою SOLIDWORKS Design Study виявлено вплив діаметра db і довжини lb БС на напруження σ_m в ШН з сферичним дефектом для значень $R=20$ мм, $h=6$ мм, $\sigma_p=150$ МПа (табл. 5.6).

Таблиця 5.6 – Залежність максимальних σ_m в ШН з сферичним дефектом (МПа) від db та lb (мм)

$db \backslash lb$	22	26	30	34	38	42
20	372	375	368	360	353	347
30	372	373	366	358	350	349
40	372	372	365	357	349	341
50	372	370	364	356	348	341
60	372	370	363	356	348	340

Регресійну залежність шукали у вигляді:

$$\sigma_m = a + b \cdot db^{0,1} + c \cdot db^{0,2} + d \cdot db^{0,3} + e \cdot lb^1 + f \cdot lb^2 + g \cdot lb^3$$

Значення коефіцієнтів (рис. 5.35): $a=-161675,195$, $b=341716,598$, $c=-239879,229$, $d=56050,1370$, $e=0,463537305$, $f=-0,0174011266$, $g=0,000158055024$. Коефіцієнт детермінації $R^2=0,741$.

Значення db , які більші 40 мм, і значення lb , які більші 50 мм, не значно зменшують напруження (рис. 5.35), тому значення $db=40$ мм та $lb=50$ мм можуть бути оптимальними. Залежність може бути використана для обчислення ККН (σ_m / σ_p) в ШН з БС і граничних навантажень на них. Для прикладу для $db=40$ мм, $lb=50$ мм знаходимо $\sigma_m(db=40, lb=50)=345$ МПа, $ККН=345/150=2,3$. Тоді для допустимого напруження $[\sigma]=200$ МПа робочі напруження розтягу повинні бути

не більші $\sigma_p = [\sigma] / KKH = 200 / 2,3 = 87$ МПа. Для ШН без такого БС $\sigma_m = 372$ МПа; $KKH = 372 / 150 = 2,48$; $\sigma_p = [\sigma] / KKH = 200 / 2,48 = 80$ МПа.

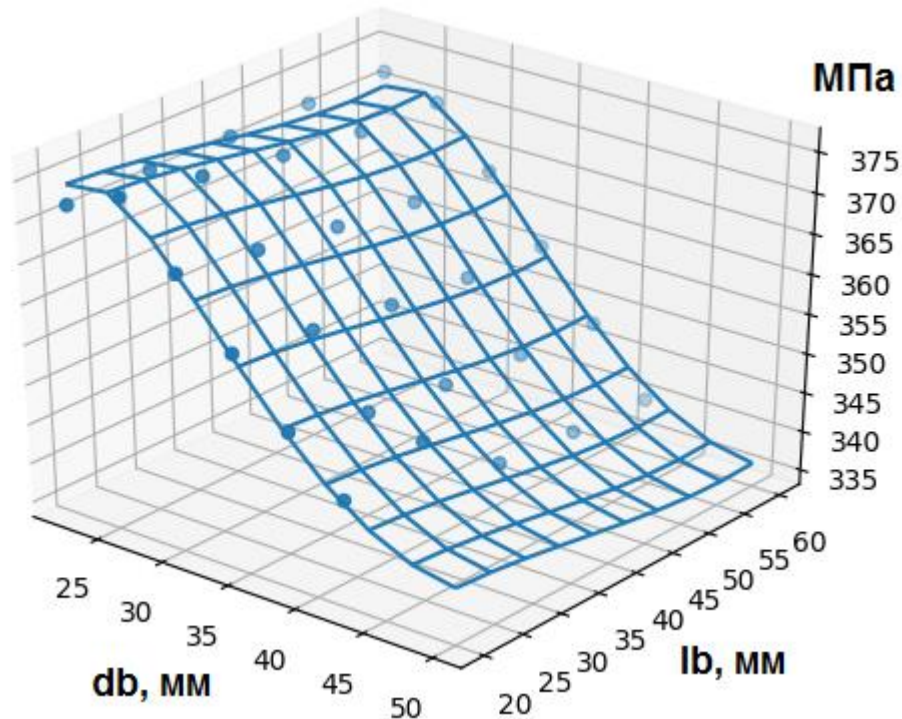


Рисунок 5.35 – Регресійна залежність максимальних σ_m в ШН з дефектом від db і lb

5.2.7 Прогнозування ресурсу штанги з тріщиною і бандажем за допомогою коефіцієнта інтенсивності напружень

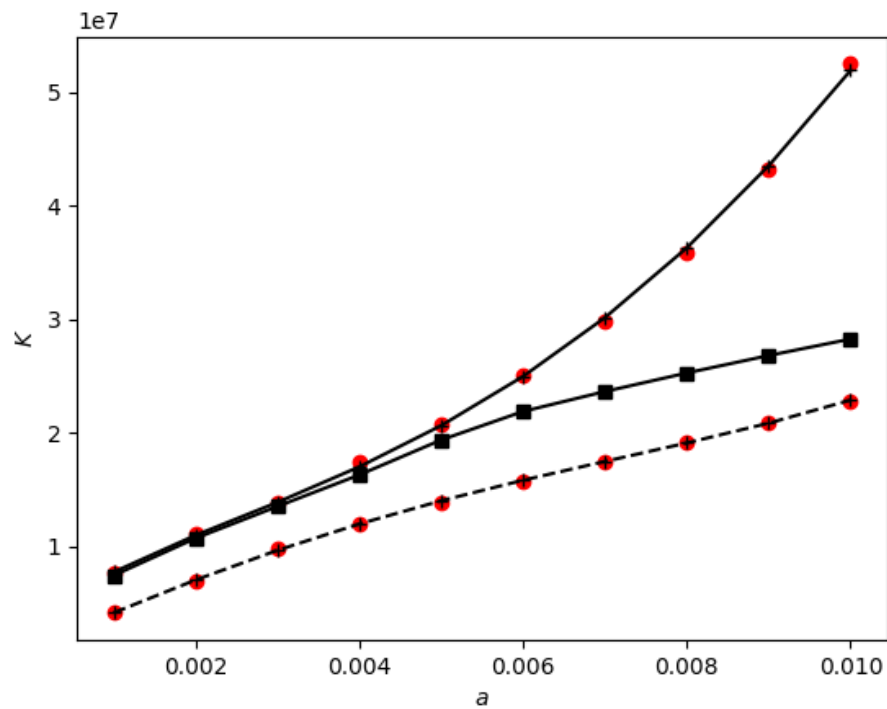
Під час навантажень розтягу і згину, в ШН може утворитись сегментна поперечна тріщина, яка розвивається в глибину тіла перпендикулярно осі стержня. Для обчислення КІН в такому стержні, існують декілька рішень [341, 345-347], отриманих різними методами (МСЕ, об'ємних сил, податливості, похідній жорсткості). Для спрощення задачі розглядали тріщину з прямолінійним фронтом. Значення КІН визначались за допомогою МСЕ з використанням

формули (1.13) [348]. Для швидкої зміни значень параметрів застосовувалась тривимірна параметрична модель ШН з тріщиною глибиною a . Характеристики матеріалу ШН $E=210$ ГПа, $\nu=0,28$, $G=79$ ГПа. Для БС: $E_r=4,1$ ГПа, $E_\theta=11$ ГПа, $E_z=2,1$ ГПа, $\nu=0,28$. Після обчислення значень КІН (рис. 5.36) знаходили відповідні значення поправочної функції Y (1.15), яка враховує вплив відносної глибини тріщини $\varepsilon=a/D$. За допомогою регресійного аналізу (лістинг С.1) знаходили функцію $Y(\varepsilon)$ для ШН без БС і з БС (рис. 5.37):

$$Y = 6,83070156\varepsilon^2 - 0,89912424\varepsilon + 0,95087371, R^2 = 0,998;$$

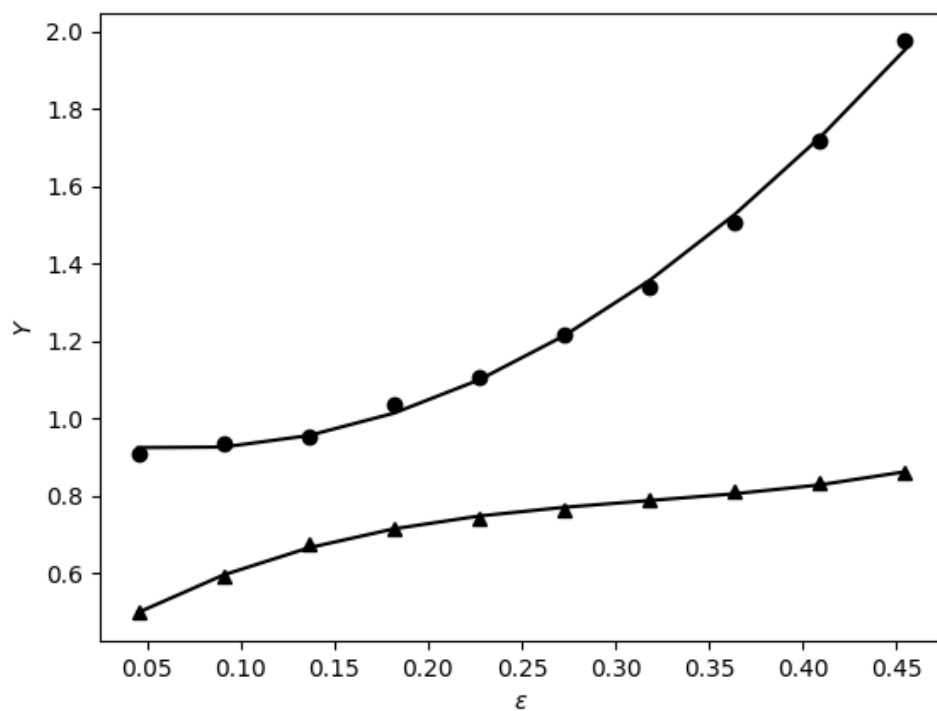
$$Y = 9,47189249\varepsilon^3 - 9,00886033\varepsilon^2 + 3,22132507\varepsilon + 0,3692415, R^2 = 0,997.$$

Обчислення циклічної довговічності N ШН з тріщиною виконували за формулою (1.18) і використовували метод числового інтегрування. Значення швидкості росту $V(\Delta K)$ отримували з залежності Періса (1.17). Середовище – 3% NaCl (параметри діаграми втомного руйнування: $C=9,14 \cdot 10^{-13}$ м/(МПа·м^{1/2})ⁿ, $n=3,6$). Отримано криві росту тріщини $\sigma_{\max}(\lg N)$ від $a=1$ мм до $a=10$ мм) для віднульового циклу навантажування ШН діаметром 22 мм зі сталі 20Н2М (рис. 5.38). Помітно, що ремонт БС підвищує циклічну довговічність N ШН з тріщиною з $10^{4,7}$ до $10^{5,5}$ циклів для $\sigma_{\max}=300$ МПа та середовища 3% NaCl. Теоретична довговічність за умови відсутності доступу до середовища рівна 10^7 . Однак експериментальна довговічність бездефектних ШН з БС $t=0,5$ мм менша $10^{6,6}$ [126] (дещо більша ніж для випробування гладких зразків на повітрі – $10^{6,5}$). Це пояснюється проникненням середовища в зону тріщини на кінцевих стадіях її росту внаслідок руйнування БС. Отже можна вважати, що довговічність ШН з тріщиною $a=1$ мм і БС в 3% NaCl буде в межах $10^{5,5}$ - $10^{6,6}$ циклів. Це відповідає збільшенню довговічності в 6,3-79 раз.



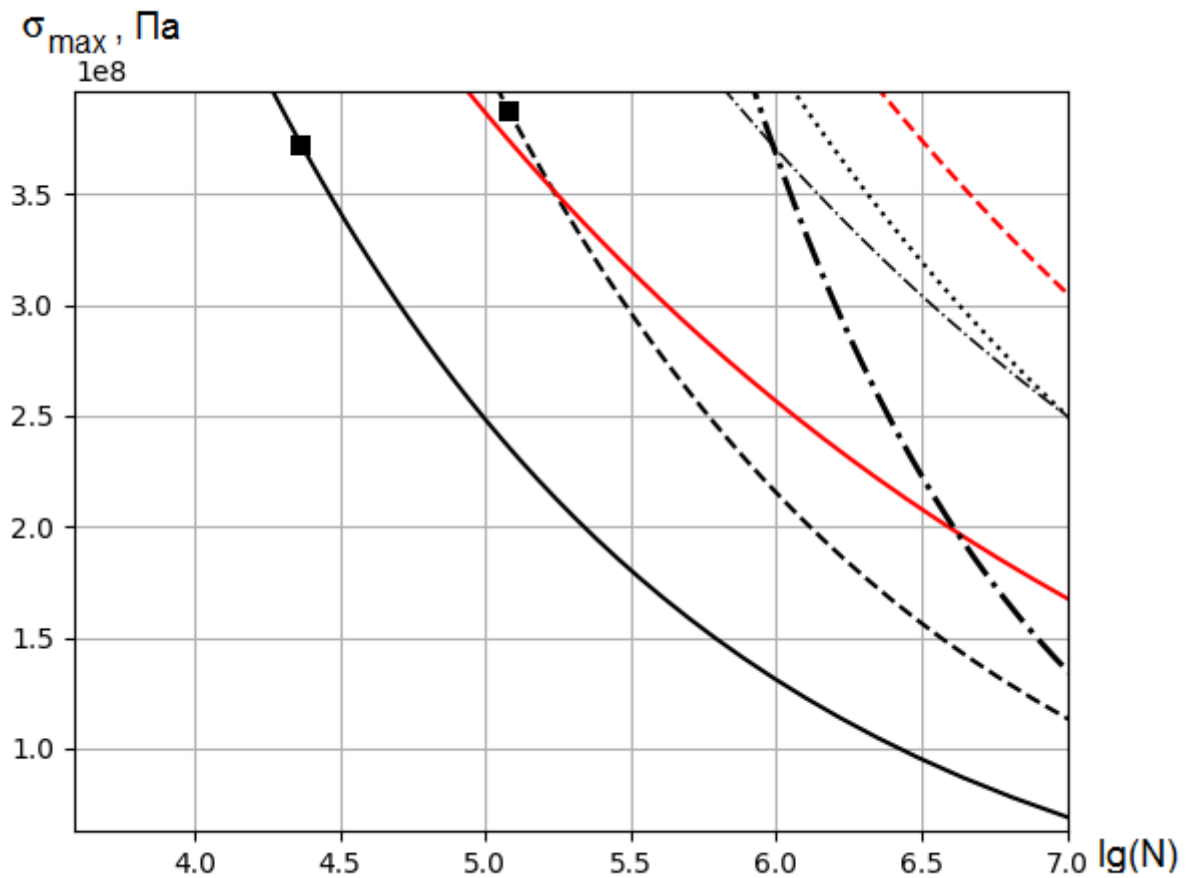
● – емпіричні значення; + – теоретичні значення за формулою (1.16); ■ – теоретичні значення для тріщини з еліптичним фронтом $a/z=0,5$ (z – велика піввісь, використання формули коректне для $a < 5,5$ мм) [341]; (—) – без БС; (- -) – з БС

Рисунок 5.36 – Залежність КІН ($\text{Па}\cdot\text{м}^{0,5}$) від a (м)



(●) – без БС; (▲) – з БС

Рисунок 5.37 – Значення поправочної функції $Y(\epsilon)$



(■—) – без БС в %3 водному розчині NaCl;

(—) – без БС на повітрі;

(■- -) – з БС $t=4$ мм в %3 NaCl;

(- -) – з БС $t=4$ мм на повітрі;

(· · ·) – гладкий зразок зі сталі 15НЗМА з БС $t=0,5$ мм в %3 NaCl [126];

(- · -, жирна) – гладкий зразок в %3 NaCl [59];

(- · -, тонка) – гладкий зразок на повітрі [59]

Рисунок 5.38 – Криві втоми гладких зразків зі сталі 20Н2М та криві росту тріщини від $a=1$ мм до $a=10$ мм в ШН діаметром 22 мм:

За результатами випробування на симетричний згин в %3 NaCl отримано рівняння Гудмена для уживаної ШН зі сталі 20Н2М з комплексним дифузійно-цинковим покриттям з БС ($\sigma_{\max}=138R+205$, МПа) і для такої ж ШН без покриття ($\sigma_{\max}=243R+100$, МПа) [349, 350]. Границя витривалості σ_0 збільшується в 2 рази. Для нових штанг на повітрі: $\sigma_{\max}=93R+250$, МПа. Для нових штанг у %3 NaCl: $\sigma_{\max}=203R + 140$ (МПа) [59]. Відповідні залежності з рис. 5.38: $\sigma_{\max}=223R+120$ (МПа) (нижня штрихова крива) та $\sigma_{\max}=283R+60$ (МПа) (нижня суцільна крива). Границя витривалості σ_0 збільшується в 2 рази.

БС можуть також бути використані для зміцнення ШН без дефектів в найбільш небезпечних зонах, зокрема в зоні піделеваторного бурта, в якій нерідко з'являються втомні тріщини.

5.3 Скінченно-елементне моделювання та аналіз втомної міцності насосних штанг в зоні скруглення між піделеваторним буртом і тілом штанги

Для обґрунтування доцільності збільшення радіуса скруглення між піделеваторним буртом і тілом ШН r необхідно отримати залежність коефіцієнта запасу D від радіуса r та згинального навантаження σ_{zz} . Для цього на базі Abaqus 6.11 розроблено СЕМ частини ШН в зоні піделеваторного бурта та макрос мовою Python для автоматизації процесу зміни параметрів, перебудови моделі, симуляції та отримання результатів [351]. Обчислення втомної міцності виконували в fe-safe® 6.2 – програмному комплексі для аналізу втоми матеріалів, який використовує результати моделювання напружено-деформованого стану МСЕ.

Для нормальної роботи елеватора протектори-центратори і БС не можна установлювати безпосередньо біля піделеваторного бурта. Потрібно забезпечити між ними мінімальну відстань 300-400 мм. В таких умовах не рідкий згин і втома ШН в цій зоні, що підтверджено промисловими даними [44]. Розрахункова схема моделі показана на рис. 5.39. Тут 1 – муфта штангова, 2 – ШН, 3 – внутрішня поверхня НКТ, 4 – центратор для ШН. Геометричні параметри моделі відповідають ГОСТ 13877-96 для штанги ШН22. Вибирались такі допустимі

значення розмірів згідно ГОСТ, які утворюють найбільш небезпечну конструкцію з точки зору втомної міцності. Радіус r змінювали наступним чином: 67 мм (мінімально допустимий згідно ГОСТ), 250, 500, 750, 1000 мм. Матеріал ШН – сталь 40. Для розрахунку втомної міцності в базі даних fe-safe був вибраний її аналог – SAE1040. Модель матеріалу пружна ($E=200$ ГПа, $\nu=0,33$). Симулювались два кроки навантаження, які утворюють один цикл навантаження під час роботи колони ШН.

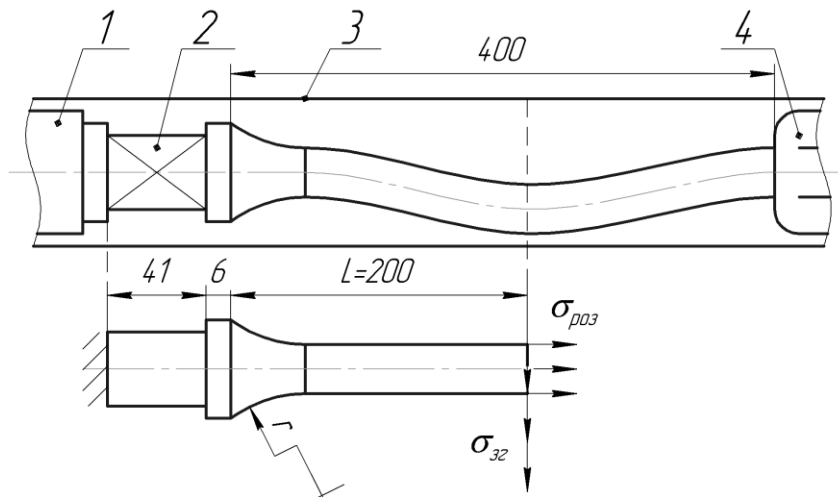


Рисунок 5.39 – Розрахункова схема

Перший крок відповідає ходу колони ШН уверх. Осьове навантаження розтягу утворює напруження в тілі ШН $\sigma_{p03}=170$ МПа. Це відповідає максимальному допустимому напруженню циклу для сталі 40 згідно ГОСТ 13877-96 за умови, що мінімальне напруження циклу рівне нулю. Навантаження згину відсутнє. Другий крок відповідає ходу колони ШН униз. Осьове навантаження відсутнє. Навантаження згину моделювалось дотичним напруженням до перетину тіла ШН:

$$\sigma_{32} = \frac{F_{32}}{0,25\pi d^2},$$

де F_{32} – згинаюча сила, d – діаметр тіла ШН.

Напруження σ_{32} змінювали від 1 до 3 МПа з кроком 0,5 МПа. Для кожної пари значень r та σ_{32} за допомогою fe-safe розраховували циклічну довговічність,

коефіцієнт запасу D для 10^7 циклів та відсоток відмов після $5 \cdot 10^6$ циклів (за умови мінливості навантаження в межах 10% від заданого значення). В даному випадку під циклічною довговічністю N_f слід розуміти кількість повторів повних циклів втомного навантажування до моменту утворення втомної тріщини, а під відмовою слід розуміти появу втомної тріщини [202]. Для обчислення N_f та D використовували критерій Брауна-Міллера (1.11).

Результати розрахунку для стандартної штанги ШН22 з $r=67$ мм та удосконаленої штанги з $r=500$ мм для $\sigma_{32}=3$ МПа, $\sigma_{роз}=170$ МПа показані на рис. 5.40. Для стандартної штанги значення логарифму циклічної довговічності lgN_f , коефіцієнта D та відсотку відмов після $5 \cdot 10^6$ циклів відповідно дорівнюють 6,176; 0,733; 72,16, а для удосконаленої ШН – 6,77; 0,924; 46,90. Тобто для удосконаленої ШН значення D збільшилося у 1,26 рази (рис. 5.41).

В програмі Maple виконано апроксимацію залежності D від σ_{32} та r наступним поліномом від двох змінних:

$$D = 1,3059 + 0,39 \cdot 10^{-3} \cdot r - 0,2 \cdot 10^{-6} \cdot r^2 - 0,1817 \cdot \sigma_{32} - 0,53 \cdot 10^{-2} \cdot \sigma_{32}^2 + 0,5 \cdot 10^{-4} \cdot r \cdot \sigma_{32}$$

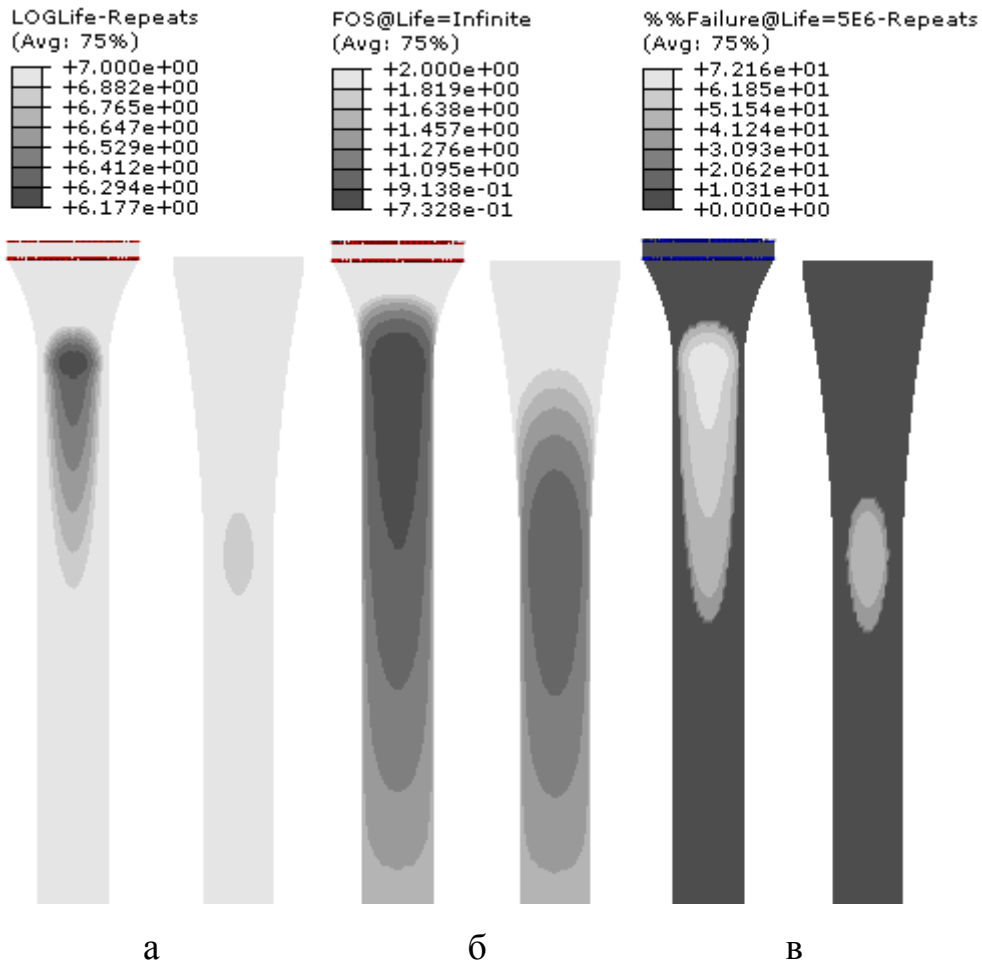
В графічному вигляді ця залежність показана на рис. 5.42.

Аналіз в fe-safe можна виконувати інтерактивно або в пакетному режимі. Наприклад, для запуску двох різних задач можна створити такий пакетний файл (bat файл):

```
start /wait fe-safe_cl.exe -s j=c:\1\Job-1.odb b=c:\1\my.stlx o=c:\1\results1.odb
start /wait fe-safe_cl.exe -s j=c:\1\Job-2.odb b=c:\1\my.stlx o=c:\1\results2.odb
```

Якщо задачу fe-safe необхідно виконати з макросу Abaqus CAE, то для запуску процесу fe-safe_cl.exe можна використати функцію subprocess.Popen():

```
s=r'd:\Program Files\Safe_Technology\fe-safe\version.6.2\exe\fe-safe_cl.exe
-s j=c:\1\{iodb}.odb b=c:\1\{istlx}.stlx o=c:\1\{oodb}.odb'
s=s.format(iodb=input_odb, istlx=input_stlx, oodb=output_odb)
subprocess.Popen(s).communicate() # виконує обчислення та чекає завершення
```



а) – $\lg N_f$; б) – D ; в) – відсоток відмов після $5 \cdot 10^6$ циклів

Рисунок 5.40 – Порівняння характеристик втомної міцності стандартної штанги ШН22 з $r=67$ мм (ліворуч) та удосконаленої ШН з $r=500$ мм (праворуч)

для $\sigma_{32}=3$ МПа, $\sigma_{роз}=170$ МПа

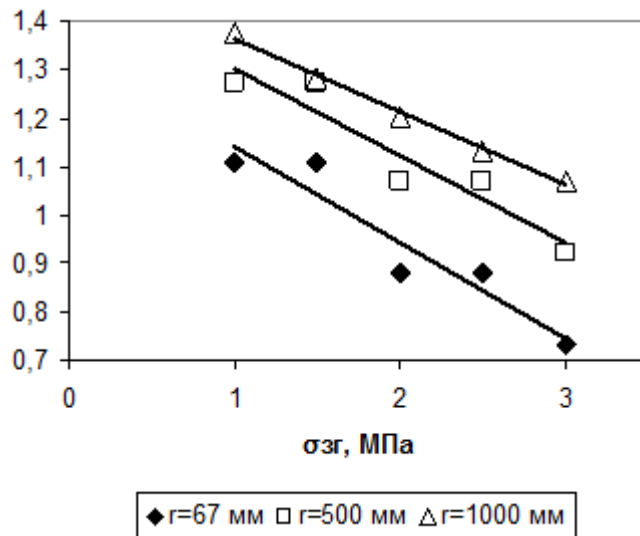


Рисунок 5.41 – Лінійна апроксимація залежностей D від σ_{32}

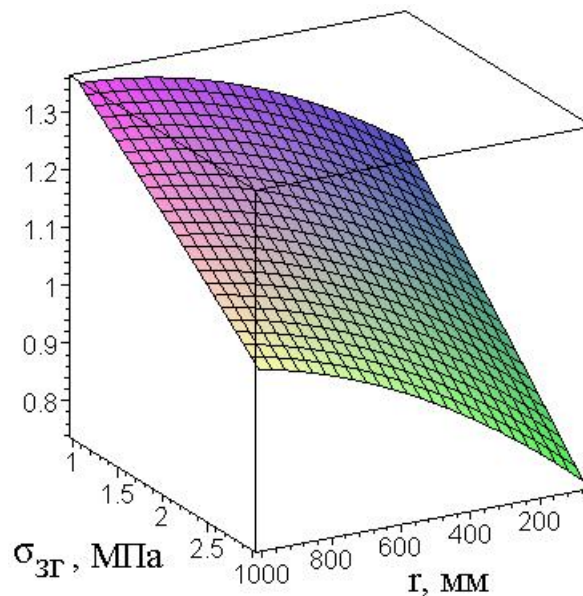


Рисунок 5.42 – Апроксимація залежності D від σ_{zr} та r поліномом

Макрос Abaqus CAE мовою Python для аналізу та оптимізації конструкції за критерієм втомної міцності за допомогою fe-safe наведено в додатку (лістинг Т.1). Тут `set_values()`, `mesh_all()`, `JobSubmit()`, `writeLDFfile()`, `runFeSafe()`, `readODB_set_()` – функції розроблені автором. Основний його алгоритм на псевдокодї:

відкрити csv-файл

FOR EACH r IN (0.067, 0.25, 0.5, 0.75, 1.0):

установити значення радіуса r , оновити модель, створити сітку елементів, виконати задачу

FOR EACH load IN (0.1, 0.15, 0.2, 0.25, 0.3):

записати значення навантаження згину load у LDF файл fe-safe, виконати задачу fe-safe, відкрити базу даних результатів, отримати $lg(Nf)$ у множині вузлів Set-1, знайти мінімальне значення, отримати D у множині вузлів Set-1, знайти мінімальне значення, отримати відсоток відмов у множині вузлів Set-1, знайти максимальне значення, записати дані у файл, закрити базу даних результатів

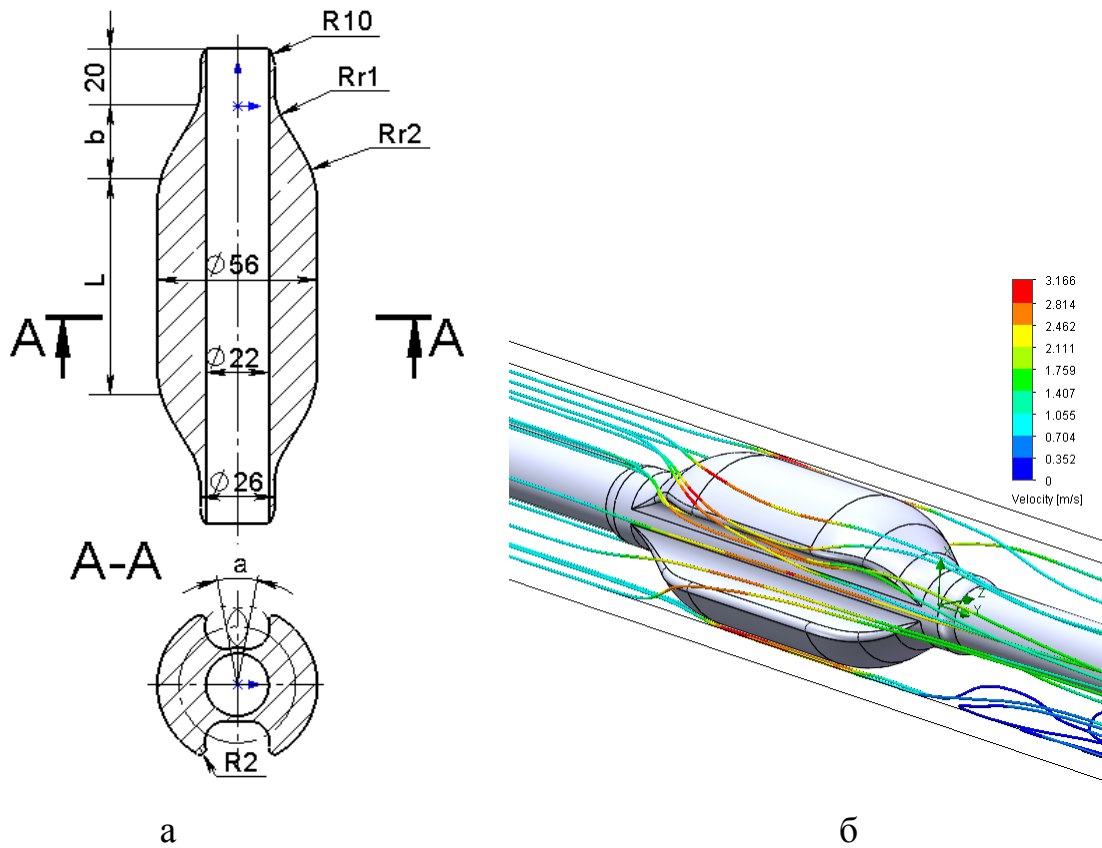
закрити csv-файл

Отже збільшення значення r розглянутої моделі до 500 мм призводить до збільшення коефіцієнта запасу D орієнтовно в 1,2 рази. Розроблені СЕМ та макрос можуть бути використані для аналізу та оптимізації ШН іншого типорозміру чи конструкції для будь-яких значень навантажень або властивостей матеріалів.

5.4 Гідродинамічна модель протектора для насосних штанг

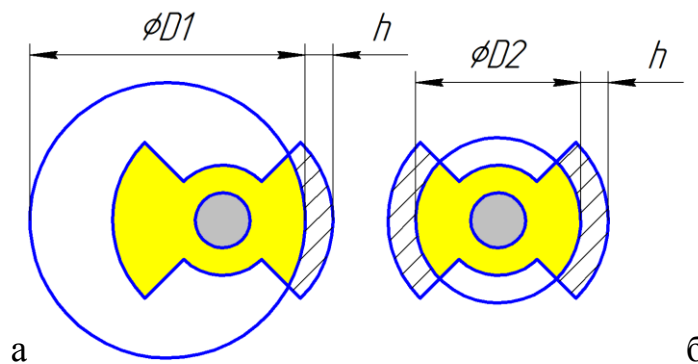
На основі аналізу конструкцій протекторів [21, 63] автором запропоновано конструкцію протектора, в якій поперечний перетин канавки являє собою сектор кільця з кутом a та максимально можливими радіусами скруглення по краях, а зовнішній діаметр цього сектора кільця дорівнює діаметру протектора (рис. 5.43а) [352]. Така конструкція дозволяє зменшити гідродинамічний опір та збільшити площу тертя з НКТ. Окрім того, для збільшення площі тертя такі протектори необхідно монтувати на штанзі з поворотом на 90 градусів один відносно іншого. Для комплексного забезпечення працездатності ШН автор рекомендує формування протектора на поверхні БС для їхнього додаткового захисту від механічного пошкодження і водопоглинання. В праці [63] виконано аналіз впливу параметрів такого чотирилопатевого протектора на силу гідродинамічного опору і площу тертя. Проте не запропоновано методики оптимізації параметрів та не виявлено оптимальних їхніх значень.

З метою виявлення оптимальних значень параметрів дволопатевого протектора, автором розроблено його тривимірну параметричну модель в SolidWorks® 2013 (посилання наведено в додатку У). Лопаті моделі можуть бути прямими (рис. 5.43а) або гвинтовими. Розрахунок сили гідродинамічного опору виконували MCE за допомогою SOLIDWORKS Flow Simulation 2013 (рис. 5.43б). Параметри гідродинамічної моделі: рідина – вода, початкова швидкість рідини – 1 м/с. Обчислення середньої площі тертя виконували за допомогою удосконаленого макросу для SOLIDWORKS [1]. Так як зі спрацюванням лопатей площа тертя може змінюватись (рис. 5.44), новий макрос (лістинг У.1) обчислює площі тертя для різних величин спрацювання і розраховує їхнє середнє значення.



а) – ескіз; б) – траєкторії потоку рідини

Рисунок 5.43 – Протектор з двома лопатями



а) – нерівномірне; б) – рівномірне; h – глибина спрацювання; $D1$ – постійний діаметр кола (діаметр НКТ) зі змінним центром; $D2$ – змінний діаметр кола з постійним центром

Рисунок 5.44 – Моделі спрацювання лопатей

Спрацювання лопатей під час застосування штангообертача буде рівномірним. На рис. 5.43а показані параметри протектора для ШН діаметром 22 мм та НКТ з внутрішнім діаметром 59 мм. В праці [352] для протектора з двома лопатями ($L=120$ мм та $a=50^\circ$) знайдено оптимальні значення параметрів гідродинамічного схилу за критерієм мінімального гідродинамічного опору: $b=18,67$ мм, $r1=33,24$ мм, $r2=24,15$ мм. Проте не було враховано критерій максимальної площі тертя.

Зважаючи на значну обчислювальну складність оптимізації використовували наступний ітераційний алгоритм:

1. Побудова плану експерименту.

2. Обчислення сили опору та середньої площі тертя для цього плану.

3. Побудова регресійної моделі $f(\mathbf{x})$ за узагальненим критерієм.

4. Пошук мінімуму $f(\mathbf{x})$ в заданих границях. Використовували популярний квазі-ньютонівський метод L-BFGS-B, який призначений для нелінійних задач з великою кількістю невідомих, з пакету SciPy.

5. Звуження границь в околі знайденого мінімуму, зменшення розміру сітки (якщо потрібно) та перехід до п.1

Були вибрані наступні змінні параметри: b , $r1$, $r2$, $L/2$, a . Для зручності позначимо їх як відповідні елементи вектора $\mathbf{x}=(x_1, x_2, x_3, x_4, x_5)$. На основі аналізу існуючих конструкцій були вибрані початкові границі їх значень (табл. 5.7). Для візуалізації (рис. 5.45) значення \mathbf{x} з основних границь масштабували до інтервалу $(-1,+1)$, а з розширених – до $(-1,41, +1,41)$.

Таблиця 5.7 – Границі значень параметрів

Границі		$b(x_1)$, м	$r1(x_2)$ м	$r2(x_3)$, м	$L/2(x_4)$, м	$a(x_5)$, рад
основні	<i>min</i>	0,02514	0,02159	0,02159	0,036591	0,37664
	<i>max</i>	0,03986	0,03841	0,03841	0,053409	0,67002
розширені	<i>min</i>	0,015	0,01	0,01	0,025	0,17444
	<i>max</i>	0,05	0,05	0,05	0,065	0,87222

Для оптимізації параметрів протектора використовували критерій мінімальної сили гідродинамічного опору y_1 (Н) та критерій максимальної середньої площі контакту лопаті зі стінкою НКТ y_2 (мм²). Отримані за результатами симуляції значення y_1 (та y_2), з використанням їхнього середнього арифметичного μ і стандартного відхилення σ , нормувались за формулою

$$y_{ni} = \frac{y_i - \mu}{\sigma}$$

таким чином, щоб вектор y_n мав середнє значення 0 та дисперсію 1.

Задачу оптимізації двома критеріями зводили до задачі з одним узагальненим критерієм. Для скаляризації використовували зважену суму

$$y = \omega_1 y_{n1} + \omega_2 y_{n2},$$

де ω_1, ω_2 – вага критеріїв.

Розглядали три випадки – гіпотезу однакової важливості критеріїв ($\omega_1=1, \omega_2= -1$), гіпотезу важливості мінімального гідродинамічного опору ($\omega_1=1, \omega_2=0$) та гіпотезу важливості максимальної площі тертя ($\omega_1=0, \omega_2= -1$). Для зменшення об'єму обчислень використовували методи планування експерименту. План експерименту – центральний композиційний план, кількість факторів – 5, кількість експериментів – 44 (таблиця У.1). Після обчислення значень у шукали функцію $f(\mathbf{x})$ шляхом апроксимації значень у поліномом (таблиця У.2). Для пошуку найкращого полінома застосовували метод групового урахування аргументів (МГУА) [144] та програмне забезпечення для його реалізації GMDH Shell 3 [293]. Використовували наступні налаштування алгоритму: псевдовипадковий спосіб перемішування спостережень, перехресна перевірка (кількість частин – 2), критерій відбору – середній квадрат помилки, ранжування

змінних – за кореляцією, основний алгоритм – комбінаторний, поліном може містити члени $x_i x_j$ та x_i / x_j (окрім x_i).

Результати оптимізації для $\omega_1=1, \omega_2=-1$ показані в табл. 5.9 та на рис. 5.45а. На рисунку показані масштабовані функції $f(x_i, \mathbf{x}^*)$ одного аргумента x_i ($i \in \{1,2,3,4,5\}$), тоді як значення інших аргументів \mathbf{x}^* відповідають мінімуму $\mathbf{x}^* \subset \arg \min f(\mathbf{x})$. Значення $f(\mathbf{x})$ найбільше зменшується в околі знайденого мінімуму у разі збільшення півдовжини протектора $x_i=L/2$, кута канавки $x_i=a$, довжини $x_i=b$ та у разі зменшення радіуса $x_i=r_1$. Радіус r_2 впливає незначно.


Обчислювальна складність гідродинамічної задачі та точність її результатів залежить від розміру (і кількості N) елементів скінченно-елементної сітки. Такі залежності можна використовувати для екстраполяції більш точних розв'язків. Наприклад з табл. 5.8 видно, що зі збільшенням N до 200 розв'язок наближається до значення 1,4.

Таблиця 5.8 – Залежність сили гідродинамічного опору y_l (Н) оптимальної моделі від кількості комірок базової скінченно-елементної сітки вздовж моделі N

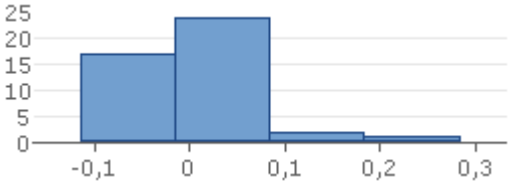
N	36	52	74	102	146
y_l	2,198	1,912	1,758	1,623	1,53

Результати оптимізації для $\omega_1=1, \omega_2=0$ показані в табл. 5.10 та на рис. 5.45б. Найбільше сила гідродинамічного опору зменшується в околі знайденого мінімуму у разі збільшення кута канавки a та у разі зменшення радіуса r_1 . Результати оптимізації для $\omega_1=0, \omega_2=-1$ показані в табл. 5.11 та на рис. 5.45в. Найбільше площа тертя в околі знайденого мінімуму зростає у разі збільшення довжини протектора L та зменшення кута канавки a .

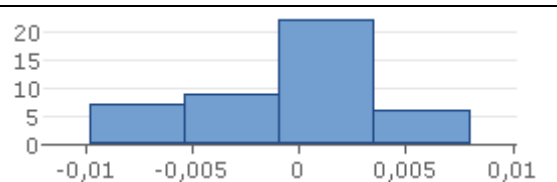
Таблиця 5.9 – Модель для $\omega_1=1, \omega_2 = -1$

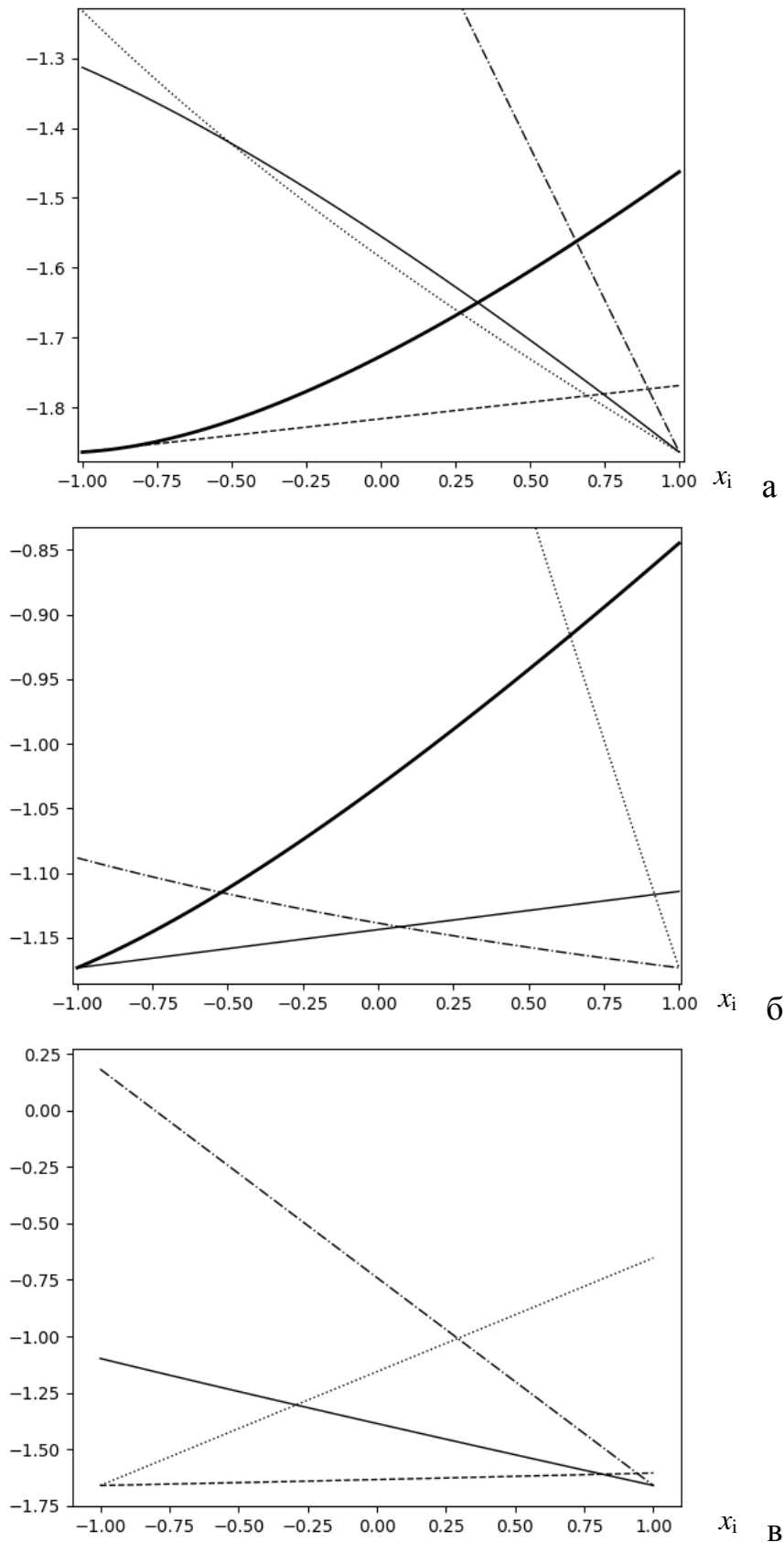
Назва	Значення	
	Навчання	Екзамен
Кількість спостережень	35	9
Максимальне від'ємне відхилення	-0,254638	-0,129206
Максимальне додатне відхилення	0,158218	0,223686
Середній модуль помилки	0,0720449	0,10199
Середньоквадратичне відхилення	0,0896377	0,120782
Сума відхилень	-5,33684E-13	0,138165
Стандартне відхилення залишків	0,0896377	0,119803
Коефіцієнт детермінації (R^2)	0,993222	0,991403
Коефіцієнт кореляції	0,996605	0,997034
Найкраща модель $f(\mathbf{x})$	$1,43243 - 1078,61 \cdot x_1 \cdot x_4 + 0,225551 \cdot x_3/x_1 - 0,470969 \cdot x_4/x_1 + 33,8941 \cdot x_2/x_5 + 0,0148706/(x_2 \cdot x_5) - 23,2641 \cdot x_4 \cdot x_5 - 22,3094 \cdot x_4/x_5$	
Гістограма відхилень (%)		
Важливість змінних \mathbf{x} (%)	(17,44; 21,43; 1,48; 77,90; 49,74)	
Мінімум $f(\mathbf{x})$ в точці: для осн. границь для розш. границь	(0,03986; 0,02159; 0,02159; 0,053409; 0,67002) (0,05; 0,02094605; 0,01; 0,065; 0,872222)	

Таблиця 5.10 – Модель для $\omega_1=1, \omega_2=0$

Назва	Значення	
	Навчання	Екзамен
Кількість спостережень	35	9
Максимальне від'ємне відхилення	-0,10566	-0,112759
Максимальне додатне відхилення	0,132103	0,283129
Середній модуль помилки	0,0365388	0,0799934
Середньоквадратичне відхилення	0,0466674	0,112436
Сума відхилень	-1,10165E-12	0,353775
Стандартне відхилення залишків	0,0466674	0,105341
Коефіцієнт детермінації (R^2)	0,997925	0,981733
Коефіцієнт кореляції	0,998962	0,99202
Найкраща модель $f(\mathbf{x})$	$3,21439 + 185,91 \cdot x_1 \cdot x_2 - 125,042 \cdot x_2 + 0,457499 \cdot x_2/x_4 + 94,7473 \cdot x_2/x_5 - 0,0832421 \cdot x_5/x_2 + 0,0429714/(x_2 \cdot x_5) - 3,62757/x_5$	
Гістограма відхилень (%)		
Важливість змінних \mathbf{x} (%)	(1,66; 27,73; 0; 3,38; 94,56)	
Мінімум $f(\mathbf{x})$ в точці: для осн. границь для розш. границь	(0,02514; 0,02159; -; 0,053409; 0,67002) (0,015; 0,01; -; 0,065; 0,872222)	

Таблиця 5.11- Модель для $\omega_1=0, \omega_2=-1$

Назва	Значення	
	Навчання	Екзамен
Кількість спостережень	35	9
Максимальне від'ємне відхилення	-0,00831414	-0,00973994
Максимальне додатне відхилення	0,00792095	0,0050289
Середній модуль помилки	0,00290339	0,00378398
Середньоквадратичне відхилення	0,00369975	0,00466711
Сума відхилень	-1,83566E-12	-0,0111444
Стандартне відхилення залишків	0,00369975	0,00449985
Коефіцієнт детермінації (R^2)	0,999986	0,999979
Коефіцієнт кореляції	0,999993	0,999991
Найкраща модель $f(x)$	$5,4275 - 41,3274 \cdot x_1 + 18,8829 \cdot x_1 \cdot x_5 + 0,186815 \cdot x_3/x_1 - 2,26718 \cdot x_3 \cdot x_5 + 2,41347e-05/(x_3 \cdot x_4) - 128,403 \cdot x_4 + 51,1759 \cdot x_4 \cdot x_5$	
Гістограма відхилень (%)		
Важливість змінних x (%)	(29,00; 0; 3,42; 83,41; 44,54)	
Мінімум $f(x)$ в точці: для осн. границь для розш. границь	(0,03986; -; 0,02159; 0,053409; 0,37664) (0,05; -; 0,01054248; 0,065; 0,174444)	

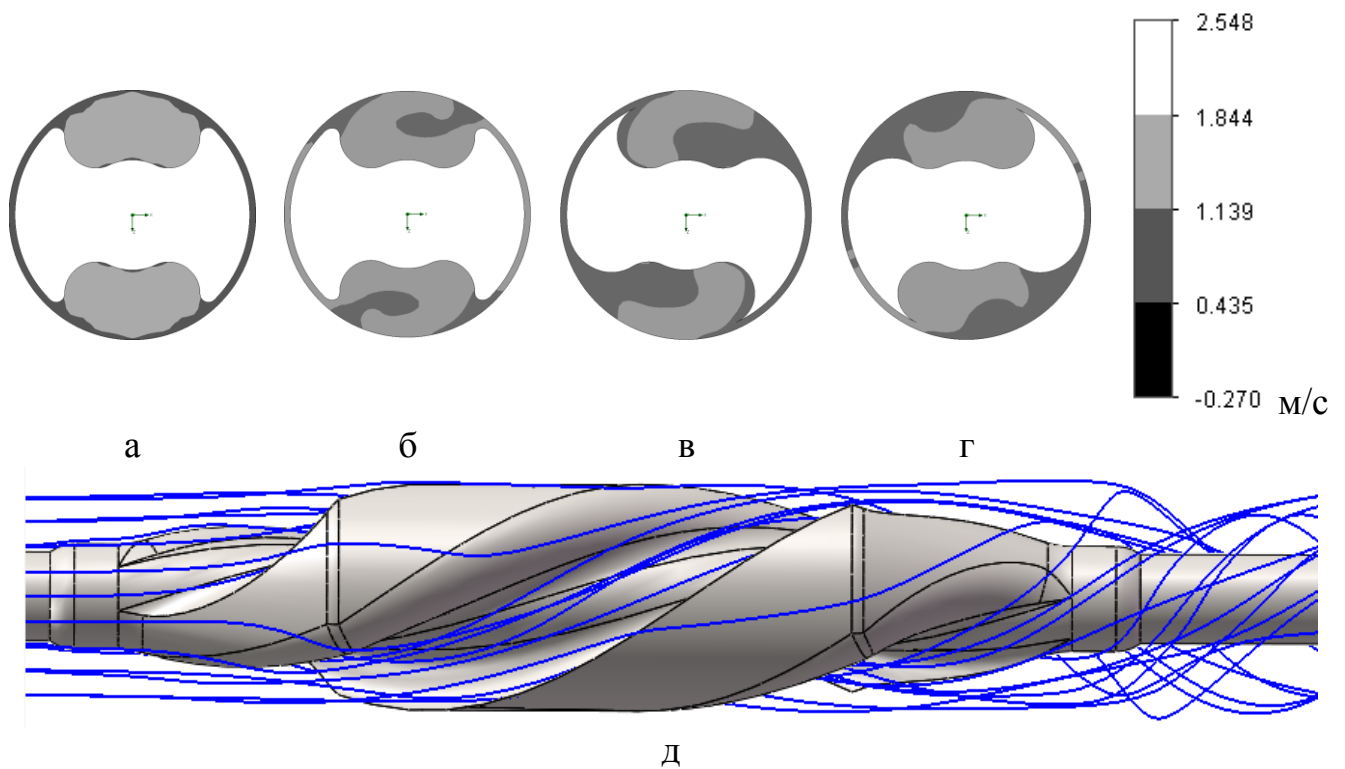


масштабовані x_i : b (—), r_1 (—), r_2 (- -), $L/2$ (- . -), a (..)

а) – модель з $\omega_1=1, \omega_2=-1$; б) – $\omega_1=1, \omega_2=0$; в) – $\omega_1=0, \omega_2=-1$

Рисунок 5.45 – Значення $f(x_i, \mathbf{x}^*)$ в околі мінімуму

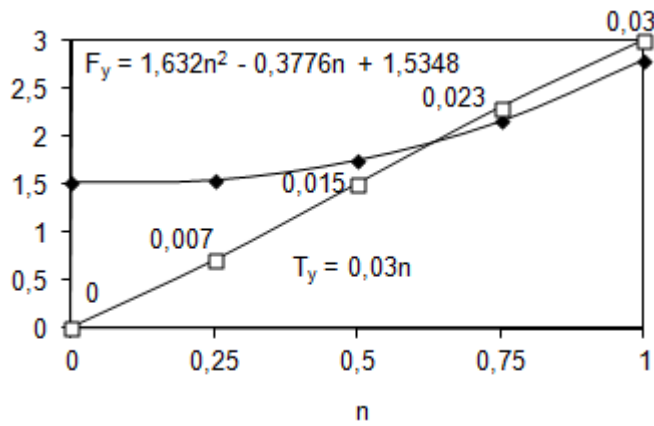
Розглянемо оптимізовану за узагальненим критерієм ($\omega_1=1, \omega_2=-1$) конструкцію протектора з гвинтовими канавками, у яких гвинтова лінія має крок $p=260$ мм та кількість поворотів n . Гвинтові канавки створюють крутний момент T_y , вектор якого збігається з віссю ШН, і змінюють напрямок швидкості рідини (рис. 5.46 д), зокрема зменшують значення швидкості рідини в напрямку осі V_y (рис. 5.46 б). Зі збільшенням n крутний момент T_y збільшується лінійно, але сила опору F_y зростає за квадратичною залежністю (рис. 5.47). Ці залежності можна використовувати для вибору значення n .



а) – прямі канавки (вигляд в напрямку вектора V_y); б-г) – гвинтові канавки;

в) – $R_{л}=10$ мм, $R_{п}=0$ мм; г, д) – $R_{л}=0$ мм, $R_{п}=10$ мм

Рисунок 5.46 – Розподіл значень швидкості рідини V_y в центральному перетині протектора (а-г) та траєкторії потоку рідини (д)



◆ – F_y (Н); □ – T_y (Нм)

Рисунок 5.47 – Залежність сили гідродинамічного опору F_y та крутного моменту T_y від кількості поворотів гвинтової лінії n

Зміна профілю канавки дозволяє збільшити T_y і дещо зменшити F_y . Для цього можна застосувати лопаті з неоднаковими радіусами скруглення ребер лопаті. Розглянемо гвинтові канавки з $n=1$. Якщо радіуси скруглення ребер однакові (2 мм), то $F_y=2,795$ Н, $T_y=0,03$ Нм. Якщо радіус скруглення лівого ребра $R_{л}=10$ мм, а правого $R_{п}=0$ мм, то $F_y=2,525$ Н, $T_y=0,026$ Нм. Якщо $R_{л}=0$ мм, а $R_{п}=10$ мм, то $F_y=2,509$ Н, $T_y=0,038$ Нм. Розподіл швидкості V_y для цих варіантів показаний на рис. 5.46 в,г.

Проведені дослідження показали, що для покращення якості протектора потрібно у першу чергу збільшувати довжину протектора L і кут канавки a . Проте замість збільшення довжини слід віддавати перевагу збільшенню кількості протекторів на штанзі з взаємним перпендикулярним розташуванням їхніх лопатей. Додатково, для забезпечення достатньої площі контакту з ШН, їх потрібно монтувати групами без зазорів вздовж ШН. Довжина протектора буде рівна довжині БС у випадку формування протектора на БС. Слід також відзначити, що збільшення кута канавки призводить до зменшення міцності лопаті. Для збільшення крутного моменту, що створює протектор з гвинтовими канавками, потрібно змінювати профіль канавки шляхом застосування неоднакових радіусів скруглення ребер лопатей. Макрос для обчислення середньої площі тертя, програма для мінімізації функції $f(\mathbf{x})$, результати симуляції

та посилання на тривимірну параметричну модель наведені в додатку У. Результати моделювання можуть бути використані для підбору оптимальних значень для інших умов експлуатації. Дана модель та методика оптимізації можуть бути використані для оптимізації параметрів протекторів інших типорозмірів.

5.5 Висновки до розділу

1. Розроблено принципи побудови САПР пресового з'єднання тіла склопластикової ШН зі сталеву головою. САПР базується на осесиметричних СЕМ з'єднання в системах Abaqus/CAE [331] і CalculiX [18] та на програмах мовою Python для автоматизації її перебудови і розрахунку. За допомогою САПР отримано розподіли контактного тиску в різних варіантах з'єднання та залежності для визначення оптимальних параметрів з'єднання. Розроблені САПР можуть бути використані для ґрунтового різностороннього аналізу з'єднань такого типу. Зокрема можна оптимізувати інші параметри з'єднання, такі як зовнішній діаметр головки, форму штампів та попередній натяг, розраховувати з'єднання на втомну міцність, удар, аналізувати міцність з'єднання для різних механічних характеристик матеріалів [16].

2. Запропонована методика побудови та побудовані залежності напруження руйнування і середнього контактного тиску в з'єднанні під час обтискання від глибини переміщення штампів для різних значень довжини обтискання головки і границі плинності сталі [331]. Аналіз таких залежностей дозволяє вибрати оптимальні параметри з'єднання – глибину переміщення штампів, границю плинності сталі, довжину обтискання.

3. Гармонічний скінченно-елементний аналіз пресового з'єднання [332] дозволив виявити його перші дві власні частоти (8039,6 Гц та 18944 Гц) та отримати амплітудно-частотні характеристики в околі цих частот. Виявлено, що осьове навантажування з'єднання другою власною частотою може спричинити вирив стержня з ніпеля або втомне руйнування стержня в верхній частині. Осьове

навантажування з'єднання першою власною частотою більш небезпечно з точки зору втомного руйнування ніпеля в нижній частині.

4. Запропонована методика обґрунтування допустимих зовнішніх гармонічних навантажень за критерієм втомної міцності ніпеля [334]. Виявлено що експлуатація з'єднання ШН діаметром 22 мм з амплітудою напружень розтягу в штанзі 1 МПа, їхнім середнім значенням 116 МПа та частотами в діапазоні 8002..8080 Гц не забезпечує циклічної довговічності 10^7 циклів сталевого ніпеля зі сталі 40.

5. Описано способи побудови для SOLIDWORKS геометричних тривимірних параметричних моделей НКТ з різноманітними окремими дефектами або їхніми комбінаціями, які за допомогою MCE дозволяють обґрунтовувати можливість їхньої експлуатації, а також досліджувати вплив розмірів того чи іншого дефекту на напруження в трубі [17]. Розроблені моделі дозволяють обґрунтовувати ефективність ремонту НКТ з різноманітними дефектами БС та оптимізувати параметри БС [5].

6. Порівнювались результати експериментальних, аналітичних і чисельних обчислень значень напружень від внутрішнього тиску в трубах з БС, товщина яких близька до товщини стінки труби [3]. Результати показують, що такий БС зменшує кільцеві напруження в 1,18-1,54 рази, а осьові – в 1,05-1,21 рази. Розроблена параметрична модель НКТ з БС і корозійним дефектом правильної форми, який являє собою поверхню обертання, що отримується обертанням еліпса навколо осі, перпендикулярній осі труби [5]. Перевагами такого способу побудови є можливість отримання великої кількості різних форм внутрішніх і зовнішніх дефектів, які описуються невеликою кількістю параметрів. На основі моделі отримано залежності максимального еквівалентного напруження в НКТ від товщини БС. Збільшення товщини БС більш ефективно для ремонту глибоких внутрішніх дефектів, а ремонт зовнішніх дефектів вимагає попереднього вирівнювання дефекту заливним ПКМ.

7. Виявлено, що нанесення БС без фаски на тіло ШН зменшує коефіцієнт запасу D з 2,43 до 2,05 для циклу втомного навантаження $\sigma_p=0...150$ МПа. На

основі аналізу різних варіантів конструкції фаски БС розроблено параметричну геометричну модель і СЕМ ШН (НКТ) з БС і з еліптичною фаскою. Виявлено, що еліптична фаска БС ШН з $a=20$ мм та радіусами скруглення і додатковим шаром епоксидної смоли є оптимальним варіантом конструкції та забезпечує мінімальне значення $D=2,41$. Для НКТ діаметром 73 мм та з товщиною стінки 5,5 мм і з БС товщиною 5,5 мм оптимальною фаскою є еліптична фаска з $a=32$ мм.

8. Запропонована методика скінченно-елементного аналізу труб з зовнішньою осьовою тріщиною і склопластиковим бандажем [4], яка може бути використана для обґрунтування зміцнення БС НКТ з іншими типами тріщин. Розроблено параметричну модель для визначення КІН (прямим методом) в ремонтваній БС НКТ діаметром 73 мм та товщиною стінки 5,5 мм з осьовою тріщиною на зовнішній поверхні з круговим фронтом, якщо на НКТ діє циклічний внутрішній тиск. Виявлено, що нанесення БС товщиною 5,5 мм на НКТ з тріщиною 0,5 мм може підвищити її циклічну довговічність з 1 до 10 млн. циклів для циклічного віднульового внутрішнього тиску 27 МПа (що відповідає циклічним напруженням 0-150 МПа для даної НКТ). Отримано залежність КІН в БС від його товщини і виявлено, що товщина більша 4-5 мм майже не зменшує значення КІН і може бути прийнята за оптимальну.

9. За допомогою SOLIDWORKS розроблені параметричні моделі тіла ШН з БС, яка має сферичний або циліндричний корозійний дефект або втомну тріщину з прямолінійним фронтом. Моделі можуть бути використані для оптимізації конструкції бандажа МСЕ. Виявлені регресійні залежності еквівалентних напружень в штанзі діаметром 22 мм з БС від радіуса і глибини корозійного дефекту. Залежності підтверджують зміцнюючий ефект БС. Виявлені регресійні залежності еквівалентних напружень в штанзі з сферичним дефектом ($R=20$ мм, $h=6$ мм) від довжини і діаметра БС. Виявлені регресійні залежності КІН від глибини втомної тріщини з прямолінійним фронтом і побудовані криві втоми ШН з БС і без БС, якщо ШН має початкову тріщину глибиною 1 мм. Показана їхня відповідність результатам відомих втомних випробувань. Моделі, методика аналізу і залежності можуть бути використані для обґрунтування доцільності

ремонту ШН з дефектами БС і оптимізації їхньої конструкції. Обчислена за моделями довговічність ШН з тріщиною глибиною 1 мм в середовищі %3 NaCl збільшується в 6,3-79 раз після ремонту БС.

10. В умовах додаткових навантажень згину $\sigma_{зг}$ збільшення радіусу заокруглення r між піделеваторним буртом і тілом ШН діаметром 22 мм до 500 мм призводить до збільшення коефіцієнта запасу втомної міцності D за критерієм Брауна-Міллера орієнтовно в 1,2 рази [351]. Отримано регресійну залежність $D=f(r, \sigma_{зг})$. Розроблені СЕМ та макрос можуть бути використані для аналізу та оптимізації ШН іншого типорозміру чи конструкції для будь-яких значень навантажень або властивостей матеріалів.

11. Розроблено параметричну гідродинамічну СЕМ протектора для ШН та запропоновано методику оптимізації його параметрів за критеріями мінімального гідродинамічного опору та максимальної площі тертя. Методика основана на ітераційному алгоритмі, який містить етапи: побудова плану експерименту, побудова регресійної моделі за узагальненим критерієм, пошук мінімуму. Запропоновані залежності для вибору значень параметрів в залежності від важливості кожного критерію. Для протектора з гвинтовими лопатями отримані залежності сили гідродинамічного опору та крутного моменту від кількості поворотів гвинтової лінії. Для збільшення крутного моменту слід змінювати профіль канавки шляхом застосування неоднакових радіусів скруглення ребер лопатей. Результати можуть бути використані для пошуку оптимальних значень параметрів для інших типорозмірів та умов експлуатації.

РОЗДІЛ 6

ПРИНЦИПИ ПОБУДОВИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПРОЕКТУВАННЯ І ПІДТРИМКИ ЖИТТЄВОГО ЦИКЛУ ОБЛАДНАННЯ ШСНУ

6.1 Абстрактна модель інформаційної системи

6.1.1 Модель інформаційної системи та її відповідність загальносистемним закономірностям

Описані в попередніх розділах моделі, прикладні САПР, результати симуляції, факти та інші інформаційні ресурси повинні бути об'єднані в єдину ІС з ціллю ефективного вирішення проблем проектування обладнання комплексно-працевдатної ШСНУ, а також підтримки інших етапів ЖЦ ШСНУ, на системному рівні. В цьому розділі розроблені принципи побудови такої ІС.

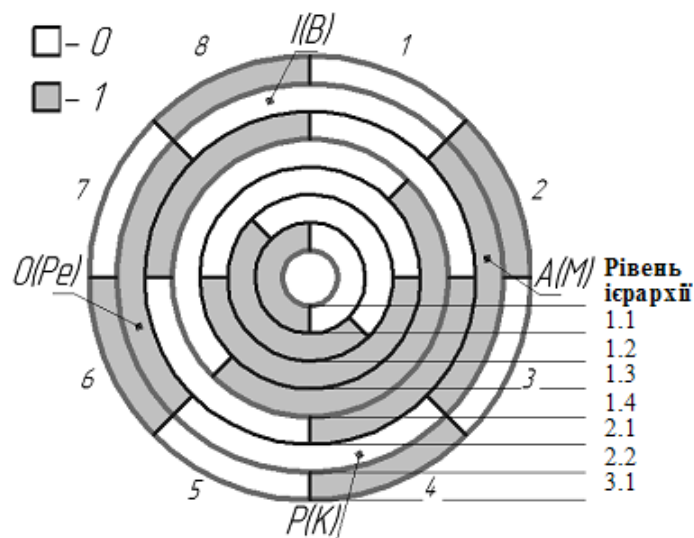
Міждисциплінарний аналіз загальносистемних закономірностей [213] дозволяє розробити абстрактну модель ІС. *Абстрактна модель ІС* відображає тільки найбільш загальні її характеристики, шляхом дихотомічного ділення ЖЦ виробу виділяє класи функціональних елементів ІС та їхню ієрархію, а також володіє основними закономірностями складних систем [213]. Кожний функціональний елемент ІС можна віднести до певного класу. *Клас елемента ІС* описує множину однотипних спеціалізованих елементів ІС, які призначені для досягнення певних цілей ІС та можуть взаємодіяти з іншими елементами для досягнення заданого рівня синергії. В основу класифікації елементів ІС покладемо ізоморфізм структур ЖЦ виробу і ІС, який кожному класу ЖЦ виробу ставить у відповідність один і тільки один клас ІС. Виконаємо класифікацію ЖЦ виробу методом дихотомії. Дихотомічне ділення понять доволі просте і ефективне – два класи завжди виключають одне одного. Ігнорування дихотомією під час

класифікації може призвести до виділення тільки деяких найбільш очевидних елементів, тоді як менш очевидні ігноруються. Кожний клас володіє унікальною *множиною ознак* – множиною індексів класів на нижчому рівні ієрархії. Якщо розрізняти n рівнів ієрархії, то на нижньому її рівні отримаємо 2^n класів. Відповідно до закону формальної логіки про обернене співвідношення змісту і обсягу поняття підкласи будуть мати менший обсяг (кількість ознак) і більший зміст у порівнянні з надкласами. *Біполярними множинами ознак* назвемо дві множини ознак, які ділять множину усіх ознак на дві рівні частини. Поділимо класи ЖЦ на *етапи ЖЦ* і *потоки ЖЦ*. Відповідні класи ІС будемо називати *етапи ІС* та *потоки ІС*. Етап ЖЦ існує в часі послідовно з іншими етапами ЖЦ. Потік ЖЦ існує в часі паралельно з іншими потоками. Для прикладу розглянемо модель з $n=3$ (рис. 6.1), у якої останній рівень 3.1 містить 8 етапів. Розділимо весь ЖЦ на два нерозривні в часі етапи користуючись біполярними множинами ознак $\{1, 2, 3, 4\}$, $\{5, 6, 7, 8\}$ і отримаємо рівень 1.1. Не важко переконатися, що тут існує чотири способи поділу ЖЦ на два нерозривні в часі етапи (рівні 1.1, 1.2, 1.3, 1.4). Знову розділимо кожний з цих етапів на два і отримаємо рівні 2.1 і 2.2. Таким чином, перший рівень можна класифікувати чотирма різними способами (1.1, 1.2, 1.3, 1.4), другий – двома (2.1, 2.2), третій – одним (3.1). Існує рекомендація не дуже подрібнювати ЖЦ на етапи [259]. Кількість етапів повинна бути оптимальною з точки зору ефективності управління ЖЦ. Зі збільшенням кількості етапів зростає проблема їхньої інтерпретації. Можна обмежитись розглядом восьми етапів ЖЦ з поділом кожного етапу на два паралельні потоки – **а** та **б** [213]. Внаслідок наявності двох потоків етапи 1-8 володіють множинами ознак з двох елементів, наприклад етап 1 володіє множиною ознак $\{1a, 1б\}$. Таку модель назвемо 8.2, де 8 – кількість етапів, 2 – кількість потоків.

В табл. 6.1 та табл. Ф.1 наведені орієнтовні змістові інтерпретації деяких класів ІС, які були отримані шляхом аналізу закономірностей складних систем,

прикладів ЖЦ різних природних та штучних систем та методик системного аналізу [213]. Зокрема були використані дослідження в галузі психологічних типологій [231-233, 353]. Детальніша інтерпретація повинна бути предметом подальших таких міждисциплінарних досліджень. Зауважимо, що існує 6435 способів поділу 16 елементів на дві частини та 35 способів такого поділу 8 елементів. Тому існують і інші біполярні множини ознак, наприклад $\{1, 4, 6, 7\}, \{2, 3, 5, 8\}$ або $\{1, 3, 6, 8\}, \{2, 4, 5, 7\}$, але їхній зміст є предметом подальших досліджень.

Розглянемо рівень ієрархії 2.2, який поділяє ЖЦ на 4 етапи з поділом кожного етапу на два потоки. Тут існує вісім класів елементів ІС. Перелічимо їх за множинами ознак: $\{8a, 1a\}, \{8b, 1b\}, \{2a, 3a\}, \{2b, 3b\}, \{4a, 5a\}, \{4b, 5b\}, \{6a, 7a\}, \{6b, 7b\}$. Для зручності надамо їм імена: *I* (ідея), *B* (вимога), *A* (аналіз), *M* (мотив), *P* (рішення), *K* (контроль), *O* (оцінка), *Pe* (реалізація). Опис цих класів з прикладами елементів ІС подано в табл. 6.1. Опис деяких інших класів на інших рівнях ієрархії подано в табл. Ф.1.



0(□),1(■) – біполярні множини ознак

Рисунок 6.1 – Ієрархія етапів для моделі 8.2

Таблиця 6.1 – Опис класів елементів ІС на рівні ієрархії 2.2

Клас, ознаки	Назва класу, приклади елементів (методи, засоби, моделі) та їхньої інформаційної продукції
<i>I</i> {8a,1a}	<i>Ідея.</i> Генерація ідей відповідно вимог. Формулювання гіпотез, формулювання концепції, інформаційний пошук, пошук знань, виділення перспектив, розробка варіантів і моделей прийняття рішення, шляхи модернізації продукції, концептуальні моделі, МАІС.
<i>B</i> {8б,1б}	<i>Вимога.</i> Формулювання вимог відповідно перспектив. Технічне завдання, технічна діагностика, збір даних про експлуатацію, спостереження, споживання, визначення актуальності проблем, виявлення проблеми, маркетингові дослідження.
<i>A</i> {2a,3a}	<i>Аналіз.</i> Аналіз ідей відповідно мотивів. Побудова математичних моделей, формально-логічні методи, узагальнення, класифікація, СА, інтелектуальний аналіз даних, технічна концепція, МФПС.
<i>M</i> {2б,3б}	<i>Мотив.</i> Вироблення мотивів відповідно аналізу. Постановка цілей на об'єктах, які відповідають вимогам. Співставлення вимог з об'єктами, обґрунтування прийняття рішення, методи експертного оцінювання альтернатив і пошуку рішення.
<i>P</i> {4a,5a}	<i>Рішення.</i> Прийняття рішень відповідно прогнозів. Впровадження у виробництво, виділення ресурсів для виробництва, залучення інвестицій, технологічна підготовка виробництва, організація виробництва, освоєння, виготовлення дослідного зразка, випробовування, експериментальні методи.
<i>K</i> {4б,5б}	<i>Контроль.</i> Контроль прийняття рішень. Критичний огляд проаналізованих ідей, методи прогнозування, розрахунок надійності, оптимізація параметрів, планування виробництва, управління часом, контроль якості, експертиза і коригування технічної документації, модифікація продукції, доопрацювання дослідного зразка.
<i>O</i> {6a,7a}	<i>Оцінка.</i> Оцінка рішень відповідно реалізацій. Порівняння з традиційними рішеннями, використання уніфікованих та стандартизованих технологій, стабілізація виробничих процесів, уніфікація продукції, сертифікація продукції.
<i>Pe</i> {6б,7б}	<i>Реалізація.</i> Реалізація рішень відповідно оцінок. Накопичення фактів, міркування на основі прецедентів, методи синтезу, серійне і освоєне виробництво, програми оптимізації виробництва і експлуатації, управління витратами, постачання і реалізація продукції, технічне обслуговування і ремонт.

Проаналізуємо наскільки повно ця модель відповідає основним загальносистемним закономірностям [213].

Закономірність емерджентності. ІС повинна містити функціональні елементи, взаємодія яких повинна давати позитивний ефект, якого немає у цих елементів. Цей ефект є цілком такою взаємодією. Кожний клас ІС має свої цілі, тому у складних ІС важко визначити цілі. Цілі елементів можуть мати протилежну направленість, що призводить до конфліктних ситуацій під час спроби їх неправильно поєднати. Наприклад, на етапах рівня ієрархії 3.1 з ознаками 0 цілі ставляться більш загальні, ніж на етапах з ознаками 1. Їхня паралельна робота неможлива. Але елементи різних класів мають спільні цілі, якщо на вищих рівнях ієрархії (рис. 6.1) вони утворюють цілісність – мають спільні етапи. Аналіз більшості описаних стадій творчого мислення, методик прийняття рішення та системного аналізу [213] дозволяє виділити в них перший етап генерації ідей (етап пошуку, інсайт, "абстрактне рішення", розробка концепції, МАІС) і наступний за ним етап аналізу цих ідей (етап аналізу, перевірка, "конкретне рішення", технічна розробка, МФПС). Для простоти назвемо класи елементів ІС, які використовуються на цих етапах, "ідеї" (*I*) та "аналіз" (*A*). Їхня послідовна взаємодія дозволяє досягти спільної цілі – продукувати глибоко проаналізовані ідеї. Елементи *I* на виході генерують у великій кількості ідеї, які надходять на вхід елементів *A*. На виході елементи *A* видають тільки обґрунтовані ідеї.

Закономірність інтегративності. Відповідно до діалектичного закону єдності та боротьби протилежностей елементи ІС повинні бути протилежними. Наприклад, елементи *I* можна вважати ірраціональними так як вони переважно містять методи, точність яких раціонально не доведена (наприклад евристичні алгоритми), але які часто дають прийнятний результат. Елементи *A* містять гарантовано точні та оптимальні методи (наприклад методи математичної логіки), тому є раціональними. У системі ці елементи працюють послідовно, тобто розділені у часі та є протилежними відносно певного моменту часу. З цих точок зору елементи *I* та *A* можна вважати протилежними.

Якщо елементи I та A працюють послідовно, то відповідно закономірності інтегративності повинна існувати протилежність до них, яка працює з ними паралельно. Під паралельною роботою елементів слід розуміти їхній обмін інформацією з набагато вищою швидкістю, ніж під час послідовної роботи. Елементи, які працюють послідовно, розділені в часі. Елементи, які працюють паралельно, розділені в просторі. Паралельна протилежність елемента I повинна його синхронно (або майже синхронно) доповнювати до цілісності. Аналіз прикладів ЖЦ показує, що на першому етапі поряд з генерацією ідей нерідко існує етап висування вимог (проблемна ситуація, інкубація, орієнтування, формулювання потреб, вимірювання, спостереження, виявлення проблеми, функціональна недостатність). Назвемо його просто "вимоги" (B). Може скластись враження, що спочатку повинні висуватись вимоги, а потім генеруватись ідеї. Однак висування великої кількості вимог не можливе без їхнього одночасного обстеження на перспективність. Недосяжні вимоги повинні відразу відкидатись елементами I . Інакше кажучи, елементи I, B майже синхронно продукують ті ідеї, які відповідають вимогам, і висувають ті вимоги, які перспективні. Це помітно на першому етапі ЖЦ системи "розробка концепції" в системній інженерії [253], який на вході одночасно отримує інформацію "функціональна недостатність" (вимоги) та "технічні можливості" (перспективи). В методиках системного аналізу їм відповідає етап виявлення проблеми і постановки цілей.

Аналогічно елемент A повинен мати паралельну протилежність, яка його синхронно доповнює до цілісності, підтримує, запобігає надлишковому і непродуктивному аналізу. Назвемо її "мотив" (M). В психології мотивація визначає спрямованість, організованість, активність і стійкість поведінки людини. Мотив конкретно вказує на ресурс, за допомогою якого забезпечуються вимоги. Елементи A паралельно допомагають у виборі цього ресурсу шляхом аналізу перспектив. Елементи A, M використовують більш раціональні методи ніж I, B .

Закономірність ієрархічності. Відповідно до закономірності ієрархічності елементи I, B, A, M утворюють цілісність, яка є частиною ІС і виконує функції

проектування виробу або управління реалізацією. Очевидно, що вона не включає функціональні елементи для реалізації та супроводження системи. Теорія управління стверджує, що етап управління повинен бути послідовно доповнений етапом виконання. Більшість ЖЦ прийняття рішення містять тільки етап проектування. Центральним в них є етап аналізу (A). Але коли іде мова про розробку ІС підтримки ЖЦ продукту, повинні розглядатись також етапи реалізації. Етап реалізації теж повинен складатись з послідовних етапів – ірраціонального і раціонального. Як правило етап реалізації починається з прийняття рішення (P) і ним же і закінчується етап проектування. Синхронно з прийняттям рішення виконується прогнозування та критичний огляд результатів проектування (K). Закінчується етап реалізації власне реалізацією (Pe) з синхронною оцінкою результатів (O). Таким чином, існує чергування ірраціонального (образного) і раціонального (формального) етапів в ЖЦ (табл. 6.1). На такому чергуванні основана методика поступової формалізації моделей прийняття рішень [259].

Відповідно до закономірності цілісності множина елементів $\{I, B, A, M\}$ повинна бути доповнена множиною з чотирма елементами, які забезпечують реалізацію і супроводження виробу – $\{P, K, O, Pe\}$. Множину елементів I, A, P, O назовемо "*потік а*", а множину елементів K, Pe, B, M – "*потік б*". Відомою є практика приблизно в середині періоду проектування (або ЖЦ) системи починати проектування системи наступної черги [354]. Тут системі першої черги відповідає "*потік а*" (I, A, P, O, I), а системі другої черги – "*потік б*" (K, Pe, B, M, K), який починається з етапу K . Елементи K , на відміну від I , не генерують щось принципово нове, а критично оглядають попередні досягнення. Таким чином, "*потік б*" доповнює "*потік а*".

Ці вісім елементів утворюють ієрархічні структури в часі та просторі. Прикладом ієрархічної структури в часі є послідовність двох підсистем (I, A), (P, O). Прикладом ієрархічної структури в просторі та часі є множина $\{(I, A), (B, M)\}$. Помітно, що протилежні класи на одному рівні мають спільні риси з протилежними класами на іншому рівні. Так на кожному рівні виконується

перехід від загального етапу до конкретного. Разом з тим, класи на різних рівнях відрізняються. Тобто помітний ізоморфізм і поліморфізм класів та фрактальна структура ієрархії.

Щоб показати властивість фрактальності розглянемо відповідність цих класів стадіям розроблення конструкторської документації, які описані в стандартах ГОСТ 2.103-2013 та ДСТУ 3974-2000. Ці стадії відповідають етапу проектування (класи I, B, A, M) в ЖЦ виробу:

$\{I, B\}$ – Технічна пропозиція. Ескізний проект. (B) – Технічне завдання. Технічні вимоги (вимоги, норми і характеристики, які визначають показники якості та експлуатаційні характеристики виробу). Області застосування і умови експлуатації. Обґрунтування доцільності розроблення. (I) – Вивчення і аналіз технічного завдання. Аналіз відомих рішень. Виявлення варіантів можливих рішень, установлення їхніх особливостей і конструкторське пророблення. Передпроектне теоретичне дослідження. Загальні уявлення про принцип роботи і будову виробу. Вибір варіантів виробу. Характеристики варіантів. Конструкторське пророблення варіантів (достатнє для їхнього співставлення). Орієнтовні розрахунки.

$\{A, M\}$ – Технічний проект. (M) – Виявлення кінцевих технічних рішень. Якісна оцінка відповідності вимогам. Визначення технічного рівня продукції. (A) – Уточнені розрахунки, що підтверджують техніко-економічні показники та показники надійності виробу (кількісні). Функціональні та принципові схеми. Аналіз технологічності в умовах конкретного виробництва.

$\{P, K\}$ – Робоча конструкторська документація дослідного зразка. Виготовлення дослідного зразка. (P) – Розроблення конструкторської документації дослідного зразка виробу. Технологічна підготовка виробництва дослідного зразка. Розроблення засобів технологічного оснащення і контрольно-вимірювальної апаратури. Виготовлення і попередні випробування дослідного зразка. (K) – Коригування конструкторської документації за результатами виготовлення і попередніх випробувань. Повне комплектування розробленої

документації. Нормоконтроль. Проект технічних умов. Розроблення програм випробувань.

$\{O, Pe\}$ – Робоча конструкторська документація дослідного зразка. Приймальні випробування дослідного зразка. (Pe) – Виготовлення дослідного зразка. Приймальні випробування дослідного зразка. (O) – Оцінка результатів випробувань. Коригування конструкторської документації за результатами випробувань. Розроблення конструкторської документації на виріб серійного виробництва. Сертифікація продукції.

Крім того, відповідно ГОСТ 2.119-73, етап ескізного проекту додатково може містити підетапи подібні за ознаками на описані вище: уточнення вимог (В), створення та аналіз моделей (А), відносна оцінка варіантів (М), вибір оптимального варіанту (К), оцінка технологічності (К), прийняття принципових рішень (Р), виготовлення (Р) і випробування (Pe) дослідного зразка, оцінка результатів випробування (О), оцінка показників стандартизації та уніфікації (О). Це ж стосується інших етапів проектування. Помітно, що на усіх рівнях ієрархії кожний наступний етап подібний на попередній, але характеризується більш конкретними рішеннями (діалектичний принцип «від абстрактного до конкретного»).

Застосовані в розділі 4.8 цикли проектування РЗ (ітерації) містять елементи класів I (гіпотези) і B (вимоги) на етапі 1, A (аналіз гіпотез) і M (аналіз вимог) – на етапі 2, P (симуляція) і K (оптимізація) – на етапі 3, O (оцінка) і Pe (запис результатів) – на етапі 4. Крім того, етап 3 поділявся на субітерації, призначені для оптимізації параметрів і подібні за структурою на ці ітерації.

Отже модель має фрактальну структуру і тому може описувати не тільки ІС підтримки всього ЖЦ виробу, але й будь-якого його етапу (проектування, виробництва, експлуатації) і підетапу. Розроблена модель є ізоморфною до структур ціленаправленої діяльності в різних складних ІС, що показано шляхом міждисциплінарного аналізу [213].

Закономірності розвитку. У діалектичній логіці під логічною категорією розуміється поняття, що відображає послідовну стадію становлення будь-якого

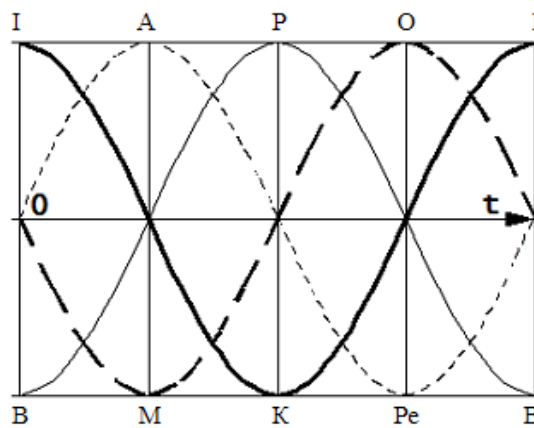
конкретного цілого. Такими категоріями є класи ІС, а послідовна робота елементів класів $\{I, B\}$, $\{A, M\}$, $\{P, K\}$, $\{O, Pe\}$ утворює робочий цикл ІС. Цій абстрактній моделі ІС є ізоморфною динамічна математична модель ІС "гармонічний осцилятор" (6.1) з початковими умовами $x(0)=1$, $y(0)=0$. Графік функцій $x=x_2$, $y=y_2$ показано на рис. 6.2. Кількісно охарактеризувати рівень елемента I можна змінною x , а рівень елемента Pe (або O) – змінною y . Нехай x – це кількість ідей, а y – кількість реалізацій. Тоді кількість реалізацій y пропорційна швидкості збільшення кількості ідей x , а кількість ідей x пропорційна швидкості зменшення кількості реалізацій y . Від'ємні значення x (або y) слід інтерпретувати як її протилежність. Наприклад, $x>0$ відповідає ідеї, $x<0$ – рішенню, $y>0$ – реалізації, $y<0$ – аналізу. Коливання осцилятора спричинені наявністю в системі протилежностей x і y . Спочатку кількість ідей найбільша, а реалізацій немає ($x(0)=1$, $y(0)=0$). З часом кількість ідей зменшується і зростає рівень проаналізованих ідей. Завершення етапу аналізу ($y(\pi)=0$) збігається з моментом прийняття найбільшої кількості рішень ($x(\pi)=-1$). Після цього починається етап реалізацій, завершення якого збігається з завершенням циклу ($t=2\pi$). В середині етапу реалізації починає зростати кількість ідей, наприклад внаслідок проблем експлуатації виробу. Ця модель може використовуватись для опису ЖЦ на обмежених проміжках часу (в межах періоду).

$$\begin{cases} dx/dt = y, \\ dy/dt = -x. \end{cases} \quad (6.1) \quad \begin{cases} dx/dt = 0,1x + y, \\ dy/dt = 0,1y - x. \end{cases} \quad (6.2) \quad \begin{cases} dy_1/dt = x_2, \\ dx_2/dt = y_2, \\ dy_2/dt = x_1, \\ dx_1/dt = y_1. \end{cases} \quad (6.3)$$

Закономірність історичності вказує на те, що завершення одного ЖЦ продовжується наступним з виходом функціональних елементів на новий рівень розвитку. Для цього архітектура ІС повинна мати можливості розширення: нарощування кількості функцій, компонентів, даних та знань. Найпростіше таку ІС можна змодельовати системою рівнянь (6.2), фазова траєкторія якої являє собою спіраль. Тут загальний рівень розвитку може бути визначений як радіус-

вектор спіралі $R = \sqrt{x^2 + y^2}$. Якщо величина R не суттєво зростає на сусідніх циклах, то можна перейти до простішої моделі (6.1). В моделі системи якості «спіраль якості» Джурана якість виробу підвищується на кожному витку спіралі [355]. В циклах проектування двоопорного РЗ (розділ 4.8), фаски БС (розділ 5.2.3), протектора для ШН (розділ 5.4) якість виробу покращується на кожній ітерації.

Врахувати елементи B, M, K, O можна в моделі гармонічного осцилятора з чотирма змінними (6.3) з початковими умовами $x_1(0) = -1, x_2(0) = 1, y_1(0) = 0, y_2(0) = 0$. Тут позитивні значення змінних x_1, y_1, x_2, y_2 відповідають елементам P, A, I, O , а негативні – елементам B, Pe, K, M (рис. 6.2). За потреби цю модель можна легко модифікувати в осцилятор з прогресуючими коливаннями.



(—) – x_1 ($max=P, min=B$); (- - -) – y_1 ($max=A, min=Pe$);

(- · - ·) – x_2 ($max=I, min=K$); (· · · ·) – y_2 ($max=O, min=M$)

Рисунок 6.2 – Динамічна модель ІС за рівнянням (3)

Закономірність самоорганізації в ІС можна спостерігати, якщо перейти до дещо складнішої динамічної моделі, наприклад рівняння Релея. Майже незалежно від початкових умов модель показує часову самоорганізацію – автоколивання. В цій же моделі проявляється і *закономірність еквіфінальності* – фазова крива прямує до граничного циклу, що означає наявність границі розвитку ІС ($R \rightarrow const$). Для подолання границі ІС повинна об'єднуватись з протилежною до неї відповідно закономірності емерджентності. Для такого об'єднання ІС повинна володіти множиною входів і виходів, як це вимагає *закономірність комунікативності*. Функціональна недостатність ІС може також призводити до її

розвитку шляхом дихотомічного ділення її з часом t на дві протилежні спеціалізовані підсистеми за законом 2^t .

Закономірності здійсненності систем. Відповідно до закономірності "необхідної різноманітності" "різноманітність" ІС повинна бути більшою "різноманітності" системи, якою вона управляє – технічної системи. Наприклад, різноманітність забезпечують методи "мозкової атаки" або колективної генерації ідей та інші методи вироблення колективних рішень. Різноманітність також може бути забезпечена утворенням конструкцій з елементів за допомогою різних комбінаторних конфігурацій (перестановок, розміщень, комбінацій, розбиття), як це робиться в морфологічному підході. Перший закон перетворення композицій системи [356] стверджує, що в системі можуть змінюватись: елементи (E), кількість елементів (K), відношення між елементами (R), та поєднання KR, RE, KE, KRE. Таким чином, є 7 способів змінити систему. Для прикладу, кількість різнотипних елементів (K) можна збільшити шляхом збільшення рівнів ієрархії (рис. 6.1). Розроблені автором елементи ІС володіють високим рівнем "різноманітності" та забезпечують різносторонній погляд на проблему ефективного проектування обладнання та забезпечення працездатності ШСНУ.

Множина класів ІС повинна забезпечити цілісний і різноманітний погляд на ЖЦ виробу. Такі множини можна отримувати шляхом дихотомічного ділення системи за різними її ознаками так, щоб вони були ізоморфні до абстрактної моделі ІС. Зокрема в динамічних системах можна виділити три незалежні біполярні ознаки: *елемент-відношення, стан-перехід, процес-«зміна процесу»* [213]. Фрактальна структура ієрархії відповідних класів проявляється в тому, що класи *елемент, стан, процес* розглядають систему з певної "статичної" на всіх рівнях ієрархії точки зору, а класи *відношення, перехід, зміна процесу* – з "динамічної" [213].

Різноманітність також забезпечується множиною відношень між класами ІС. Функціональні елементи ІС взаємодіють шляхом обміну інформацією. Для дослідження таких взаємодій виконаємо аналіз *бінарних відношень* між функціональними елементами ІС різних класів. Множину бінарних відношень Y

можна отримати шляхом прямого (декартового) добутку множини класів X з собою $Y=X \times X$. Для прикладу розглянемо відношення між етапами ІС за ознаками 2.1 (класи – об'єкти), або відношення між етапами ІС за ознаками 2.2 (ірраціональність – раціональність). Множина Y бінарних відношень 4 етапів буде містити 16 відношень (впорядкованих пар), з яких 4 рефлексивні, наприклад $(\{1,2\}, \{1,2\})$. Можна виділити 4 типи відношень (табл. Ф.2):

1. *Тотожність*. Синхронність, цілісність в просторі.

2. *Пряма послідовність*. Прямий зв'язок. Утворює цілісність в часі, синергію.

3. *Антипослідовність*. Розділеність в часі, асинхронність. Синхронна або послідовна взаємодія елементів цього відношення характеризуються певною диссинергією, конфліктністю, призводить до зниження ефективності функціонування системи та зростання адитивності елементів.

4. *Обернена послідовність*. Зворотний зв'язок. Утворює з прямою послідовністю цілісність. У зв'язку з однонаправленістю осі часу відношення "пряма послідовність" має вищий рівень синергії у порівнянні з відношенням "обернена послідовність".

Множина бінарних відношень восьми класів (I, B, \dots, Pe) буде містити 64 відношення, з яких 8 рефлексивні. Можна виділити 8 типів відношень за ознакою часової віддаленості етапів ІС (табл. Ф.3). Цифрою 1 позначені типи відношень між елементами з однаковою ознакою "потік a "-"потік b ", а цифрою 2 – з різними ознаками. Аналіз відношень дозволить підвищити ефективність інформаційних взаємодій функціональних елементів ІС, збільшити синергію і зменшити диссинергію. Наприклад, класи ідея-вимога $\{I, B\}$ не можуть функціонувати паралельно з рішенням-критика $\{P, K\}$, переважання в ІС відношення (I, P) призводить до ефекту "необдумані рішення", а (I, K) – до ефекту "надлишкова критика ідей", відношення (A, A) – до ефекту "надлишковий аналіз", (O, O) – до ефекту "інерція мислення" і т.д. Методи МАІС, які основані на елементі (I) , будуть працювати найефективніше під час синхронності $\{I, B\}$, ефективно під час прямої послідовності (I, A) , (I, M) , (O, I) , (Pe, I) , і не ефективно під час антипослідовності $\{I, P\}$, $\{I, K\}$ та оберненої послідовності (I, O) , (I, Pe) , (A, I) ,

(M, I). Наприклад, метод "мозкової атаки" [222], який відноситься до МАІС, найефективніший тоді, коли у великій кількості висуваються будь-які, навіть абсурдні, ідеї, на етапі генерації повністю відсутня критика $\{I, K\}$, інерція мислення, обмеження та оцінювання ідей (I, O), генератори ідей отримують позитивну підтримку, комфорт і заохочення ($\{I, B\}, (I, M)$), після етапу генерації відбувається аналіз усіх висунутих ідей (I, A). Засобами підвищення ефективності методу експертного оцінювання "Дельфі" [222], який також відноситься до МАІС, є: подолання конформізму $\{K, I\}, \{P, I\}$ і підвищення мотивації експертів (I, M), організація зворотного зв'язку експертів з результатами, удосконалення методів аналізу (I, A).

Синергія елементів ІС полягає у тому, що результат взаємодії двох або більше елементів різних класів є ефективнішим за їхній сумарний результат без взаємодії. *Рівень синергії* підсистеми ІС можна розглядати як ступінь її цілісності та орієнтовно визначати як значення функцій $SR(x)$, яке залежить від множини елементів підсистеми x ІС з заданою ієрархічною схемою (рис. 6.1). В залежності від виду ІС можна запропонувати різні такі функції та схеми. Наприклад, розглянемо систему з 4 класами. Булеаном множини $\{1, 2, 3, 4\}$ є множина з $2^4=16$ елементами: $\{\{\}, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \{1, 2, 3\}, \{1, 2, 4\}, \{2, 3, 4\}, \{1, 3, 4\}, \{1, 2, 3, 4\}\}$. В цьому випадку рівень синергії найпростіше визначити як кількість елементів підсистеми x мінус 1: $SR(1, 2) = SR(1, 3) = SR(1, 4) = SR(2, 3) = SR(2, 4) = SR(3, 4) = 1$, $SR(1, 2, 3) = SR(1, 2, 4) = SR(2, 3, 4) = SR(1, 3, 4) = 2$, $SR(1, 2, 3, 4) = 3$. За схемою з 8 класами (рис. 6.1) булеан множини буде містити 256 елементів. В цьому випадку значення функції SR можна визначити як підсумовану для усіх можливих пар елементів підсистеми кількість кільцевих секторів круга спільних для пари елементів. Наприклад: $SR(1)=0$; $SR(1, 5)=1$; $SR(1, 4)=2$; $SR(2, 3)=5$; $SR(1, 2)=5$, $SR(1, 2, 5) = SR(1, 2)+SR(2, 5)+SR(1, 5) = 5+2+1=8$, $SR(1, 2, 3, 4) = SR(1, 2)+SR(2, 3)+SR(3, 4)+SR(1, 4)+SR(1, 3)+SR(2, 4) = 5+5+5+2+3+3 = 23$. У випадку застосування різних схем функцію SR бажано нормувати так щоб $0 \leq SR_{норм}(x) \leq 1$. Наприклад, $SR_{норм}(x) = SR_{max} / SR(x)$, де SR_{max} – максимальний рівень синергії для даної схеми. Низький рівень синергії слід називати диссинергією.

Закономірність адитивності. Елементи системи повинні бути незалежними. У випадку розпаду системи вони повинні виконувати свою функцію, але без системного ефекту. Незалежність елементів ІС необхідна для спрощення її розробки та супроводження. Відповідно до *закономірності прогресуючої систематизації* ІС система повинна постійно збільшувати свій системний ефект. Наприклад, в моделі за рівнянням 2 в якості величини системного ефекту можна взяти сумарний рівень рішення задачі R , який постійно зростає. Відповідно до *закономірності прогресуючої факторизації* елементи системи повинні постійно підвищувати свій рівень незалежності. Таким чином, розвиток ІС передбачає постійне підвищення величин системного ефекту і незалежності елементів.

Парадоксально, але відповідно до закономірностей інтегративності та "необхідної різноманітності" процеси систематизації («прогресу») повинні доповнюватись процесами факторизації («регресу»). Відомо, що у складних системах процеси «регресу» є неминучими та дозволяють долати граничні можливості системи і виходити їй на новий рівень розвитку. В ІС можна штучно чергувати фази систематизації та факторизації. Наприклад умови адитивності та регресу можна створити шляхом взаємодії елементів класів I,P або A,Pe .

6.1.2 Приклад класифікації елементів інформаційної системи

Нижче показано приклад класифікації елементів (або цілей) ІС підтримки ЖЦ ШСНУ. За її результатами можливо визначити класи бінарних відношень елементів і обчислити їхній рівень синергії. Це дозволить підвищити ефективність функціонування ІС. Табл. 6.2 містить значення відомих ознак ($x \in \{0,1\}$) класів елементів ІС моделі 8.2 на рівні 3.1 (рис. 6.1, табл. 6.1, Ф.1). Рекомендується також доповнити таблицю іншими ознаками, які відомі досліднику.

Таблиця 6.2 – Відомі ознаки класів елементів ІС моделі 8.2 на рівні 3.1

<i>i</i>	Назва класу u_j на рівні 3.1: Назва класу на рівні 2.2: Ознака x_i	<i>j</i>															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
		1a	1b	2a	2b	3a	3b	4a	4b	5a	5b	6a	6b	7a	7b	8a	8b
1	етап проектів (1) етап реалізацій (0)*	1 0,7	1 0,7	1 1,0	1 1,0	1 1,0	1 1,0	1 0,7	1 0,7	0 0,3	0 0,3	0 0,0	0 0,0	0 0,0	0 0,3	0 0,3	
2	теоретичний етап (1) практичний етап (0)*	1 1,0	1 1,0	1 1,0	1 1,0	1 0,7	1 0,7	0 0,3	0 0,3	0 0,0	0 0,0	0 0,0	0 0,0	0 0,3	0 0,3	1 0,7	1 0,7
3	етап концепцій (1) етап технологічний (0)*	1 1,0	1 1,0	1 0,7	1 0,7	0 0,3	0 0,3	0 0,0	0 0,0	0 0,0	0 0,0	0 0,3	0 0,3	1 0,7	1 0,7	1 1,0	1 1,0
4	етап експлуатацій і вимог (1) конструкторсько-технологічний (0)*	1 0,7	1 0,7	0 0,3	0 0,3	0 0,0	0 0,0	0 0,0	0 0,0	0 0,3	0 0,3	1 0,7	1 0,7	1 1,0	1 1,0	1 1,0	1 1,0
5	класи (1) об'єкти (0)	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
6	іраціональні етапи (1) раціональні етапи (0)	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1
7	процеси (1) цілі (0)	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
8	потік 1 (1) потік 2 (0)	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
9	властивості (1) відношення (0)	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0
10	зміст (1) обсяг (0)	1	0	1	0	0	1	0	1	0	1	0	1	1	0	1	0
11	кількість (1) якість (0)	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1
12	I	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
13	B	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
14	A	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
15	M	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
16	P	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
17	K	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
18	O	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
19	Pe	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0
20	етап науково-дослідницький	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
21	етап підготовки виробництва	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
22	сер. виробництва і дослідження збуту	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
23	етап збуту і експлуатації	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Примітка: *В нижньому ряду показані значення, що відповідають $x_i(t)$ (рис. 6.2)

Для підвищення точності опису класів в табл. 6.2 значення ознак повинні відповідати значенням функцій $x_i(t)$ (рис. 6.2). Зокрема значення ознаки $i=3$ повинні відповідати значенням функції $x_2(t)$ і визначатись за формулами $\cos(\pi/8+0,5(j-1)\pi/4)$ де $j \in \{1,3,5,7,9,11,13,15\}$ та $\cos(\pi/8+0,5(j-2)\pi/4)$ де $j \in \{2,4,6,8,10,12,14,16\}$. Таким чином, замість значень $x_3=(1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1)$, слід використовувати $x_3=(0,92, 0,38, -0,38, -0,92, -0,92, -0,38, 0,38, 0,92)$ або $x_3=(0,96, 0,69, 0,31, 0,04, 0,04, 0,31, 0,69, 0,96)$, якщо виконати перетворення $x_{3j}=x_{3j}/2+1/2$.

В таблиці 6.3 показані значення ознак деяких елементів ІС, які досліджувались в роботі. Граф усіх використаних в роботі елементів ІС (моделей, їхніх цілей та засобів) з ребрами «має ціль», «має засіб», «має базовий клас» показано на рис. Ф.1. Вибір значень ознак $\{x' \in R: 0 \leq x \leq 1\}$ доволі суб'єктивний, тому для підвищення точності варто застосовувати метод експертного оцінювання. Для визначення класу елемента ІС використовували критерій мінімальної (серед усіх класів 1-16) суми квадратів відхилень значень кожної ознаки:

$$s = \min \left(\sum_{i=1}^{23} (x_{ij} - x'_i)^2 \right),$$

де x_{ij} – значення ознаки i класу j елемента ІС на рівні 3.1 (табл. 6.2),

x'_i – значення ознаки i елемента ІС (табл. 6.3).

Якщо табл. 6.2 містить декілька класів з однією назвою (але, можливо, з іншими значеннями ознак), то для визначення класу потрібно застосувати один із методів задачі класифікації, наприклад метод k-сусідів.

Таблиця 6.3 – Приклад класифікації елементів ІС (класи ЖЦ) за значеннями їхніх ознак x'_i

№	Елементи ІС	$i=$																							s	j
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23		
1	Якісні дані про відмови	0	0	1	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	3	14	
2	База знань з проблем надійності РЗ	1	1	1	0	1	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	3	
3	Статистична модель відмов	1	1	1	0	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	0	0	2	1	
4	Технологічність пресового з'єднання ШН з ПКМ	1	0	0	0	0	0	1	1	1	0	1	0	0	½	0	½	0	0	0	0	1	0	2	5;7	
5	САПР пресового з'єднання ШН з ПКМ	1	1	1	0	1	0	0	1	1	1	1	½	0	½	0	0	0	0	0	0	1	0	3	3	

За результатами класифікації (табл. 6.3) раціональними будуть бінарні відношення елементів (1, 3), (3, 2) та (5, 4). Ці відношення близькі до класу відношень «Пряма послідовність» (табл. Ф.3), а їхні елементи мають спільні об'єкти досліджень. Тому їм характерні спільні цілі та високий рівень синергії. Системи, утворені з цих пар, можуть породити нові елементи, які мають більш загальні цілі. Вони теж класифікуються і процес цей повторюється, поки не будуть досягнуті задані цілі на вищих рівнях ієрархії цілей. Також доцільно провести аналогічну класифікацію елементів ІС, які використовуються на етапі проектування, і визначити їхній клас етапу проектування і класи відношень.

6.1.3 Алгоритм інформаційної системи на основі загальносистемних закономірностей

На основі міждисциплінарних досліджень складних систем можна запропонувати наступний алгоритм функціонування ІС [357]. Алгоритм включає механізми ізоморфні до механізмів соціальних систем і біологічної еволюції (соціальні групи, політика, природній добір і мутація).

1. Створюється початкова множина з гетерогенними незалежними функціональними елементами ІС. Функціональний елемент може розв'язати частину задачі, має цілі та володіє інформаційним входом і виходом для взаємодії з іншими елементами. Це можуть бути лінгвістичні ресурси, бази даних, бази знань, прикладні програми, комп'ютерні моделі, апаратне забезпечення, користувачі. Різноманіття елементів забезпечує стійкість ІС, так само як біорізноманіття забезпечує стійкість біоценозів. Якщо синергія механізмів мутації та природного добору є причиною біорізноманіття, то і ІС повинна володіти подібними механізмами.

2. Вибирається множина ознак (координатних осей) для класифікації елементів (рис. 6.3). Наприклад можна взяти дві ознаки: етап ЖЦ (*A*) та рівень динаміки системи (*B*) [213].

3. Відповідно ознак множина функціональних елементів ІС ділиться на класи методом дихотомії [213] таким чином, щоб класи мали різні цілі (рис. 6.3). Розташовані рядом на осі класи повинні мати спільні цілі на вищому рівні

ієрархії. У найпростішому випадку можна обмежитись двома класами за кожною ознакою, наприклад {"етап 1", "етап 2"}, {"статика", "динаміка"}. Якщо поділити кожен етап ЖЦ на два класи ще раз, то отримаємо всього 8 класів A_i, B_j , де $i \in \{1, 2, 3, 4\}$ – етап ЖЦ, $j \in \{1, 2\}$ – рівень динаміки (рис. 6.3б).

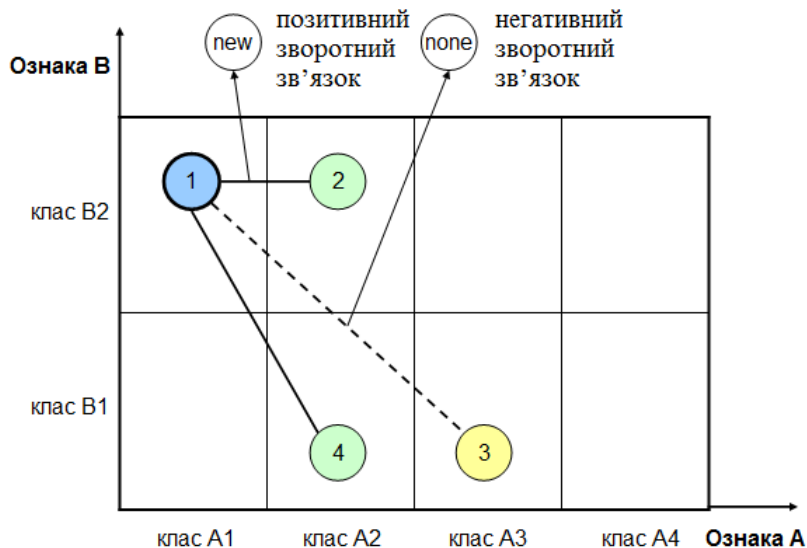
4.Вибирається модель взаємодії класів. Опираючись на принцип ізоморфізму, для побудови моделей взаємодії класів, наприклад, може бути використаний аналіз різних типів ідеологічних і політичних систем в суспільстві. Очевидно, що ці типи відповідають задачам, які виникають перед суспільством, і етапам ЖЦ суспільства. Модель взаємодії може генерувати спеціальний елемент ІС "координатор". Він повинен вибиратись тимчасово з того класу, який відповідає задачі. Наприклад можна запропонувати синергетичну з точки зору подібності цілей модель взаємодії – елементи взаємодіють тільки з сусідніми класами на координатній осі, наприклад (1, 2) (рис. 6.3). В такій моделі є спільні цілі та виникає мінімум конфліктів. У випадковій моделі взаємодіють довільні пари елементів, наприклад (1, 3), що може призводити до дисинергії. Взаємодія може бути послідовна (1, 2) або паралельна (2, 4) за ознакою "етап". Відповідно закону "необхідної різноманітності" необхідно забезпечити різноманітність таких моделей і дотримуватись рівноваги між "порядком" і "хаосом". Елемент може мати пам'ять, в якій зберігаються оцінки взаємодій і яка використовується для обґрунтування вибору партнера.

5.Виконується взаємодія елементів. В результаті взаємодії можуть утворитися нові елементи (рішення), які додаються в множину. Якщо є новий результат (*new*), то елемент запам'ятовує позитивну оцінку взаємодії (позитивний зворотний зв'язок). У іншому випадку (*none*) запам'ятовується негативна оцінка (негативний зворотний зв'язок).

6.Відбувається повтор – перехід до п. 5 (або 4, або 3, або 2). Таким чином формується ЖЦ ІС і його ієрархія.

Отже такій ІС будуть властиві усі базові закономірності складних адаптивних систем. В даному випадку дихотомічна класифікація забезпечує емерджентність, інтегративність та ієрархічність; незалежність елементів – адитивність; різнотипність елементів і відношень – "необхідну різноманітність"; ЖЦ ІС – історичність, еволюцію і зворотні зв'язки; різнотипна координація –

самоорганізацію. Реалізувати таку ІС можна мовою програмування загального призначення Python, на основі принципів запропонованих в праці [32]. Кожен елемент є незалежним Python-класом (агентом), в якому описано алгоритм його функціонування.



(—) – синергетичні відношення; (- -) – дисинергетичні відношення

Рисунок 6.3 – Відношення між елементами ІС

6.2 Принципи створення баз знань з проблем надійності обладнання ШСНУ

6.2.1 Використання мови OWL

На даний час накопичено багато знань, пов'язаних з проблемами надійності та довговічності РЗ, в тому числі РЗ ШСНУ. Проте існує проблема ефективного використання цих знань. Як правило основним джерелом знань, пов'язаних з проблемами надійності та довговічності РЗ є наукова література [71, 358-361]. Проте в цьому вигляді ці знання не доступні для машинної обробки. Для збереження їх у базі знань необхідно застосувати один з методів подання знань. Не важко переконатись, що в загальному знання в цих джерелах представлені у вигляді фактів, які містять причинно-наслідкові зв'язки між різноманітними факторами і відмовами (або ймовірностями відмов): *<факт> = <фактор> <є причиною> <відмови>*. Цим фактам можуть бути властиві різноманітні закономірності, пояснення та посилання на джерела інформації: *<факт> <має залежність> <залежність>*, *<факт> <має джерело> <джерело>*. Для

представлення таких знань зручно застосувати семантичні мережі, які кодуються мовою OWL.

Автором запропоновано принципи розробки бази знань з проблем надійності та довговічності РЗ [29, 93, 362]. Ця база знань містить поняття відмов та факторів, які збільшують чи зменшують імовірність відмов, і причинно-наслідкові зв'язки між цими поняттями. Вона може бути використана інженерами-конструкторами під час проектування РЗ та вченими, які займаються питаннями підвищення надійності та довговічності РЗ. В якості джерел експертної інформації були вибрані літературні джерела, зокрема [71, 321, 358]. Для подання знань були вибрані семантичні мережі, які кодувались мовою опису онтологій OWL Lite.

Після того як були визначені задачі та мета розробки, користувачі, джерела експертної інформації, виконувався змістовний аналіз предметної області, виявлялись поняття та їхні взаємозв'язки. Будувалась ієрархія класів понять. Клас OWL (*Class*) визначає групу індивідів, яких об'єднує наявність деяких загальних властивостей [283]. Клас може бути підкласом іншого класу (*rdfs:subClassOf*). Приклад ієрархії класів, у якій підкласи вказані в дужках: *Фактор* (*Відмова* (*Втомна відмова*, *Статична відмова*, ...)), *Деталі* (*Ніпель*, *Муфта*, ...), *Концентрація напружень*, *Матеріал*, *Навантаження*, *Середовище*, *Складання* (*Момент згинчування*, ...), ...).

Індивідами OWL (*Individual*) називають представників класу. Для пов'язування індивідів між собою використовуються властивості (*rdf:Property*). Властивості поділяються на властивості-об'єкти (*ObjectProperty*) – їхніми значеннями є індивіди, і властивості-значення (*DatatypeProperty*) – їхніми значеннями є дані типів XML Schema, наприклад *xsd:string*, *xsd:integer*, *xsd:dateTime* [283]. Так в нашу онтологію додані об'єктні властивості «є причиною» і «є наслідком». Кожна властивість володіє областю визначення (*rdfs:domain*), яка обмежує індивідів, до яких ця властивість може бути застосована, та діапазоном (*rdfs:range*), який обмежує індивідів, які можуть бути значенням цієї властивості [283]. Областю визначення і діапазоном наших властивостей може бути базовий клас «Фактор». Властивості в OWL Lite можуть мати такі характеристики, які дозволяють їм бути інверсними (*inverseOf*),

транзитивними (*TransitiveProperty*), симетричними (*SymmetricProperty*), мати унікальне значення (*FunctionalProperty*) [283]. Так, властивості «є причиною» і «є наслідком» є транзитивними і взаємно інверсними. Наприклад, їхня інверсність дає змогу з твердження 1 (табл. 6.4) зробити логічний висновок – твердження 2, а транзитивність дає змогу з тверджень «нерівномірний розподіл навантажень по витках різьби є причиною концентрації напружень» і «концентрація напружень є причиною збільшення втоми» зробити логічний висновок: «нерівномірний розподіл навантажень по витках різьби є причиною збільшення втоми». Властивості також можуть утворювати ієрархію за допомогою конструкції *rdfs:subPropertyOf* [283]. Так, властивість «є причиною» могла б мати підвластивості «є причиною збільшення» та «є причиною зменшення». В онтологію додано також властивість «величина» з діапазоном *xsd:string*, яка характеризує відносну величину фактора (великий, малий, оптимальний). Для прикладу, її значення «великий» може означати: «збільшення», «підвищення», «значний», «наявність» і т.д. Ця властивість дозволяє створювати такі індивіди, як «висока концентрація напружень», «низька концентрація напружень», «оптимальна довжина зарізьбової канавки» і т.д. OWL Lite підтримує також обмеження властивостей (вони задають, які значення можуть використовуватись представниками класів), обмежену кардинальність (скільки значень може використовуватись) та перетин класів [283]. Проте ці конструкції не використовувались в нашій онтології.

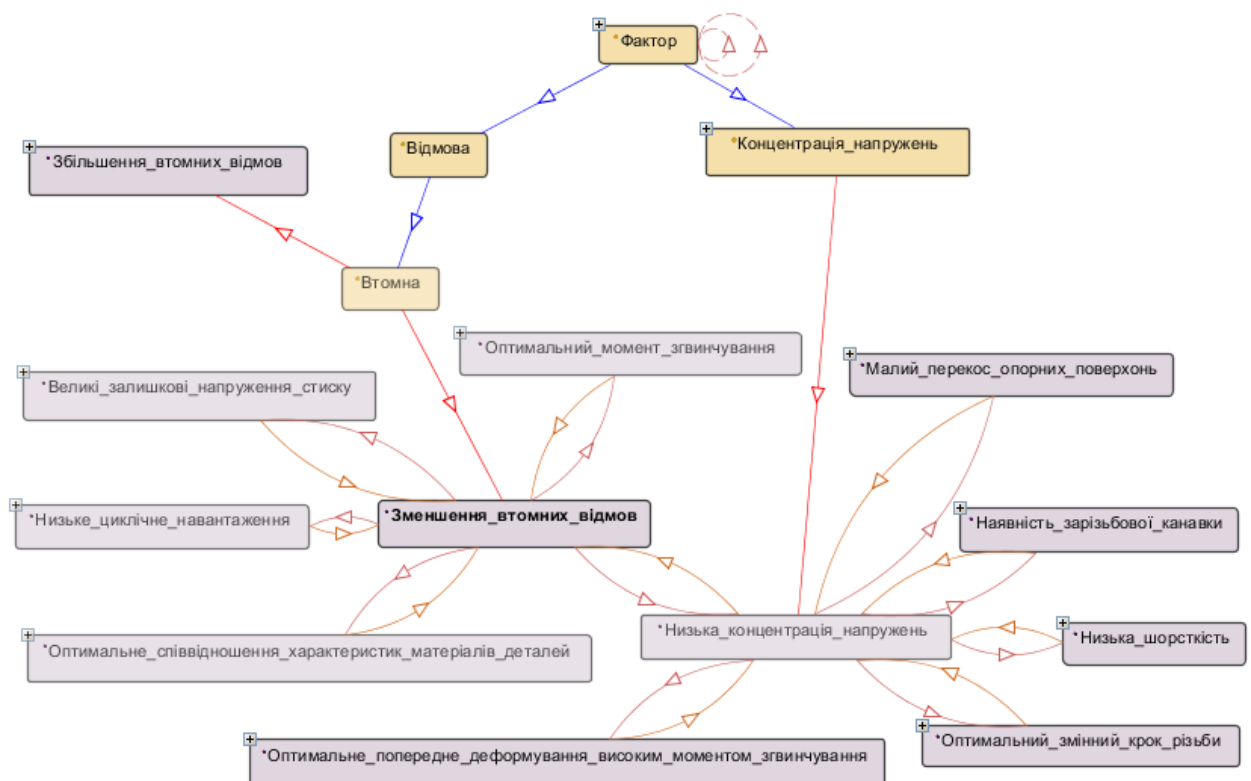
Далі інформація з джерел перетворювалась в чітко формалізований вигляд для того, щоб її можна було закласти в базу знань. Це найбільш відповідальний і складний етап, оскільки не будь-які вихідні факти можна легко формалізувати. Наприклад твердження «міцність згвинченого РЗ під час змінних навантажень вища, ніж незгвинченого» [71] можна формалізувати наступним чином (табл. 6.4).

Таблиця 6.4 – Факти бази знань у формалізованому вигляді

№п/п	Суб'єкт	Предикат	Об'єкт
1	«згвинчування» (індивід класу Фактор)	«є причиною» (об'єктна властивість)	«зменшення втоми» (індивід класу Фактор)
2	«зменшення втоми» (індивід класу Фактор)	«є наслідком» (об'єктна властивість)	«згвинчування» (індивід класу Фактор)

Покажемо приклад створення індивідів «Збільшення втомних відмов» та «Зменшення втомних відмов». Для першого індивіда значенням властивості «величина» є «великий», значеннями властивості «є наслідком» будуть індивіди «Значне циклічне навантаження», «Висока концентрація напружень», «Неоптимальний момент згвинчування» і т.д. Для другого індивіда властивість «величина» має значення «малий», а властивість «є наслідком» – «Низька концентрація напружень», «Оптимальний момент згвинчування», «Оптимальне співвідношення характеристик матеріалів болта та гайки» і т.д.

Protégé [286] містить засоби перевірки онтології на наявність логічних протиріч, має багато додаткових модулів, зокрема для візуалізації онтології (рис. 6.4), а також засоби розробки запитів до бази знань, наприклад мовою SPARQL. Усе це дає змогу використовувати Protégé разом з розробленою базою знань як повноцінну експертну систему.



дуги – інверсні властивості «є наслідком»-«є причиною»,

прямі лінії – предикати «має підклас» та «має екземпляр»

Рисунок 6.4 – Візуалізація частини онтології у вигляді семантичної мережі

В основному розроблена автором онтологія містить множину індивідів класу "Фактор", які пов'язані між собою транзитивними і взаємно інверсними властивостями "isCause" (є причиною) і "isEffect" (є наслідком). Індивіди класу "Фактор" мають також властивості "Not", "And", "Or", які дозволяють створення нових факторів з існуючих за допомогою відповідних логічних операцій. Ще одна транзитивна властивість індивідів класу "Фактор" "SubClassOf" (не плутати з `rdfs:subClassOf` [283]) дозволяє створювати нові індивіди, які успадковуватимуть значення властивостей "isCause" і "isEffect" свого базового індивіда. Подібну функціональність можна було б реалізувати і за допомогою `rdfs:subClassOf`, яка пов'язує клас з більш загальним класом.

Онтологія також може містити класи "Залежність", "Джерело", "Факт". Клас "Залежність" дозволяє детально описати залежність між причиною і наслідком, наприклад за допомогою графічної залежності. Клас "Джерело" описує посилання на джерело інформації. Клас "Факт" описує множину фактів у вигляді триплетів ("Фактор", "isCause/isEffect", "Фактор"). Для пов'язування фактів і факторів між собою факти повинні мати властивості "суб'єкт", "об'єкт" (мають значення індивідів класу "Фактор") та "предикат" (має значення "isCause" або "isEffect"). Індивіди класу "Факт" також володіють властивостями "має залежність" і "має джерело".

Покажемо приклади створення запитів до нашої бази знань мовою SPARQL. Наступний запит знаходить усі наявні RDF триплети онтології.

```
SELECT * WHERE {?s ?p ?o}
```

Іншими словами, цей запит вибирає все, що відповідає шаблону триплета `{?s ?p ?o}`, де `?s`, `?p`, `?o` – змінні. Перегляд результатів цього запиту дає змогу легко зрозуміти сутність RDF. Деякі результати запиту у вигляді триплетів: (:Збільшення_втомних_відмов, `rdf:type`, :Втомна), (:Збільшення_втомних_відмов, :величина, «великий»), (:Висока_шорсткість, :є_причиною, :Висока_концентрація_напружень), (:Момент_згвинчування, `rdfs:subClassOf`, :Складання), (:Геометричні_параметри, `rdf:type`, `owl:Class`), (:є_причиною,

`rdf:type, owl:ObjectProperty), (:є_причиною, rdf:type, owl:TransitiveProperty),`
`(:є_причиною, rdfs:domain, :Фактор), (:є_причиною, rdfs:range, :Фактор),`
`(:є_наслідком, owl:inverseOf, :є_причиною), (:величина, rdfs:range,`
`owl:oneOf{«великий» «малий» «оптимальний» «неоптимальний»}), (:величина,`
`rdf:type, owl:DatatypeProperty).`

Наступний запит знаходить усі прямі (безпосередні) причини зменшення втомних відмов.

```
SELECT ?x WHERE {?x :є_причиною :Зменшення_втомних_відмов}
```

Інший спосіб цього запиту слід читати так: знайти все, що є причиною екземплярів класу «Втомна» зі значенням властивості «величина» «малий»:

```
SELECT ?x
WHERE {_:y rdf:type :Втомна . _:y :величина "малий" . ?x :є_причиною _:y .}
```

Наступний запит знаходить усі прямі причини зменшення втомних відмов та усі прямі причини цих причин. Іншими словами, так можна знайти усі прямі та непрямі причини зменшення втомних відмов. Тут ключове слово **OPTIONAL** позначає необов'язкову складову шаблону і необхідне, оскільки непрямих причин може і не існувати.

```
SELECT * WHERE {{?x :є_причиною :Зменшення_втомних_відмов}
OPTIONAL {?y :є_причиною ?x}}
```

Наступний запит знаходить усі прямі причини зменшення втомних відмов, назва яких містить вирази "напруж" або "різьб".

```
SELECT ?x WHERE {?x :є_причиною :Зменшення_втомних_відмов.
FILTER (regex(str(?x),"напруж","i") || regex(str(?x),"різьб","i"))}
```

Після того як в базу знань були введені усі факти, необхідно створити правила логічного висновку. Для цього можна застосувати мову SWRL. В нашу базу знань були додані наступні два правила:

```
default:SubClassOf(?x,?y)^default:isCause(?y,?p)→default:isCause(?x,?p)
default:Not(?x,?nx)^default:isCause(?nx,?ny)^default:Not(?y,?ny) →
default:isCause(?x,?y)
```

Ці правила вводяться в *SWRLTab* – розширенні Protégé, яке дозволяє редагування та виконання SWRL правил. Для роботи з SWRLTab необхідний механізм виконання правил (rule engine) Drools (входить в Protégé 3.5) або Jess (необхідна ліцензія, встановлюється окремо). Обое цих механізмів використовують розширену версію алгоритму Rete, який дозволяє ефективно обробляти велику кількість правил. Фактично вони і є машиною виведення нашої експертної системи, розробленої в межах Protégé. Запуск Drools показав, що наша онтологія містить 523 індивідів, 2508 введених аксіом і 16418 виведених аксіом. Онтологія з новими аксіомами може бути збережена в OWL.

Після заповнення бази знань можна створювати запити до неї. SWRLTab має плагін *SQWRLQueryTab*, який дозволяє виконувати запити до бази знань мовою *SQWRL* (*Semantic Query-Enhanced Web Rule Language*). Це мова основана на SWRL для запитів в OWL онтологіях. Наприклад:

```
default:isCause(?x,default:Концентрація_напружень.
Під_головкою_болта.Низька) → sqwrl:select(?x)
```

Цей запит виведе усі фактори, які є причиною зменшення концентрації напружень під головкою болта. Наприклад, серед них є фактор *default:Деталі.Шайба.З_низьковуглецевої_сталі*. Факт про те, що цей фактор є причиною зменшення концентрації напружень під головкою болта відсутній серед введених аксіом, але є виведений за допомогою машини виведення. Попередній запит мовою SPARQL виглядатиме так:

```
SELECT ?x WHERE
{?x default:isCause default:Концентрація_напружень.Під_головкою_болта.Низька}
```

6.2.2 Використання мови Python

На основі описаного підходу розроблено принципи побудови експертної системи (ЕС) з проблем надійності РЗ мовою Python [30, 363, 364]. Класи розробленої онтології створені на основі класів мови Python, індивіди – на основі об'єктів Python, атрибути і відношення – на основі атрибутів об'єктів Python.

Відношення між індивідами онтології створюються за допомогою об'єктів спеціального класу `Property` (властивість) (лістинг X.2). Об'єкти цього класу мають атрибути `subj` (суб'єкт властивості – це індивід, який має дану властивість), `name` (назва властивості), `inverseName` (назва інверсної властивості), `functional` (визначає чи властивість функціональна), `symmetric` (визначає чи властивість симетрична), `transitive` (визначає чи властивість транзитивна), `set` (множина значень властивості) та методи `__init__()` (конструктор, створює властивість), `add()` (додає об'єкт в множину) і `get()` (повертає множину значень властивості). В даній онтології використовуються такі властивості: "Not" (симетрична), "And", "Or", "SubClassOf" (транзитивна), "isCause" (інверсна до "isEffect", транзитивна), "isEffect" (інверсна до "isCause", транзитивна), "hasReference" (інверсна до "isReference"), "isReference" (інверсна до "hasReference"), "hasDependence" (інверсна до "isDependence"), "isDependence" (інверсна до "hasDependence").

Базовий клас онтології "Base" (лістинг X.2) має властивості "Not", "And", "Or", які дозволяють створення нових індивідів з існуючих за допомогою відповідних логічних операцій, та властивість "SubClassOf", яка дозволяє створювати нові індивіди шляхом успадкування значень заданих властивостей базового індивіда.

Онтологія також містить класи "Фактор", "Факт", "Джерело", "Залежність" та інші класи, які успадковують клас "Base". Клас "Фактор" описує множину факторів, які впливають на надійність РЗ. Приклад створення індивіда "Корозійне пошкодження" класу "Фактор": `KB[r""Корозійне пошкодження""]=KB[r""Фактор""]()`. Класи та індивіди онтології зберігаються в словнику KB. Тут `r""Корозійне пошкодження""` – необроблюваний рядок, який використовується як ключ об'єкта в словнику KB. Клас "Фактор" має властивості "isCause" (є причиною) і "isEffect" (є наслідком). Значеннями цих властивостей є множина індивідів класу "Фактор". Наприклад твердження

"Корозійне пошкодження є причиною зменшення міцності" можна формалізувати так:

```
KB[r"Корозійне пошкодження"].__dict__["isCause"].add(KB[r"Міцність\Зменшення"])
```

Клас "Факт" описує множину фактів у вигляді триплетів суб'єкт-предикат-об'єкт. Для пов'язування фактів і факторів між собою факти повинні мати атрибути `subjName` (назва фактора-суб'єкта), `objName` (назва фактора-об'єкта) та `propName` (назва предиката – властивості "isCause" або "isEffect"). Наприклад попередній факт можна формалізувати і таким чином:

```
KB[r"Корозійне пошкодження є причиною зменшення міцності"]=KB[r"Факт"]()
X=KB[r""Корозійне пошкодження є причиною зменшення міцності""]
X.create(r""Корозійне пошкодження"", 'isCause', r""Міцність\Зменшення"")
```

Тут функція `create` присвоює значення атрибутам `subjName`, `objName`, `propName` та додає об'єкт `objName` в множину значень властивості `propName` суб'єкта `subjName`.

Клас "Залежність" дозволяє детально описати залежність між причиною і наслідком, наприклад за допомогою графічної залежності. Він має властивість "isDependence" (є залежністю), атрибут `xu` (дані залежності) і функції, які дозволяють будувати графік та знаходити значення інтерполяцією. Клас "Джерело" описує посилання на джерело інформації. Він має властивість "isReference" (є джерелом). Для пов'язування фактів з джерелами та залежностями клас "Факт" має властивості "hasReference" (має джерело) і "hasDependence" (має залежність). Об'єкти цього класу необхідно створювати в тому випадку, коли вони мають конкретні значення цих властивостей. Наприклад для факту X:

```
X.hasReference.add(KB[r""И.А.Биргер, Резьбовые и фланцевые соединения""])
X.hasDependence.add(KB[r""Залежність статичної міцності від ступеня корозійного пошкодження""])
```

Клас "Модель" описує параметричну модель РЗ для САД/САЕ систем. Клас "Параметр" описує параметр моделі. Підтримка Python повної інтроспекції часу виконання дозволяє легко програмувати механізми виведення та створювати запити до бази знань. Логічне виведення реалізоване простим алгоритмом:

ПОКИ ІСТИНА:

```
N1 = кількість фактів в базі знань
Застосувати правила виведення до кожного факту
базі знань і додати виведені факти
N2 = кількість фактів в базі знань
Зупинити цикл, якщо N1=N2
```

В даному випадку були застосовані такі правила логічного висновку:

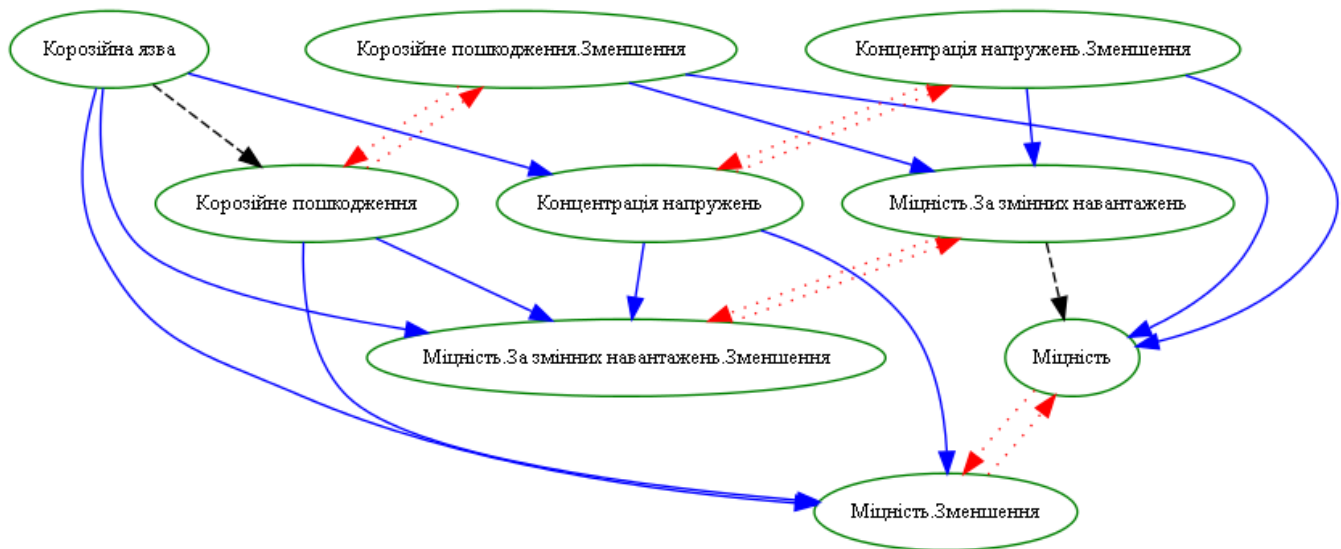
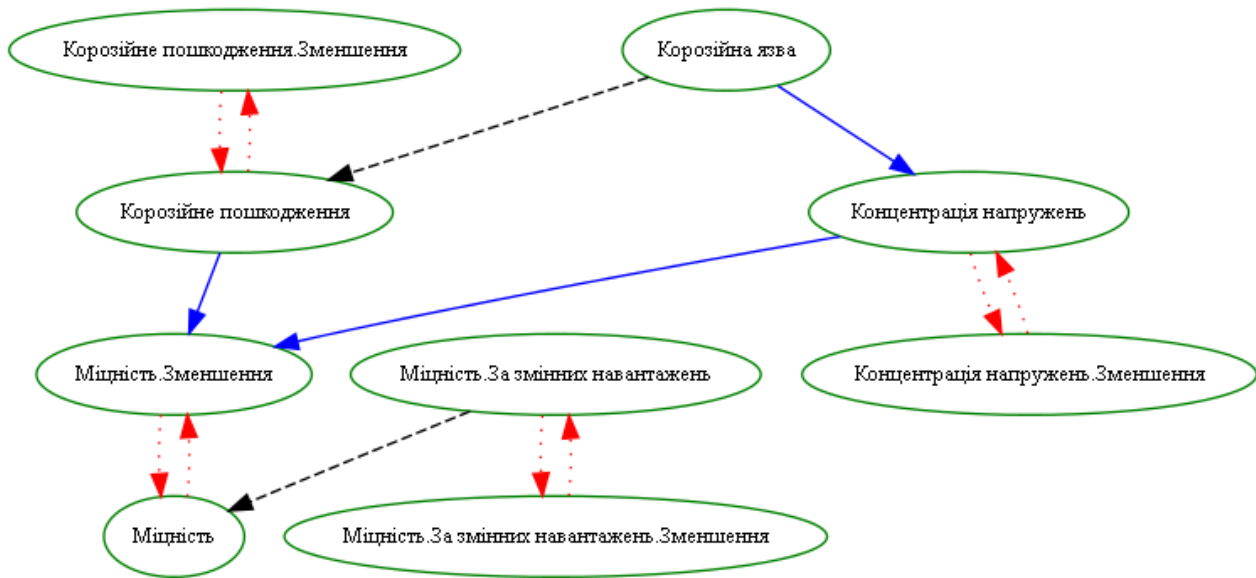
1. Логічне виведення для інверсних, симетричних і транзитивних властивостей.
2. `SubClassOf(?x, ?y) & isCause(?y, ?z) -> isCause(?x, ?z)`
3. `SubClassOf(?x, ?y) & isEffect(?y, ?z) -> isEffect(?x, ?z)`
4. `Not(?x, ?nx) & isCause(?nx, ?ny) & Not(?y, ?ny) -> isCause(?x, ?y)`
5. `Not(?x, ?nx) & isEffect(?nx, ?ny) & Not(?y, ?ny) -> isEffect(?x, ?y)`

Для великої кількості фактів і правил можна застосувати більш швидкодіючі алгоритми, наприклад алгоритм співставлення зі взірцем Rete, який реалізований в безкоштовних системах Rychinko+FuXi, Pyke, Drools. Ці механізми виконання правил можна легко під'єднати до даної експертної системи.

Запити до бази знань створюються мовою Python. Приклад простого запиту, який виводить усі причини фактора "Міцність\Зменшення":

```
for x in KB[r"Міцність\Зменшення"].isEffect(True):    print x.name
```

Можна створювати запити будь-якої складності, виводити результати в форматі таблиць CSV, графів мовою DOT (рис. 6.5) та онтологій OWL у форматі Turtle.



властивості: isCause (—), SubClassOf (- - -), Not (· · ·)

а) – введені аксіоми; б – виведені аксіоми

Рисунок 6.5 – Семантична мережа демонстраційної бази знань

Користувач може створювати власні способи подання і виведення знань. За допомогою відповідних скриптів-інтерфейсів запити можуть звертатися до CAD/CAE систем (SolidWorks®, Abaqus®, fe-safe®) за результатами моделювання РЗ в цих системах [365]. Результатами моделювання можуть бути значення об'єму, маси, деформацій, еквівалентних напружень, циклічної довговічності тощо. Зокрема ЕС може обробляти наступні запити: знайти усі причини втомних руйнувань РЗ; знайти усі наслідки недостатнього моменту згинчування, які призводять до руйнування болта, та відповідні джерела інформації; знайти оптимальне значення довжини розвантажувальної канавки за критерієм втомної міцності для конкретних умов роботи заданого РЗ.

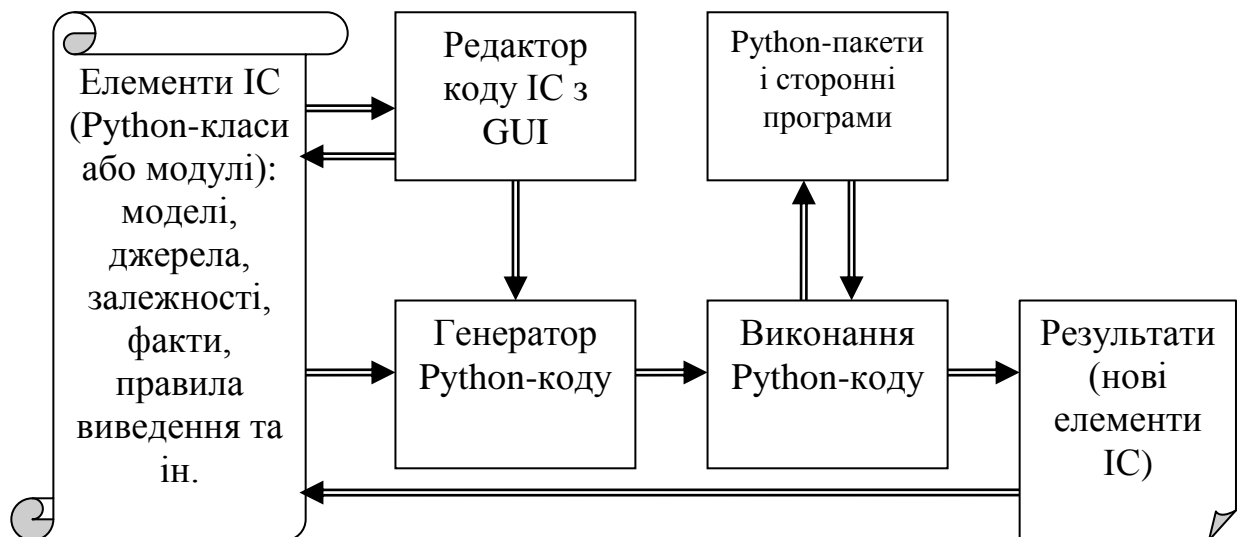
База знань, машина виведення і запити розташовані в модулі Python, який повинен мати блочну структуру – спочатку створюються класи онтології, потім індивіди, потім задаються властивості індивідів, потім виконується логічне виведення, а в кінці виконуються запити до бази знань. Для спрощення створення таких модулів автором розроблена програма-редактор експертних систем з графічним інтерфейсом (GUI) на основі Tkinter [365, 366] (URL: <https://github.com/vkorey/TreePyKB>). Інтерфейсом (рис. 6.6б) розроблена програма подібна на традиційний аутлайнер (outliner) – програму, яка дозволяє організувати текст в деревовидну структуру або ієрархію. На рисунку показано головне вікно, вікна меню (File, Edit, Text) та вікно для заповнення шаблону коду (Dialog Template). Зліва у вікні програми знаходиться дерево каталогів експертної системи. Частина python-коду експертної системи знаходяться в цих каталогах у текстових файлах з розширенням `pykb`. Каталоги з файлом `class.pykb` відповідають класам або індивідам онтології. Справа подається Python-код вибраного файлу `pykb`. Наприклад каталог зі значенням відносної адреси "Base\Фактор\Концентрація напружень\Під головкою болта" відповідає індивіду онтології, який має це ж значення назви (атрибут `name`) і ключа в словнику KB. У файлі `class.pykb` цього каталогу необхідно створити цей об'єкт:

```
KB[r"Base\Фактор\Концентрація напружень\Під головкою болта"]=KB[r"Фактор"]()
```

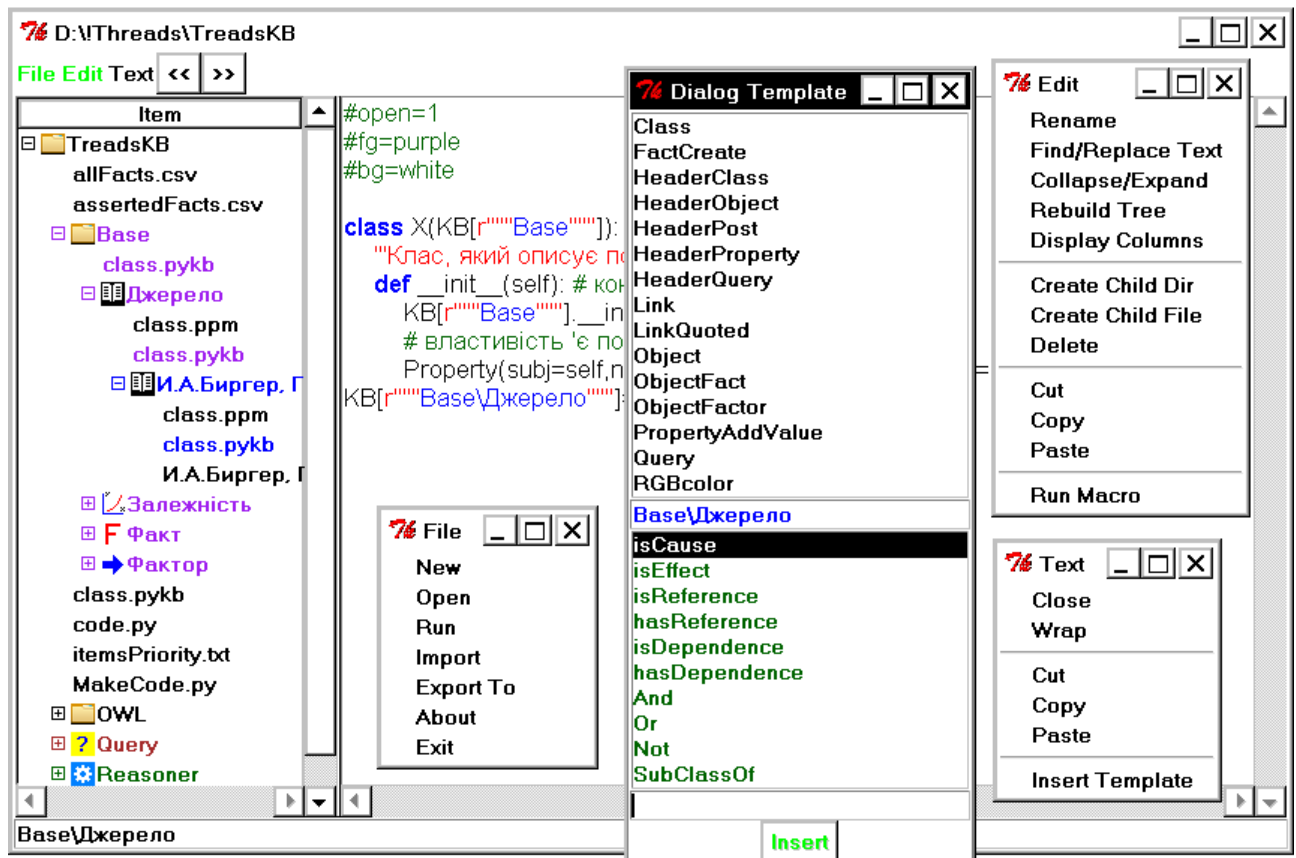
Перші три рядки кожного файлу `рукb` можуть бути Python-коментаріями і задавати властивості елементу дерева каталогів: стан гілки каталогу (розкрита чи закрита), колір шрифту, колір фону. В цьому ж каталозі можна створити файл `проп.рукb`, в якому будуть задаватись значення властивостей об'єкту:

```
KB[r""Base\Фактор\Концентрація напружень\Під головкою
болта""].SubClassOf.add(KB[r""Base\Фактор\Концентрація напружень""])
KB[r"Base\Фактор\Концентрація напружень\Під головкою болта"].Not.add(
    KB[r"Base\Фактор\Концентрація напружень\Під головкою болта\Низька"])
KB[r"Base\Фактор\Концентрація напружень\Під головкою болта"].isCause.add(
    KB[r""Base\Фактор\Пошкодження\Втомна тріщина\Під головкою болта""])
```

Каталоги "Base", "Base\Фактор", "Base\Фактор\ Концентрація напружень" теж відповідають окремим об'єктам бази знань – базовому класу, класу, який описує фактори, і його індивіду відповідно. Саме розташування каталогу "Під головкою болта" в каталозі "Концентрація напружень" ще не означає, що другий є індивід є узагальненням першого. Але таке узагальнення можна створити за допомогою властивості "SubClassOf", як це показано в прикладі вище. Факти бази знань (461 факт) у вигляді триплетів (суб'єкт; предикат; об'єкт) наведено у додатку X (лістинг X.1). Після вибору меню File/Run усі файли `рукb` об'єднуються в один Python-модуль (лістинг X.2), який динамічно виконується за допомогою команди Python `exec` (рис. 6.6a). Послідовність об'єднання файлів `рукb` різного типу описується у файлі `itemsPriority.txt`. Для ідентифікації типу використовується колір шрифту елемента (fg). Використання файлової системи забезпечує унікальність назв об'єктів, їхню ієрархію, швидку навігацію та дозволяє зберігати в каталогах додаткові файли, які мають відношення до експертної системи. Описані принципи побудови бази знань лягли в основу багатоагентної ІС, що розглядається в наступному підрозділі.



а



б

Рисунок 6.6 – Схема експертної системи (а) та графічний інтерфейс редактора експертних систем (б)

6.3 Принципи реалізації багатоагентної інформаційної системи для проектування і підтримки життєвого циклу обладнання ШСНУ

Нижче описані принципи розроблення ІС різьбових з'єднань ШСНУ [31, 32] та роботи з нею. Вище було сказано, що з метою забезпечення ефективності ІС повинна бути ізоморфною до інших складних систем та володіти їхніми закономірностями, зокрема закономірностями емерджентності та «необхідної різноманітності» [213]. Запропонована система відноситься до класу гібридних інтелектуальних систем, які поєднують різні методи подання знань і прийняття рішень. Система містить базу знань з правилами логічного виведення, машину прямого виведення, редактор коду, імітаційні моделі РЗ, результати їхніх симуляцій та інші компоненти (рис. 6.5а). База знань містить факти, у яких фактори, що впливають на надійність та довговічність РЗ, пов'язані причинно-наслідковими відношеннями "є причиною" і "є наслідком", як це описано в попередньому розділі. Факти можуть мати такі властивості як джерело інформації, залежність величин, імітаційна модель тощо. Правила логічного виведення дозволяють отримувати нові факти з бази знань та імітаційних моделей. Усі компоненти системи або програмні інтерфейси до них розробляються високорівневою мовою програмування загального призначення Python, що дозволяє спростити реалізацію системи та об'єднання різнотипних компонентів. Зокрема до системи можна легко під'єднати будь-який пакет Python (рис. 6.6а). Класи та індивіди онтології декларативно описуються класами і об'єктами Python, атрибути і відношення – атрибутами об'єктів Python. Для зручного редагування код системи поділяється на частини, які перед виконанням об'єднуються в один Python-модуль (рис. 6.6а).

ІС призначена для прийняття оптимальних рішень під час проектування, виробництва, ремонту та експлуатації РЗ. Оптимальні рішення можна приймати за критеріями надійності та довговічності РЗ. Код прикладу системи наведено в додатку Ц (лістинги Ц.1-Ц.42).

Система складається з об'єктів (агентів), що створюються на основі класів, похідних від класу **Agent**. Це елементи ІС (рис. 6.6а). Усі об'єкти зберігаються в словнику KB, де ключами є унікальні імена об'єктів – значення їхнього атрибута `__name__`. Модуль `tools.py` (лістинг Ц.2) містить словник KB і допоміжні функції для роботи з системою, доступні для агентів. Об'єкти можуть створюватись динамічно під час діалогу користувача з системою або генеруватись іншими об'єктами під час роботи системи. У зв'язку з цим, зручніше кожен об'єкт описувати в окремому Python-модулі. З назви файлу цього модуля (`ClassName_AgentName.py`) автоматично визначається клас (`ClassName`) і назва (`AgentName`) об'єкта. Запобігати конфлікту імен можна, наприклад, за допомогою функції для пошуку імен за регулярним виразом `find`, методу словника `has_key` або функції для генерації унікальних імен `autoKey`. Функція `createKB` для кожного такого файлу створює об'єкт `obj` відповідного класу, додає його в словник KB за ключем `obj.__name__` і виконує цей файл як Python-код в просторі імен `obj.__dict__` за допомогою `execfile` (функція для динамічного виконання Python-коду). Це дозволяє звертатись до атрибутів об'єкта в цьому файлі без застосування імені об'єкта і, таким чином, скоротити об'єм коду.

Базовий клас Agent (лістинг Ц.3) описує об'єкти, що володіють методом `rule`, атрибутом `active` (визначає чи об'єкт активний) та атрибутом `__name__` (ім'я). Усі інші, описані нижче, класи успадковують клас **Agent**. Метод `rule` визначає правило поведінки агента. Метод повинен повертати `True`, якщо його виклик призвів до яких-небудь змін атрибутів агентів. У іншому випадку метод повинен повертати `False` або `None`.

В загальному випадку правила активних агентів (з атрибутом `active=True`) застосовуються до тих пір, поки це призводить до змін в системі. Це забезпечує розвиток системи і є передумовою самоорганізації, зокрема може бути реалізована класифікація агентів (пункт 6.1.2) та алгоритм, описаний в пункті 6.1.3. Одноразове застосування правил для усіх агентів назвемо ітерацією. Ітерації можуть виконуватись у функції `applyRules(lst=KB, n=5)`. Якщо

потрібно, то функція `applyRules` може застосовувати правила тільки до об'єктів зі списку `lst`. Параметр функції `n` визначає максимальну кількість ітерацій.

Функції `saveKB` та `loadKB` можуть бути використані для збереження і завантаження агентів з постійної пам'яті. Для цього в них використовуються пакет `dill` [367] і стандартний модуль `shelve`. Використання цих функцій може бути корисним у випадку тривалих ітерацій. Крім того, збереження значень атрибутів у вихідному коді агентів може виконуватись користувачем.

Такий підхід до побудови програмних систем пов'язаний з декларативним програмуванням [168, 368] і дозволяє суттєво спростити розв'язання деяких задач.

Клас `Factor` (лістинг Ц.9) описує агенти-фактори, що впливають на надійність та довговічність РЗ. Кожен фактор володіє атрибутами `isCause` (є причиною), `isEffect` (є наслідком), `subclassOf` (є підкласом фактора), `isContraryOf` (є протилежністю фактора). Ці атрибути можуть містити множину назв інших факторів. Таким чином фактори можуть бути з'єднані відношеннями за допомогою цих атрибутів. Зокрема атрибути `isCause` та `isEffect` можуть створювати причинно-наслідкові зв'язки між факторами. Наприклад опис фактора "корозійне пошкодження" зроблено у файлі "Factor_корозійне пошкодження.py" (лістинг Ц.25), де в першому рядку визначаються кодування символів файлу, в другому – знаходиться документація об'єкта (значення атрибута `__doc__`), в третьому – присвоюється значення атрибута `isCause`, що вказує на те, що фактор є причиною концентрації напружень. Документація може містити код мовою розмітки Markdown, а код класів і агентів може перетворюватись у інтерактивні документи Jupyter Notebook [369].

Клас `Property` (лістинг Ц.4) описує множину агентів-властивостей, які використовуються для створення відношень між агентами онтології. Властивості мають атрибути подібні до атрибутів властивостей мови OWL: `inverseOf` – назва інверсної властивості, `domain` – множина класів, які володіють цією властивістю, `range` – множина класів, об'єкти, яких можуть входити в множину значень властивості, `FunctionalProperty` – властивість має тільки одне значення, якщо `FunctionalProperty=True`, та ін. Для прикладу атрибут

`isCause` класу `Factor` є властивістю, якщо існує агент `isCause`, описаний у модулі `Property_isCause.py` (лістинг Ц.34).

Властивості також володіють методом `datalogRules`, що повертає множину `pyDatalog`-правил для логічного виведення. Наприклад для властивості `isCause` наступне правило дозволяє за допомогою машини логічного виведення `pyDatalog` виводити нові факти `isEffect(X,Y)` за відомими фактами `isCause(Y,X)`:

$$\text{"isEffect}(X,Y) \leftarrow \text{isCause}(Y,X) \text{"} \quad (6.4)$$

де `X`, `Y` – назви об'єктів факторів.

Клас `Datalog` (лістинг Ц.5) призначений для створення агентів для логічного виведення за допомогою `pyDatalog`. Його метод `rule` спочатку за допомогою `KBToFacts` отримує множину усіх фактів з усіх значень властивостей усіх агентів у вигляді триплетів

$$(\text{subject}, \text{predicate}, \text{object}) \quad (6.5)$$

де `subject` – назва агента, `predicate` – назва його властивості, `object` – елемент множини значень властивості. Потім методом `getDatalogFacts` множина фактів перетворюється у список `pyDatalog`-фактів з елементами `"predicate ('subject', 'object')"`. Методом `getDatalogRules` формується множина `pyDatalog`-правил з методів `datalogRules` усіх агентів. Логічне виведення нових фактів за допомогою `pyDatalog 0.17.1` виконується у функції `runDatalog`, якій передаються списки усіх фактів, правил і предикатів. Ця функція завантажує `pyDatalog`-код функцією `load` і виводить нові факти шляхом запитів до бази знань функцією `ask` для кожного предикату. Нові факти повертаються у вигляді множини триплетів. Після кожного виклику `runDatalog` база очищується функцією `clear`. Після логічного виведення ці факти повертаються у базу знань KB функцією `factsToKB` шляхом додання `object` у множину значень властивості `predicate` агента `subject`. Метод `rule` застосовується, поки різниця множин нових фактів і старих фактів (до

застосування `rule`) не пуста. Слід відмітити, що логічне виведення може бути реалізоване і іншими методами. Наприклад можна легко реалізувати наївний алгоритм прямого виведення, що відповідає правилу (6.4), шляхом додання в метод `rule` класу `Factor` коду (лістинг Ц.9, рядки 18-24). Ви можете протестувати цей алгоритм шляхом виключення агентів `Datalog` (`active=False`). Але застосування `pyDatalog` більш продуктивне. Можна також використовувати інші сторонні машини виведення, в тому числі паралельно. Наприклад можна задіяти розроблений автором інтерфейс з `Datalog Educational System` для мови програмування `Python` [370].

Клас `Fact` (лістинг Ц.8) описує факт бази знань у вигляді триплету (6.5). Атрибут `source` – це джерело факту. Його метод `rule` дозволяє автоматично створювати значення властивостей агентів, що відповідають цьому факту. Приклад агента – лістинг Ц.17.

Клас `Model` (лістинг Ц.10) описує агент СЕМ РЗ штанг за ГОСТ 13877-96. Атрибути `paramsIn` та `paramsOut` містять словники з назвами та значеннями вхідних і вихідних параметрів моделі. Метод `rule` створює новий процес для симуляції моделі, передає йому через командний рядок значення `paramsIn`, чекає його завершення і читає результати з його стандартного потоку виведення у `paramsOut` за допомогою методу `parseOutput`. Якщо `paramsIn` пустий, то буде симулюватись модель із стандартними значеннями параметрів відповідно ГОСТ 13877-96. Модель буде симулюватись тільки тоді, коли серед значень параметрів `paramsOut` є `None`. Приклад агента – лістинг Ц.27. Симуляція відбувається за допомогою розробленої автором програм `ThreadsOCC` (додаток I).

Клас `Parameter` (лістинг Ц.11) описує агенти-параметри РЗ. Об'єкти `Parameter` володіють атрибутами – факторами `factorHigh` та `factorLow`, які дозволяють пов'язати параметри з факторами. Приклад агента – лістинг Ц.31.

Клас `Dependence` (лістинг Ц.6) описує агенти з регресійними залежностями однієї змінної. Атрибут `Xpar` – це назва незалежної змінної, атрибут `Ypar` – назва залежної змінної. Значення `Xpar`, `Ypar` є ключами агентів класу `Parameter`. Атрибути `X`, `Y` містять список значень незалежної та залежної

змінних. Атрибут `source` вказує на джерело залежності (література або модель). Приклад агента – лістинг Ц.13. Дані X , Y можуть бути автоматично отримані з множини моделей (агентів `Model`) за допомогою методу `fromModels`, якщо джерелом є модель і елементи цієї множини відрізняється від моделі тільки значенням одного параметра з `Xpar` (це перевіряється методом `isSibling` класу `Model`). Агент `Dependence` також може динамічно створювати моделі (агенти класу `Model`), якщо є значення X , але немає деяких значень Y (тобто вони `None`) і така модель не існує. На наступних кроках ітерації ці моделі будуть обчислені та дані Y будуть отримані методом `fromModels`. Метод `rule` викликає методи `fromModels`, `toModels` та будує лінійну регресію за допомогою `SciPy`. Якщо значення коефіцієнта детермінації R^2 достатньо високе ($R^2 > 0,5$) то `rule` намагається створити новий факт (лістинг Ц.6, рядки 77-80). Таким чином висока кореляція `Xpar` і `Ypar` дозволяє створювати причинно-наслідкові зв'язки між факторами.

Клас `DependenceMulti` (лістинг Ц.7) подібний на клас `Dependence` і описує регресійні залежності багатьох змінних. Атрибути `Xpars` і `Ypars` містять списки назв залежних і незалежних параметрів. Атрибути `X` і `Y` містять їхні значення. Приклад агента – лістинг Ц.16. Для обробки і аналізу даних X і Y можна використовувати пакет `pandas` і його структуру даних `DataFrame`. Для двосторонньої конвертації у `DataFrame` використовуються методи `toDataFrame` та `fromDataFrame`. Метод `linregress` використовується для побудови лінійної регресії за допомогою пакету `scikit-learn` і повертає її коефіцієнти та коефіцієнт детермінації. Якщо джерело є моделлю, а деякі елементи Y невідомі (`None`), то метод `byModels` дозволяє автоматично отримати значення Y шляхом створення тимчасових моделей та їхньої симуляції. Метод `optimize` дозволяє знаходити оптимальні значення X за критерієм `yp` шляхом симуляції моделей і застосування алгоритмів оптимізації функції багатьох змінних зі `SciPy`. `optimize` оптимізує значення, що повертає метод `simModel(x,yp)`, де x – вектор аргументів, `yp` –

назва залежного параметра. Наступний приклад показує застосування методу `optimize`:

```
KB[dependence4].optimize(yp="коефіцієнт запасу втомної міцності",
    bounds=[(2.5,3.5),(12.7, 13.26)], maximize=True)
```

На відміну від `Dependence`, автоматизоване створення причинно-наслідкових зв'язків між факторами за кореляцією `Xpars` та `Ypars` не реалізоване. Це повинен робити сам користувач.

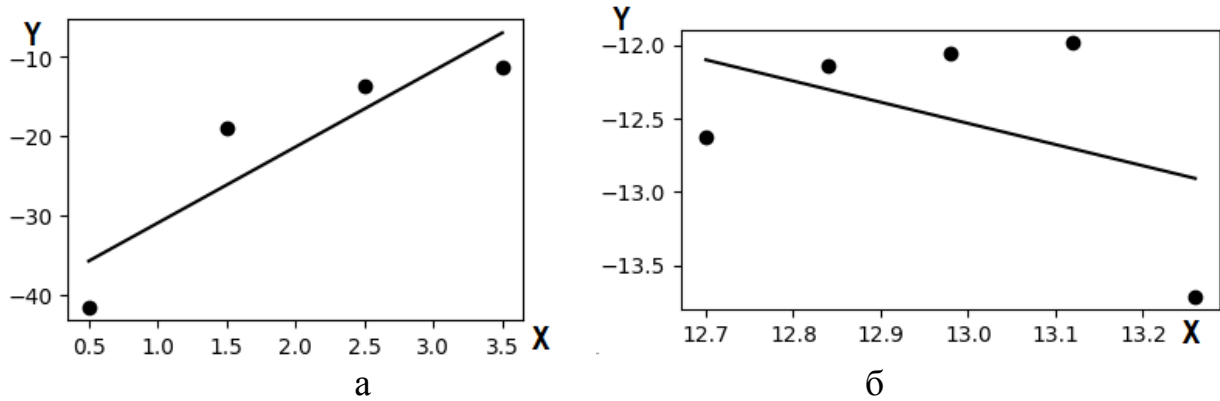
Система підтримує інтерактивну роботу в командних оболонках Python або IPython. Для спрощення редагування коду автори рекомендують застосовувати `python-IDE` з інтегрованою IPython-оболонкою. Для полегшення роботи з системою (створення та редагування агентів, створення запитів і візуалізації) можуть бути застосовані додаткові модулі з GUI (рис. 6.6). Нижче описано приклад роботи з системою в командній оболонці (див. також лістинг Ц.1):

```
>>> from tools import * # імпортувати з модуля все
>>> files=getFiles() # отримати список модулів з поточного каталогу
>>> createClasses(files) # створити класи агентів
>>> createKB(files) # створити агенти
>>> applyRules() # застосовувати правила поки є зміни
>>> saveKB() # зберегти агенти
>>> loadKB() # завантажити агенти
```

Наступний запит виводить усі причини фактора "концентрація напружень":

```
>>> for v in KB["концентрація напружень"].isEffect:
...     print v
корозійне пошкодження
корозійна язва
```

Наступний запит у функції `query2` (лістинг Ц.2) будує графік залежності "dependence2" та її лінійну регресію (рис. 6.7) за допомогою `Matplotlib`, а також виводить її параметри. Залежності на рис. 6.7 були отримані автоматично з агентів `Model`. За залежністю рис. 6.7а також автоматично виведено факт ("збільшення радіуса скруглень зарізьбової канавки", "isCause", "збільшення значення D"). Такого виведення за залежністю рис. 6.7б зроблено не було з зв'язку з малим значенням R^2 .



а) – від радіуса скруглень зарізьбової канавки, мм ($R^2=0,8$);

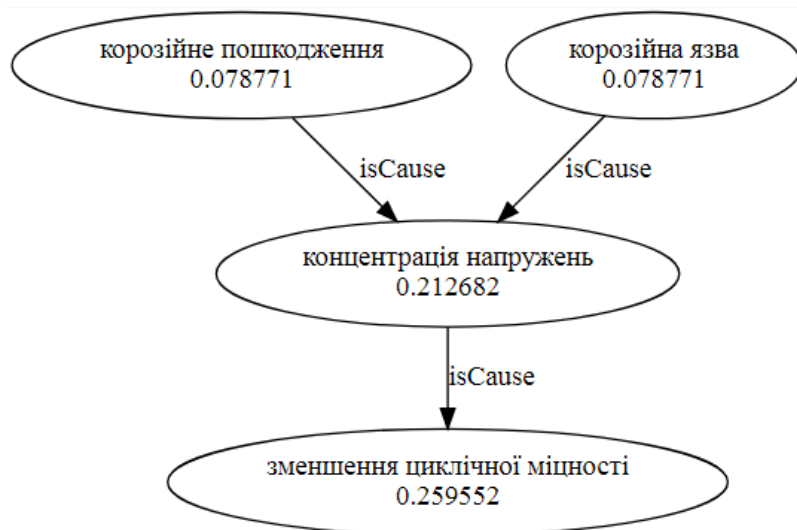
б) – від зовнішнього радіуса різьби ніпеля ($R^2=0,19$)

Рисунок 6.7- Залежності D (вісь Y) від геометричних параметрів (вісь X)

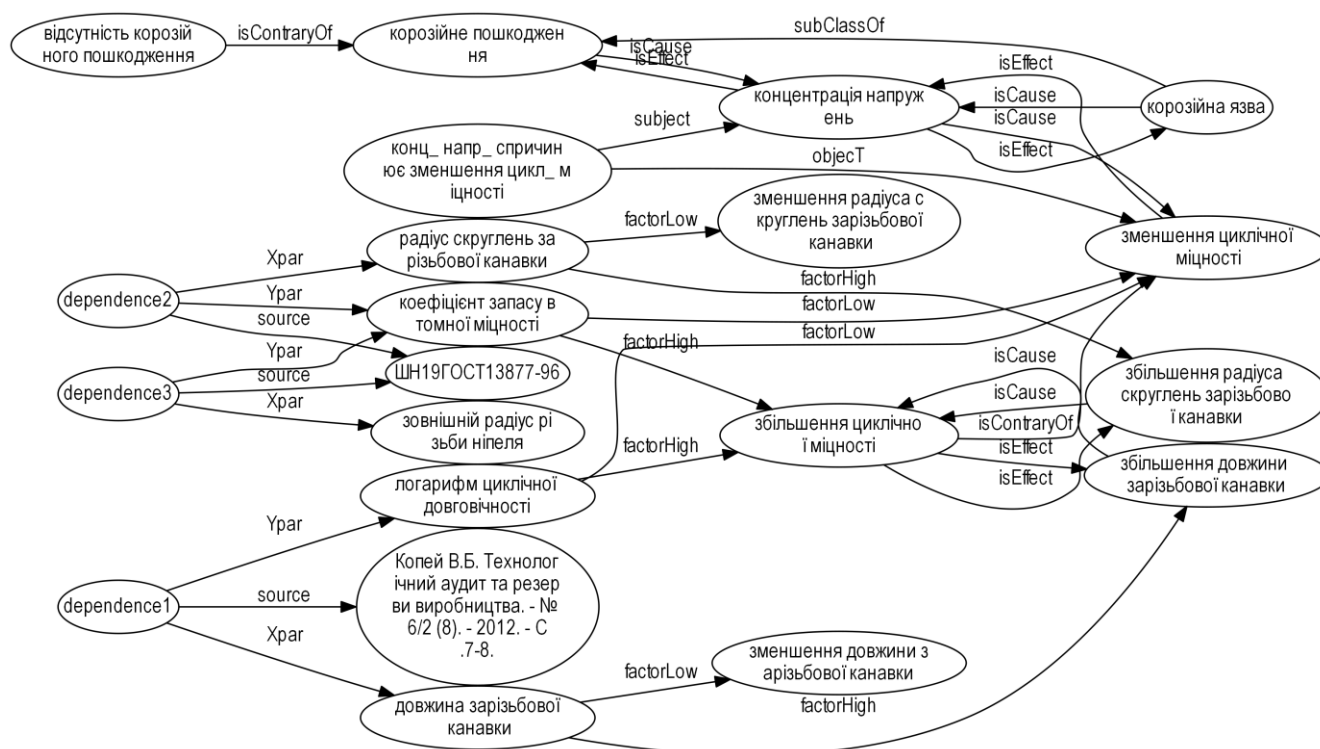
Запит у функції `query3` (лістинг Ц.2) будує орієнтований граф (рис. 6.8а) за допомогою пакету `NetworkX` з ребрами "isCause" і вершинами класу `Factor` та розраховує для них значення PageRank [371] на основі структури вхідних ребер. Ці значення PageRank можуть бути використані для оцінки важливості фактора – відносної кількості факторів, що спричиняють цей фактор. Граф зі значеннями PageRank візуалізується за допомогою `Graphviz` шляхом його конвертації у код DOT (мова опису графів) [372].

На рис. 6.8б показаний граф, що є результатом запиту з функції `query4` (лістинг Ц.2) до KB і містить вершини – агенти класів `Factor`, `Fact`, `Parameter`, `Dependence` та ребра – властивості `isEffect`, `subClassOf`, `isContraryOf`, `object`, `subject`, `factorHigh`, `factorLow`, `Xpar`, `Ypar`, `source`.

Тут продемонстровані приклади роботи з системою на стадії проектування РЗ, але система також може бути доповнена агентами для будь-яких стадій ЖЦ будь-якого обладнання ШСНУ. В систему можуть бути інтегровані моделі, залежності, факти та інші інформаційні ресурси, отримані в цій роботі. Також є можливість інтегрувати модель для геометричної симуляції нарізання різьб різцем [8, 10, 13, 197, 373], компоненти для розмірного аналізу конструкцій [374], прикладні САПР [18, 183, 193, 375], компоненти для розрахунку норм часу на технологічну операцію механічної обробки деталей [376], компоненти для автоматизованого добування знань з онлайн-баз патентів [377]. Глибоко не розглядалися питання продуктивності або побудови розподіленої системи, але це можна реалізувати в майбутньому за допомогою відомих методів і Python-пакетів [276-278].



а



б

а) – агенти класу **Factor** зі значеннями PageRank (показана частина графа);

б) – агенти класів **Factor**, **Fact**, **Parameter**, **Dependence**

Рисунок 6.8 – Візуалізація запитів до бази знань у вигляді орієнтованих графів

6.4 Висновки до розділу

Розроблено принципи побудови інформаційної системи для проектування і підтримки життєвого циклу ШСНУ, що базується на множині моделей обладнання ШСНУ, його відмов, базі знань з проблем надійності та мові Python. Компоненти ІС доступні в додатках та в GitHub (vkorey/PU. URL: <http://github.com/vkorey/PU>) як вільне програмне забезпечення.

1. На основі принципу ізоморфізму закономірностей складних систем та міждисциплінарного аналізу цих закономірностей у різних природних та штучних системах розроблено абстрактну модель інформаційної системи підтримки ЖЦ ШСНУ [213]. Модель відображає тільки найбільш загальні її характеристики, шляхом дихотомічного ділення ЖЦ виробу виділяє класи функціональних елементів ІС та їхню ієрархію, а також володіє основними закономірностями складних систем. Клас елемента ІС описує множину однотипних спеціалізованих елементів ІС, які призначені для досягнення певних цілей ІС та можуть взаємодіяти з іншими елементами для досягнення заданого рівня синергії. Наведено орієнтовні змістові інтерпретації класів. Ця модель є якісною та може бути основою для розробки більш складних кількісних математичних моделей ІС. Розроблено ізоморфні до абстрактної моделі динамічні математичні моделі, які описуються системою диференціальних рівнянь, зокрема модель "гармонічний осцилятор" з чотирма змінними. Виконано аналіз бінарних відношень між класами моделі та виділено типи відношень. Запропоновано способи розрахунку рівня синергії підсистем моделі. Модель є ізоморфною до структур ціленаправленої діяльності в різних складних ІС і має фрактальну структуру, тобто може описувати не тільки ІС підтримки всього ЖЦ виробу, але й будь-якого його етапу: проектування, виробництва, експлуатації. Також її можна використовувати як методику системного аналізу. Подано приклад класифікації розроблених в роботі елементів ІС підтримки ЖЦ ШСНУ та визначення класів

їхніх бінарних відношень з метою обчислення рівня синергії. Запропоновано алгоритм функціонування ІС який включає механізми ізоморфні до механізмів складних систем [357].

2. Запропоновані принципи побудови експертних систем з проблем надійності обладнання ШСНУ, які описують фактори, що впливають на надійність, та дозволяють логічне виведення нових знань, у тому числі шляхом симуляції моделей. Підхід оснований на семантичних мережах, логіці предикатів, мові OWL [29, 362] та використанні об'єктно-орієнтованих конструктів мови Python для створення класів, індивідів, атрибутів і відношень онтології [30, 363, 364]. Основні його переваги – універсальність і широкі можливості мови Python, легке розширення функціональності системи без необхідності винаходити ще одну спеціальну мову подання знань чи запитів до них, проста інтеграція в інші інформаційні системи.

3. Запропоновані принципи розроблення ІС, що створюють передумови до появи в них властивостей складних систем і, таким чином, підвищують їхню ефективність [32, 378]. Наведено приклад реалізації ІС РЗ ШСНУ, яка:

- направлена на просту і ефективну інтеграцію багатьох різнотипних компонентів, основана на агентно-орієнтованому підході та належить до класу мультиагентних та гібридних інтелектуальних систем;

- орієнтована на зручний і простий опис агентів, їхньої поведінки і взаємодії, динамічне створення та персистентність агентів, розширення функціональності системи шляхом створення нових класів і агентів.

- розроблена популярною мовою Python, дозволяє роботу з системою в оболонці Python, використовує різні вільні Python-пакети і стороннє програмне забезпечення, зокрема бібліотеку pyDatalog та розроблену автором програму для скінченно-елементної симуляції РЗ.

ЗАГАЛЬНІ ВИСНОВКИ

У дисертації наведено нове вирішення наукової проблеми, яка полягає в підвищенні ефективності проектування обладнання комплексно-працевдатної ШСНУ. Ця проблема вирішена шляхом розвитку методології автоматизованого проектування обладнання ШСНУ, що опирається на принципи теорії систем і використовує основу на Python інформаційну систему з гетерогенними елементами (імітаційні моделі обладнання, статистичні моделі відмов, результати симуляції, база знань та інші), та методів комп'ютерного моделювання обладнання ШСНУ, які завдяки системному підходу спрямовані на підвищення адекватності, спрощення використання, розвитку та інтеграції моделей, забезпечення можливості системного дослідження працевдатності та оптимізації параметрів обладнання. Запропоновані рішення призначені для вирішення множин проблем працевдатності ШСНУ для різних умов експлуатації з найменшими витратами. Зокрема отримано наступні результати.

1. На базі індуктивних методів самоорганізації моделей розроблено принципи побудови точних і робастних статистичних моделей відмов колон ШН і полірованого штока: моделей густини імовірності відмов як функції від відносної глибини обриву та моделей класу аварійності колони, які враховують множину факторів, пов'язаних з параметрами ШСНУ та її відмови. Обґрунтовано ефективність застосування методів ансамблів дерев рішень для прогнозування частоти відмов колон ШН (правильність моделей досягає 0,94). Побудовано відповідні моделі, що дозволяють ефективно ідентифікувати високоаварійні колони та їхні інтервали ще на етапі проектування, виявляти причини відмов та приймати рішення щодо забезпечення їхньої працевдатності.

2. Розроблено принципи компонентно-орієнтованого моделювання ШСНУ, спрямованого на моделювання факторів, які розподілені нерівномірно вздовж колони ШН, спрощення створення моделей, їхнього розвитку, використання більшою спільнотою користувачів та інтеграції в ІС. Зокрема розроблено параметричні моделі ШСНУ – модель на основі абстрактних автоматів та мови

Python з можливістю моделювання явищ, які важко сформулювати за допомогою диференціальних рівнянь; модель мовою Modelica, яка для моделювання колони ШН використовує стандартні поступальні механічні компоненти. Розроблено алгоритмічні основи та реалізацію програмного каркасу мовою Python для створення компонентно-орієнтованих, подібних на Modelica-моделі, акаузальних моделей ШСНУ та інших динамічних систем без необхідності застосування спеціалізованих мов моделювання. Адекватність моделей підтверджено шляхом порівняння результатів симуляції з практичними динамограмами. Показані приклади застосування моделей для аналізу склопластикових колон, білярезонансних частот ходів і аварійних ситуацій.

3. Шляхом використання відомих САПР і систем моделювання гідродинаміки і механіки деформівного твердого тіла та розроблених автором програмних компонентів запропоновані принципи побудови та застосування параметричних геометричних моделей, СЕМ та прикладних САПР проблемних деталей та вузлів ШСНУ, які, як показано в роботі, володіють здатністю до простої інтеграції в ІС та використовуються для різностороннього аналізу і оптимізації під час дослідження працездатності та проектування. Зокрема розроблено: гідродинамічні СЕМ кульового зворотного клапана свердловинного штангового насоса та протектора ШН; геометричні моделі та СЕМ РЗ – муфтового РЗ ШН, замкового РЗ та двоопорного РЗ порожнистих ШН, муфтового РЗ НКТ, модель РЗ з довільними геометричними відхиленнями; СЕМ пресового з'єднання тіла склопластикової ШН зі сталеву головою; геометричні моделі та СЕМ ШН і НКТ з корозійними та втомними дефектами і БС.

4. На основі моделей муфтового РЗ ШН отримано залежності для визначення оптимальної величини згвинчування та довжини зарізьбової канавки за критерієм втомної міцності; з метою оптимізації конструкції отримано залежності напружень в небезпечних зонах від міцності матеріалів та геометричних параметрів; шляхом комплексного удосконалення конструкції зменшено еквівалентні напруження в небезпечних зонах на 10...20%; виявлені залежності напружень, контактних тисків та коефіцієнта запасу втомної міцності в різьбі від відхилень максимального діаметра різьби ніпеля, кута профілю та

кроку різьби муфти з метою обґрунтування їхніх допустимих значень. За допомогою удосконаленої ітеративної методики проектування обґрунтовано доцільність застосування та розроблено конструкцію двоопорного РЗ для порожнистих ШН, яке володіє більшою міцністю під час згину, кручення та стиску, вищим опором до самовідгвинчування та герметичністю, більш рівномірним розподілом навантаження на витки та більшою втомною міцністю у порівнянні зі стандартним.

За допомогою моделей муфтового РЗ НКТ отримано залежності для визначення оптимальної величини натягу НКТ за критерієм герметичності. Для з'ясування працездатності РЗ в умовах високочастотних вібрацій запропоновано методику аналізу відклику РЗ на гармонічне зовнішнє осьове навантаження частотами 0..20 кГц; отримано залежності контактного тиску на робочій стороні профілю від частоти та залежність резонансної частоти від коефіцієнта тертя. Виявлено можливість порушення герметичності та втомного руйнування РЗ внаслідок високочастотного резонансу. За критеріями герметичності та втомної міцності обґрунтовано можливість застосування різьб ніпелів НКТ з відхиленнями кута профілю внаслідок нарізання некоригованим різцем з від'ємним переднім кутом. Розроблено методики побудови вказаних залежностей.

5. Моделі пресових з'єднань склопластикових ШН використано для отримання розподілів напружень і контактних тисків в з'єднанні; запропоновано методику та залежності для вибору оптимальних параметрів з'єднання – глибини переміщення штампів, границі плинності сталі, довжини обтискання. Проаналізовано відклик з'єднання на гармонічне зовнішнє осьове навантаження та запропоновано методику виявлення небезпечних діапазонів частот з точки зору міцності з'єднання та втомної міцності ніпеля. Рекомендовано збільшення радіусу скруглення між піделеваторним буртом і тілом ШН в умовах додаткових навантажень згину.

Шляхом використання моделей ШН та НКТ з дефектами та БС отримані регресійні залежності еквівалентних напружень від параметрів дефекту і БС, а також розроблено методику побудови та побудовані криві втоми ШН та НКТ з втомними тріщинами та БС, які підтверджують доцільність ремонту

бандажуванням. Аналітичний, чисельний та експериментальний методи обчислення напружень в трубі з БС, товщина якого рівна товщині труби, показали зменшення кільцевих напружень в 1,18-1,54 рази, а осьових – в 1,05-1,21 рази. Для зменшення концентрації напружень запропоновано методику оптимізації конструкції фаски БС.

За допомогою моделей протектора ШН запропоновано принципи оптимізації його параметрів за критеріями мінімального гідродинамічного опору, максимальної площі тертя та максимального крутного моменту.

6. Запропоновані принципи побудови та розроблено експертну систему з проблем надійності РЗ ШСНУ, база знань якої описує фактори, що впливають на надійність, та дозволяє логічне виведення нових знань, у тому числі шляхом симуляції моделей. Підхід оснований на семантичних мережах, логіці предикатів, мові OWL та використанні об'єктно-орієнтованих конструктів мови Python для створення класів, індивідів, атрибутів і відношень онтології. Основні його переваги – проста інтеграція в ІС, універсальність і широкі можливості мови Python, легке розширення функціональності системи, зокрема можливість створення власних способів подання і виведення знань.

7. На основі міждисциплінарного аналізу закономірностей складних систем розроблено абстрактну модель інформаційної системи підтримки ЖЦ обладнання ШСНУ, яка шляхом дихотомічного ділення виділяє класи компонентів ІС (моделі, результати симуляції, факти бази знань та ін.) та їхню ієрархію, а також володіє основними закономірностями складних систем, що робить її ефективнішою. На основі цієї моделі розвинуто методологічні основи автоматизованого проектування обладнання та інформаційної підтримки життєвого циклу ШСНУ і запропоновані принципи реалізації ІС, яка належить до класу мультиагентних систем, володіє закономірностями складних систем, зокрема здатністю до простого розширення та інтеграції гетерогенних компонентів. Впроваджена ІС використовується в НГВУ "Долина нафтогаз" для підвищення якості експлуатації ШСНУ та забезпечення їхньої надійності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Копей В. Б. Підвищення ресурсу штангової колони при видобутку парафіністих нафт : дис. ... канд. техн. наук : 05.05.12 : захищена 16.11.04: затв. 09.03.05. Івано-Франківськ, 2004. 175 с.
2. Копей В. В., Зінченко Ю. С., Копей В. Б. Аналіз поломок насосних штанг в промислових умовах // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2008. № 2(18). С. 49-56.
3. Experimental study of the reinforcement of damaged steel pipe by composite bandage / Kopey V., Rozgonjuk V., Kopey V., Maksymuk O., Scherbina N., Nayda A. // Wiertnictwo Nafta Gaz. 2004. r.21/1. P. 125-134.
4. Оптимізація товщини композитних бандажів при ремонті трубопроводів з дефектами / Копей В. В., Копей В. Б., Максимук О. В., Щербина Н. М., Найда А. М. // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2007. № 2(16). С. 101-107.
5. Копей В. Б., Петрина Ю. Д., Венгрынюк Т. П. Конечно-элементное моделирование ремонта труб с дефектами стеклопластиковыми бандажами в SolidWorks® // Надежность и безопасность магистрального трубопроводного транспорта : материалы VII Междунар. науч.-техн. конф., Новополоцк, 22 – 25 ноября 2011 г. / под общ. ред. д-ра техн. наук, проф. В. К. Липского. Новополоцк : Полоц. гос. ун-т, 2011. С. 248-250.
6. Сучасні методи боротьби з корозією глибинного обладнання ШСНУ / Копей В. В., Онищук О. О., Онищук С. Ю., Копей В. Б. // Нафтогазова енергетика. 2008. № 2(7). С. 13-16.
7. Finite-element analysis of the tubing thread / Kopey V., Kopey V., Bebnarz S., Savula S. // Wiertnictwo Nafta Gaz. 2006. r.23/2. P. 681-685.
8. Kopei V. B., Onysko O. R., Panchuk V. G. Computerized system based on FreeCAD for geometric simulation of the oil and gas equipment thread turning // IOP Conf. Ser.: Mater. Sci. Eng. 2019. 477:012032. DOI: 10.1088/1757-899X/477/1/012032.

9. Копей В. Б., Онисько О. Р., Жигуц Ю. Ю. Обґрунтування застосування двоопорних різьбових з'єднань пустотілих насосних штанг // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2019. №1(46). С. 7-15. DOI: 10.31471/1993-9965-2019-1(46)-7-15.

10. Kopei V., Onysko O., Panchuk V. The application of the uncorrected tool with a negative rake angle for tapered thread turning // Ivanov V. et al. (eds) Advances in Design, Simulation and Manufacturing II. DSMIE 2019. Lecture Notes in Mechanical Engineering. Cham : Springer, 2020. P. 149-158. DOI: 10.1007/978-3-030-22365-6_15.

11. Onysko O. R., Kopei V. B., Panchuk V. G. Theoretical investigation of the tapered thread joint surface contact pressure in the dependence on the profile and the geometric parameters of the threading turning tool // IOP Conf. Ser.: Mater. Sci. Eng. 2020. 749:012007. DOI: 10.1088/1757-899X/749/1/012007.

12. Копей В. Б., Панчук А. Г. Дослідження залежності напружень в муфтовому різьбовому з'єднанні насосних штанг від характеристик матеріалів деталей з'єднання // Сучасні технології в промисловому виробництві : матеріали Всеукраїнської міжвузівської науково-технічної конференції, м. Суми, 19-23 квітня 2010 р. Ч. II. Суми : Вид-во СумДУ, 2010. С. 122-123.

13. Kopei V., Onysko O., Panchuk V. Computerized system based on FreeCAD for geometric simulation of thread turning of oil and gas pipes // 6th International Conference of Applied Science : book of abstracts, May 9-11, 2018, Banja Luka, Bosnia and Herzegovina. Banja Luka : University of Banja Luka, 2018. P. 108.

14. Копей В. Б., Онисько О. Р., Жигуц Ю. Ю. Обґрунтування застосування двоопорних муфтових різьбових з'єднань пустотілих насосних штанг // Матеріали доповідей VIII Міжнародної науково-технічної конференції "Прогресивні технології у машинобудуванні РТМЕ-2019", 4-8 лютого 2019 р., м. Івано-Франківськ-Яремче. Івано-Франківськ : ІФНТУНГ, 2019. С. 146-149.

15. Kopei V. B., Kopei B. V. Harmonic axial loading analysis of the tubing threaded connection // Modern achievements of science and education : proceedings of

XIV International scientific conference, September 26 - October 3, 2019, Netanya, Israel. Khmelnytskyi : KhNU, 2019. P. 29-37.

16. Копей В. Б., Панчук А. Г. Оптимізація параметрів з'єднання тіла склопластикової насосної штанги зі сталеву головою // Инновационные технологии в машиностроении : материалы Международной научно-практической конференции, г. Запорожье, 17-21 мая 2011 г. Том 2. Запорожье : Запорожская торгово-промышленная палата, 2011. С. 58-60.

17. Копей В. Б., Венгрынюк Т. П. Моделирование дефектов труб в SolidWorks® // Надежность и безопасность магистрального трубопроводного транспорта : материалы VII Междунар. науч.-техн. конф., Новополоцк, 22 – 25 ноября 2011 г. / под общ. ред. д-ра техн. наук, проф. В. К. Липского. Новополоцк : Полоц. гос. ун-т, 2011. С. 250-251.

18. Копей В. Б., Воробець М. І. Система автоматизованого проектування металополімерних з'єднань на основі вільного програмного забезпечення // Машинобудування очима молодих: прогресивні ідеї – наука – виробництво (МOM – 2017) : матеріали тез доповідей XVII Міжнародної науково-практичної конференції, м. Чернігів, 01 – 03 листопада 2017 р. Чернігів : ЧНТУ, 2017. С. 31-33.

19. Копей В. В., Михайлюк В. В., Копей В. Б. Моделирование резьб насосных штанг методом конечных элементов // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2009. № 2(20). С. 61-67.

20. Скінчено-елементний аналіз насосних штанг з зарізьбовими канавками / Копей В. В., Копей В. Б., Петрина Ю. Д, Михайлюк В. В. // Анотації Міжнародної науково-технічної конференції "Нафтогазова енергетика: проблеми і перспективи". м. Івано-Франківськ, 20-23 жовтня 2009 р. Івано-Франківськ : ІФНТУНГ, 2009. С. 67.

21. Копей В. В., Кузьмін О. О., Копей В. Б. Механічні методи зняття відкладень парафіну та асфальто-смолистих речовин з поверхні свердловинного обладнання // Нафтогазова енергетика. 2008. № 3(8). С. 10-14.

22. Копей Б. В., Кузьмін О. О., Копей В. Б. Розроблення з'єднань склопластикових порожнистих насосних штанг та визначення навантажень на них // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2010. № 1(23). С. 77-83.

23. Використання явища резонансу для комплектування колони насосних штанг / Олійник А. П., Копей Б. В., Зінченко Ю. С., Копей В. Б. // Розвідка та розробка нафтових і газових родовищ. 2011. №1 (38). С. 69-75.

24. Копей В. Б., Копей Б. В., Кузьмін О. О. Принципи побудови моделі свердловинної штангової насосної установки для середовища Maplesoft MapleSim 7 // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2017. №2(43). С. 42-52.

25. Kopei V. B., Onysko O. R., Panchuk V. G. Component-oriented acausal modeling of the dynamical systems in Python language on the example of the model of the sucker rod string // PeerJ Computer Science. 2019. 5:e227. DOI: 10.7717/peerj-cs.227.

26. Копей В. Б., Копей Б. В., Кузьмін О. О. Принципи побудови моделі свердловинної штангової насосної установки для середовища Maplesoft MapleSim 7 // Тези доповідей Міжнародної науково-технічної конференції "Нафтогазова енергетика-2017", м. Івано-Франківськ, 15-19 травня 2017 р. Івано-Франківськ : ІФНТУНГ, 2017. С. 364-365.

27. Kopei V. B., Onysko O. R., Panchuk V. G. Component-oriented acausal modeling of the dynamical systems in Python language on the example of the model of the sucker rod string // PeerJ Preprints. 2019. 7:e27612v1. DOI: 10.7287/peerj.preprints.27612v1.

28. Kopei V., Panchuk V., Onysko O. Component-oriented modelling of dynamical systems in Python language on the example of the model of the sucker rod string // International conference Innovative Ideas in Science 2017 : book of abstracts, Banja Luka, 02 - 03 November 2017. Banja Luka : University of Business Engineering and Management, 2017. P. 33.

29. Копей В. Б., Петрина Ю. Д. Принципи розробки бази знань з проблем надійності і довговічності різьбових з'єднань // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2010. № 4(26). С. 66-69.

30. Копей В. Б., Палійчук І. І. Застосування мови програмування Python для побудови баз знань з проблем надійності і довговічності штангових свердловинних насосних установок // Нафтогазова енергетика. 2011. №2(15). С. 12-18.

31. Kopei V. B., Onysko O. R., Panchuk V. G. Principles of development of product lifecycle management system for threaded connections based on the Python programming language // J. Phys.: Conf. Ser. 2020. 1426:012033. DOI: 10.1088/1742-6596/1426/1/012033.

32. Kopei V., Onysko O., Panchuk V. Principles of the product lifecycle management system development for threaded connections based on the Python programming language // International Conference of Applied Science ICAS 2019 : book of abstracts, May 9-11, 2019, Hunedoara, Romania. Timisoara : Editura EUROBIT, 2019. P. 42.

33. Зубаиров С. Г. Разработка техники добычи высоковязких нефтей штанговыми насосами при кустовом размещении скважин : дис. ... д.т.н. : 05.04.07. 2000, Уфа. 330 с.

34. Копей В. В., Копей В. Б. Аналіз конструкцій скребків і протекторів для насосних штанг // Розвідка та розробка нафтових і газових родовищ. Серія: Нафтогазопромислове обладнання. 2001. №38(том 4). С. 42-52.

35. Копей В. Б. Розробка и аналіз конструкцій храпових штангообертачів // Розвідка та розробка нафтових і газових родовищ. 2003. № 1(6). С. 37-40.

36. Копей В. В., Копей І. Б. Надійність свердловинних насосів різного діаметра // Розвідка та розробка нафтових і газових родовищ. Серія: Нафтопромислова механіка. 1995. вип.32. С. 49-56.

37. Копей В. Б., Стеліга І. І. Аналіз відмов колон насосних штанг в НГВУ “Долина нафтогаз” // Розвідка та розробка нафтових і газових родовищ. 2002. № 4(5). С. 78-80.

38. Копей В. В., Кіндрачук С. М. Аналіз надійності НКТ в свердловинах з обводненою продукцією. Івано-Франківськ : ІФДТУНГ, 1995. 11 с. Деп. в ДНТБ України 25.11.95. №2492. Ук95.

39. Clayton T. Hendricks, Russell D. Stevens. Sucker rod failure analysis. Special report from Norris. Tulsa, 2005. 15 p.

40. Протасов В. Н. Повышение надежности оборудования скважин при насосном способе добычи нефти // Обзорная информация. Серия «Машины и нефтяное оборудование». Москва : ВНИИОЭНГ, 1986. №4(62). 68 с.

41. Круман В. В., Крупицына В. А. Коррозионно-механический износ оборудования. Москва : Машиностроение, 1968. 104 с.

42. Круман В. В. Практика эксплуатации и исследования глубиннонасосных скважин. Москва : Недра, 1964. 203 с.

43. Круман В.В. Глубиннонасосные штанги. Москва : Недра, 1977. 181 с.

44. Фаерман И. Л. Штанги для глубинных насосов. Баку : Азнефтеиздат, 1955. 323 с.

45. Копей В. В. Науково-технологічні принципи комплексного підвищення ресурсу свердловинного нафтогазового обладнання : дис. ... докт. техн. наук : 05.15.07. Івано-Франківськ, 1996. 478 с.

46. Копей В. В., Тараевский С. И. Защитные покрытия для глубиннонасосных штанг. Обзор. информация. Сер. «Коррозия и защита в нефтегазовой промышленности». Москва : ВНИИОЭНГ, 1982. 36 с.

47. Протасов В. Н., Шрейдер А. В., Бикчентаев Р. М. Исследование сероводородной коррозии стали под полимерными покрытиями // Коррозия и защита в нефтегазовой промышленности. 1977. №11. С. 19-21.

48. Протасов В. Н. Состояние и перспективы применения противокоррозионной защиты насосных штанг полимерными покрытиями //

Нефт. пром. Сер. «Коррозия и защита в нефтегазовой промышленности». Вып. 7(25). Москва : ВНИИОЭНГ, 1983. 44 с.

49. Досвід експлуатації насосно-компресорних труб та насосних штанг з дифузійним цинковим покриттям / Проскуркін Є. Н., Норвілло Н. Ю., Сухомлин А. І., Гнатюк А. М. // Нафтова і газова промисловість. 1997. №3. С. 28-31.

50. Повышение штанг алюминированием / Саакян Л. С., Соболева И. А., Толкачёв Ю. И., Пастухов И. В. // РНТС. ВНИИОЭНГ. Сер. «Коррозия и защита в нефтегазовой промышленности». 1974. №11. С. 19-21.

51. Исследование защитных покрытий глубиннонасосных штанг / Круман Б. Б. и др. // В сб.: Совершенствование систем разработки и эксплуатации нефтяных и газовых месторождений Нижнево Поволжья. Труды Волгоград НИПИНефть. Волгоград, 1983. С. 89-90.

52. Диффузионное цинкование насосных штанг / Жолудев М. Д., Проскуркин Е. В., Гарбунов Н. С., Рабинович А. М. и др. // Нефтяное хозяйство. 1972. №9. С. 53-55.

53. Тараевский С. И. Повышение долговечности насосных штанг, эксплуатирующихся в сероводородсодержащих средах : дис. ... канд. техн. наук : 05.04.07. Москва, 1984. 178 с.

54. Зохранов А. Г., Рабинович А. М., Ахмедов Б. М. Упрочнение глубиннонасосных штанг пластическим деформированием // Химическое и нефтяное машиностроение. 1974. №1. С. 29-31.

55. Джабарзаде Д. А., Атакшиев А. Н., Толкачёв Ю. И. О снижении аварийности насосных штанг на месторождениях Башкирии // Нефтепромысловое дело. 1974. №10. С. 26-28.

56. Джабарзаде Д. А., Литровенко М. Г., Ломакин А. С. Некоторые вопросы работоспособности насосных штанг, закалённых ТВЧ // Сер. Машины и нефтяное оборудование. Москва : РНТС. ВНИИОЭНГ, 1987. №2. С. 6-7.

57. Hoffman E. L. Finite Element Analysis of Sucker Rod Couplings with Guidelines for Improving Fatigue Life: Sandia report. Sandia National Laboratories, 1997. 66 p.

58. Матвейчук А. Т., Михайлов И. В. Опыт применения стеклопластиковых насосных штанг за рубежом. Москва : ВНИИОЭНГ, 1989. 18 с.

59. Копей Б. В., Копей В. Б., Копей І. Б. Насосні штанги свердловинних установок для видобування нафти. Івано-Франківськ : ІФНТУНГ, 2009. 406 с.

60. Gibbs S. G. Application of Fiberglass Sucker Rods // SPE Production Engineering. SPE, Nabla Corp., May 1991. P. 147-153.

61. Кузьмін О. О. Вдосконалення свердловинного обладнання для попередження відкладів піску, парафіну та смол : автореф. дис. ... канд. техн. наук : 05.05.12. ІФНТУНГ. Івано-Франківськ, 2012. 18 с.

62. Pat. US20120141194A1, Int. Cl. E21B 17/04, B32B 38/04. Sucker Rod End Fittings and Method of Using Same / Russell P. Rutledge, SR.; Russell P. Rutledge, JR.; Ryan B. Rutledge. Filed: Feb. 10, 2012; Pub. Date: Jun. 7, 2012. 10 p.

63. Копей В. Б., Чаплинський С. С. Аналіз і раціоналізація конструкцій протекторів для насосних штанг за допомогою параметричного тривимірного моделювання та методу скінченних елементів // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2004. №1(7). С. 23-28.

64. Пат. UA 88804, МПК E21B 17/00, E21B 37/00. Пристрій для обертання колони насосних штанг / Копей Б. В., Копей В. Б., Онищук С. Ю. № a200711768; заявл. 24.10.2007; опубл. 25.11.2009, Бюл. №22.

65. Копей Б. В., Кузьмін О. О., Онищук С. Ю. Обладнання для попередження відкладень асфальтосмолистых речовин, парафіну та піску: монографія. Серія «Нафтогазове обладнання», том 3. Івано-Франківськ : ІФНТУНГ, 2014. 216 с.

66. Визначення крутного моменту колони насосних штанг, створюваного скребками-протекторами / Копей Б. В., Онищук С. Ю., Гімер П. Р., Копей В. Б. // Анотації Міжнародної науково-технічної конференції "Нафтогазова енергетика:

проблеми і перспективи". Івано-Франківськ, 20-23 жовтня 2009 р. Івано-Франківськ : ІФНТУНГ, 2009. С. 64.

67. Розрахунок величини крутного моменту, створюваного лопатями протектора насосних штанг / Гімер П. Р., Копей В. В., Онищук О. О., Онищук С. Ю., Копей В. Б. // Розвідка та розробка нафтових і газових родовищ. 2009. №4(33). С. 63-67.

68. Визначення раціональних параметрів протектора насосних штанг / Гімер П. Р., Копей В. В., Онищук О. О., Онищук С. Ю., Копей В. Б. // Розвідка та розробка нафтових і газових родовищ. 2010. №1(34). С. 77-81.

69. Копей В. Б. Аналіз способів підвищення ресурсу муфтового різьбового з'єднання насосних штанг // Анотації Міжнародної науково-практичної конференції молодих вчених "Техніка і прогресивні технології у нафтогазовій інженерії", м. Івано-Франківськ, 16-20 вересня 2008 р. Івано-Франківськ : Факел, 2008. С. 55.

70. Копей В. Б. Аналіз способів підвищення ресурсу муфтового різьбового з'єднання насосних штанг // Розвідка та розробка нафтових і газових родовищ. 2008. № 4(29). С. 66-72.

71. Биргер И. А., Иосилевич Г. Б. Резьбовые и фланцевые соединения. Москва : Машиностроение, 1990. 368 с.

72. Михайлюк В. В. Підвищення ефективності експлуатації різьбових з'єднань насосних штанг : автореф. дис. ... канд. техн. наук : 05.05.12. ІФНТУНГ. Івано-Франківськ, 2013. 19 с.

73. Пат. UA 58828A, МПК E21B17/04. Муфтове різьбове з'єднання насосних штанг / Копей В. Б., Петрина Ю. Д., Стеліга І. І. № 2002118793; заявл. 06.11.2002; опубл. 15.08.2003, Бюл. №8.

74. The NOV Grant Prideco™ GPDS connection. URL: [https://www.nov.com/Segments/Wellbore_Technologies/Grant_Prideco/Connection_Technologies/Grant_Prideco_Double_Shoulder_\(GPDS\)_connection.aspx](https://www.nov.com/Segments/Wellbore_Technologies/Grant_Prideco/Connection_Technologies/Grant_Prideco_Double_Shoulder_(GPDS)_connection.aspx) (Last accessed: 10.02.2019).

75. US Patent No: 5908212. Ultra high torque double shoulder tool joint; filed May 2, 1997; date of patent Jun. 1, 1999.

76. Аналіз сучасних конструкцій замкових з'єднань обважнених бурильних труб / Артим В. І. та ін. // Нафтогазова енергетика. 2017. № 2(28). С. 22-30.

77. Numerical and experimental distribution of stress fields for double shoulder tool joint / Yuanhua Lin et al. // Engineering Failure Analysis. 2011. Volume 18, Issue 6. P. 1584-1594.

78. Zhu X., Zhi Z. Design of an ultra-high torque double shoulder drill-pipe tool joint for extended reach wells // Natural Gas Industry B. 2017. Volume 4, Issue 5. P. 374-381.

79. Dong L., Zhu X., Yang D. Study on mechanical behaviors of double shoulder drill pipe joint thread // Petroleum. 2018. DOI: 10.1016/j.petlm.2018.01.004.

80. Three-dimensional mechanical analysis of the double-shouldered tool joint / Di Q. et al. // Shiyou Xuebao/Acta Petrolei Sinica. 2012. Volume 33. P. 871-877.

81. The influence of clearance of secondary shoulder on performance of double shoulder drill pipe joint / Chen F. et al. // Gongcheng Lixue/Engineering Mechanics. 2013. Volume 30. P. 353-357.

82. Трубы нефтяного сортамента: Справочник / Под общей ред. А. Е. Сарояна. - 3-е изд., перераб. и доп. Москва : Недра, 1987. 488 с.

83. Сароян А. Е., Субботин М. А. Эксплуатация колонн насосно-компрессорных труб: научное издание. Москва : Недра, 1985. 216 с.

84. Классификатор. Виды повреждений насосно-компрессорных труб при эксплуатации / Антипов Ю. Н. и др. Самара : ООО "ВНИИТнефтетрубы", 2016. 28 с.

85. Богатов Н. А., Салихьянов Д. Р., Богатов А. А. Разработка и исследование технологии производства насоснокомпрессорных труб из композиционных материалов и оценка их долговечности // Инновационные технологии в металлургии и машиностроении: материалы 6-й Международной молодежной научно-практической конференции «Инновационные технологии в металлургии и машиностроении. Уральская научно-педагогическая школа имени

профессора А. Ф. Головина», г. Екатеринбург, 29 октября - 1 ноября 2012 г. Екатеринбург : Изд-во Урал. ун-та, 2012. С. 669-674.

86. Тронов В. П. Механизм образования смоло-парафиновых отложений и борьба с ними. Москва : Недра, 1970. 171 с.

87. Насосні штанги і труби з полімерних композитів: проектування, розрахунок, випробування / Копей Б. В., Максимук О. В., Щербина Н. М., Розгонюк В. В., Копей В. Б. Львів : ІППИМ ім. Я. С. Підстригача НАН України, 2003. 352 с.

88. Насосная добыча высоковязкой нефти из наклонных и обводненных скважин / Уразаков К. Р., Богомольный Е. И., Сейтпагамбетов Ж. С., Газаров А. Г. Под ред. М. Д. Валеева. Москва : ООО "Недра-Бизнесцентр", 2003. 303 с.

89. Уразаков К. Р. Эксплуатация наклонно направленных насосных скважин. Москва : Недра, 1993. 171 с.

90. Юшин Е. С. Оценка коррозионно-усталостного состояния насосно-компрессорных труб в минерализованных средах : дис. ... канд. техн. наук : 05.02.13. Ухта, 2014. 190 с.

91. Evaluation of Standard API Casing Connections and Parametric API Buttress Improvement by Finite Element Analysis / R. R. Porcaro, L. C. Candido, V. B. Trindade, G. L. de Faria, L. B. Godefroid // Materials Research. 2017. vol. 20, n. 1. P. 130-137. DOI: 10.1590/1980-5373-mr-2015-0613.

92. Modeling of preloaded threaded pipe connections / Jeroen Van Wittenberghe, Patrick De Baets, Wim De Waele // Proceedings of the 8th national congress on theoretical and applied mechanics, May 28-29, 2009, Brussels, Belgium. P. 149-156. URL: <http://hdl.handle.net/1854/LU-673974> (Last accessed: 10.02.2019).

93. Оптимізація параметрів різьбових з'єднань : Звіт про НДР (заключний) / В. Б. Копей, В. В. Михайлюк, Н. В. Шатинський / ІФНТУНГ. 2010. 143 с. Ф27/51-2010. № держреєстрації 0110U002631.

94. Використання скінченно-елементного методу для аналізу різьб насосно-компресорних труб / Копей Б. В., Копей В. Б., Савула С. Ф., Михайлюк В. В. // Сб. трудов V Международ. науч.-технич. конф. "Повышение качества, надежности и

долговечности технических систем и технологических процессов", 3-10 декабря 2006г., Шарм эль Шейх (Египет). Хмельницкий : ХНУ, 2006. С. 105-109.

95. РД 39–136–95. Инструкция по эксплуатации насосно-компрессорных труб. Введ. 1986–11–20. Самара : ВНИИТнефть, 1995. 159 с.

96. Пат. RU 2314575, МПК G10K11/172, F16L55/02. Способ снижения вибраций насосно-компрессорных труб / Савиных Ю. А., Дунаев С. А.; заявл. 03.28.2005; опубл. 01.10.2008.

97. Мищенко И. Т. Скважинная добыча нефти : учеб. пособие для вузов. Москва : ФГУП Изд-во «Нефть и газ» РГУ нефти и газа им. И. М. Губкина, 2003. 816 с.

98. Белов И. Г. Исследование работы глубинных насосов динамографом. Москва : ГОСТОПТЕХИЗДАТ, 1960. 128 с.

99. Ишемгужин И. Е. О релаксационных колебаниях погружного электроцентробежного насоса при подъеме его из скважины // Нефтегазовое дело. 2009. Том 7, № 2. С. 96-99.

100. Ишемгужин И. Е. Рассогласование движения устьевого штока и плунжера насоса при релаксационных колебаниях // Оборудование и технологии для нефтегазового комплекса. 2010. №5. С. 4-7.

101. Палійчук І. І., Копей В. Б., Марцинків О. Б. Підвищення герметичності та ремонтпридатності різьбових з'єднань обсадних і насосно-компресорних труб металізаційним покриттям // Розвідка та розробка нафтових і газових родовищ. 2011. №4(41). С. 87-91.

102. Онисько О. Р. Про функціональну залежність величини половинного кута профілю замкової нарізі від величин переднього кута, кута нахилу та половинного кута профілю різальної кромки різця // Вісник Національного університету "Львівська політехніка". Оптимізація виробничих процесів і технічний контроль у машинобудуванні та приладобудуванні. 2017. № 867. С. 19-28.

103. Видобування нафти. Глибинонасосний спосіб експлуатації свердловин. Штангові свердловинні насоси. Стандарт організації (проект). ВАТ «Укрнафта», 2007.

104. Долов Т. Р. Исследование работы клапанных узлов скважинных штанговых насосных установок : дис. ... канд. техн. наук : 05.02.13. ФГБОУ ВО Ухтинский государственный технический университет, 2017.

105. Галимуллин М. Л. Разработка технических средств повышения работоспособности скважинных плунжерных насосов : автореф. дис. ... к.т.н. : 05.02.13. Уфимский государственный нефтяной технический университет. Уфа, 2004. 25 с.

106. Копей Б. В., Копей І. Б. Надійність штангових свердловинних насосів з втулочними і безвтулочними циліндрами в свердловинах з значною обводненістю. Івано-Франківськ : ІФДТУНГ, 1995. 7 с. Деп. в ДНТБ України 29.04.96. №1041. Ук96.

107. Воробьев В. Д., Смирнова Н. М. Технический уровень штанговых глубинных насосов за рубежом // Обзорная информация. Серия "Машины и нефтяное оборудование". Москва : ВНИИОЭНГ, 1982. №7/17. 16 с.

108. Зейгман Ю. В., Гумеров О. А., Генералов И. В. Выбор оборудования и режима работы скважин с установками штанговых и электроцентробежных насосов: Учеб. пособие. Уфа : Изд-во УГНТУ, 2000. 120 с.

109. Насретдинов М. Р. Повышение эффективности эксплуатации штанговых глубинных насосов за счет применения самоустанавливающихся магнитных клапанов в ООО «Башнефть-Добыча» // Нефтегазовое дело. 2019. Том 17, № 1. С. 56-64. DOI: 10.17122/ngdelo-2019-1-56-64.

110. Копей В. Б. Моделювання клапана свердловинного штангового насоса методом обчислювальної гідродинаміки в Abaqus/CAE® // Матеріали міжнародної науково-технічної конференції "Математичне моделювання прикладних задач математики, фізики, механіки ММАР-2013", м. Харків, 5 – 25 травня 2013 р. Харків : ХНАДУ, 2013. URL: <http://files.khadi.kharkov.ua/images/Fizika.pdf> (дата звернення: 29.02.2016).

111. Моделювання роботи клапана з демпферною камерою для бурового насоса методом кінцевих елементів / Копей Б. В., Копей В. Б., Лівак І. Д., Чаплинський С. С. // Розвідка та розробка нафтових і газових родовищ. 2005. № 4(17). С. 52-55.

112. Sureshkumar C., Tummescheit H. Physical Design of Hydraulic Valves in Modelica // Proceedings of the 10 th International Modelica Conference March 10-12, 2014, Lund, Sweden. 2014. P. 627-636. DOI: 10.3384/ECP14096627.

113. Viel A. Strong Coupling of Modelica System-Level Models with Detailed CFD Models for Transient Simulation of Hydraulic Components in their Surrounding Environment // Proceedings 8th Modelica Conference, Dresden, Germany, March 20-22, 2011. P. 256-265. DOI: 10.3384/ecp11063256.

114. Моделювання роботи безманжетного клапана поршневого бурового насоса / С. С. Чаплинський, Б. В. Копей, І. Д. Лівак, В. Б. Копей // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2005. № 3(12). С. 46-49.

115. Идельчик И. Е. Справочник по гидравлическим сопротивлениям / Под ред. М.О.Штейнберга. 3-е изд., перераб. и доп. Москва : Машиностроение, 1992. 672 с.

116. Федоров А. Г. Совершенствование методики ремонта нефтегазопроводов с применением стальных обжимных муфт : автореф. дис. ... канд. техн. наук : 25.00.19. УГТУ. Ухта, 2017. 20 с.

117. Експериментальні дослідження властивостей дослідних зразків бандажів нафтогазопроводів в умовах експлуатації : Звіт про НДР (заключний) / Б. В. Копей, Т. П. Венгринюк та ін. / ІФНТУНГ. 2018. 92 с. № держреєстрації 0118U004335.

118. Копей Б. В., Стеліга І. І., Копей В. Б. Комплексне зміцнення насосних штанг металізаційними покриттями і склопластиковою ізоляцією // Нафтогазова енергетика. 2009. №2 (11). С. 5-11.

119. ВСН 39-1.10-001-99. Инструкция по ремонту дефектных труб магистральных газопроводов полимерными композиционными материалами. Москва : ОАО «Газпром», 2000. 25 с.

120. ВРД 39-1.10-013-2000. Руководящий документ по применению композитных материалов фирмы Порсил лтд. Москва : ОАО «Газпром», 2000. 91 с.

121. Belzona 1111 (Super Metal). An epoxy-based composite for metal repair. URL: <https://www.belzona.com/en/products/1000/1111.aspx> (Last accessed 01.01.2020).

122. Венгринюк Т. П. Розроблення ізоляційно-композитного покриття для підвищення міцності нафтогазопроводів з тривалим терміном експлуатації : автореф. дис. ... канд. техн. наук : 05.15.13. ІФНТУНГ. Івано-Франківськ, 2013. 20 с.

123. Експериментальні дослідження властивостей дослідних зразків бандажів нафтогазопроводів в умовах експлуатації: Звіт про НДР (заключний) / Б. В. Копей, Т. П. Венгринюк та ін. Івано-Франківськ : ІФНТУНГ, 2018. 92 с. № держреєстрації 0118U004335.

124. Копей Б. В., Найда А. М., Копей В. Б. Експериментальна оцінка ефективності бандажів для зміцнення пошкоджених трубопроводів // Нафтогазова енергетика. 2009. № 1(10). С. 60-63.

125. Гальмування росту тріщин в насосних штангах полімерними композиційними стрічками / Копей Б. В., Онищук С. Ю., Стеліга І. І., Копей В. Б. // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2009. № 4(22). С. 92-100.

126. Онищук С. Ю. Підвищення ресурсу штангової свердловинної насосної установки при високому вмісті парафінів та корозійних агентів у продукції свердловини : дис. ... канд. техн. наук : 05.05.12. ІФНТУНГ. Івано-Франківськ, 2010. 207 с.

127. РД 39-1-592-81. Типовая технологическая инструкция по подготовке к эксплуатации и ремонту насосно-компрессорных труб в цехах центральных

трубных баз производственных объединений миннефтепрома. Куйбышев : ВНИИТнефть, 1981. 81 с.

128. Окрушко Е. И., Ураксеев М. А. Дефектоскопия глубиннонасосных штанг. Москва : Недра, 1983. 112 с.

129. Алешин Н. П., Лупачев В. Г. Ультразвуковая дефектоскопия: Справ. пособие. Минск : Выш. шк., 1987. 271 с.

130. Установка УМД-104М для магнитной дефектоскопии и толщинометрии насосно-компрессорных труб повторного применения / О. А. Булычев, С. А. Шлеенков, В. М. Сенив, А. С. Шлеенков // Сварка и диагностика. Екатеринбург : УрФУ, 2015. С. 291-297.

131. Вирновский А. С. Теория и практика глубинно-насосной добычи нефти. Избранные труды. Москва : Недра, 1971. 184 с.

132. Тараевский С. И., Копей Б. В. Анализ поломок глубиннонасосных штанг в НГДУ "Долинанефтегаз" // Разведка и разработка нефтяных и газовых месторождений. Респ. межвед. научн.-техн. сборник. Вып. 19. Львов : Вища школа, 1982. С. 104-107.

133. Аналіз відмов колон насосних штанг в НГВУ "Надвірнанафтогаз" / П. В. Пушкар, Я. Ю. Павлюк, Т. Б. Матвіїшин, В. І. Артим // Розвідка та розробка нафтових і газових родовищ. 2006. №3(20). С. 122-127.

134. Хакимов Т. А. Исследование и усовершенствование технологии и технических средств добычи высоковязкой нефти штанговыми установками : автореф. дис. ... к.т.н. : 25.00.17, 05.02.13. ИМАШ РАН. Москва, 2015. 24 с.

135. Кочиков М. А. Повышение эффективности эксплуатации штанговых насосных установок в високообводненных скважинах : автореф. дис. ... к.т.н. : 25.00.17. ФГБОУ ВПО «Уфимский государственный нефтяной технический университет». Уфа, 2014. 24 с.

136. Справочное руководство по проектированию разработки и эксплуатации нефтяных месторождений. Добыча нефти. / Р. С. Андриасов, И. Т. Мищенко, А. И. Петров и др. Под общ. ред. Ш. К. Гиматудинова. Москва : Недра, 1983. 455 с.

137. Персиянцев М. Н. Добыча нефти в осложненных условиях. ООО "Недра-Бизнесцентр", 2000. 653 с.

138. Третьякова Г. И., Белозеров Г. И. Метод определения зависимости числа обрывов штанг от глубины подвески насоса по статистическим данным // Реферативный НТС «Нефтепромысловое дело», Москва : ВНИИОЭНГ, 1974. № 2. С. 21—25.

139. Ришмюллер Г., Майер Х. Добыча нефти глубинными штанговыми насосами: Пер. с нем. Москва : Фест-Альпине, 1988. 151 с.

140. Дунаев И. В. Диагностика и контроль состояния скважинной штанговой насосной установки на основе динамометрирования и нейросетевых технологий : автореф. дис. ... к.т.н. : 05.13.06. Уфимский государственный авиационный технический университет. Уфа, 2007. 19 с.

141. Способ контроля параметров работы и технического состояния штанговых скважинных насосных установок / Софьина Н. Н., Шишлянников Д. И., Корнилов К. А., Вагин Е. О. // Master's journal. 2016. №1. С. 247-257.

142. Sucker-Rod Pumping Failures Diagnostic System (SPE SPE-134975-PP) / G. V. L. Moises, S. F. A. Andrade, A. C. Bicharra, Y. S. Ferreira // SPE Annual Technical Conference and Exhibition, Florence, Italy, 19–22 September 2010. Society of Petroleum Engineers, 2010.

143. Адонин А. Н. Добыча нефти штанговыми насосами. Москва : Недра, 1979. 209 с.

144. Ивахненко А. Г., Юрачковский Ю. П. Моделирование сложных систем по экспериментальным данным. Москва : Радио и связь, 1987. 120 с.

145. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными. : Пер. с англ. Санкт-Петербург : ООО "Альфа-книга", 2017. 480 с.

146. Справочник по добыче нефти / В. В. Андреев, К. Р. Уразаков, В. У. Далимов и др.; Под ред. К. Р. Уразакова. Москва : ООО "Недра-Бизнесцентр", 2000. 374 с.

147. Gibbs S. G. Predicting the behavior of sucker rod pumping systems // *Journal of Petroleum Technology*. 1963. Vol. 15, Issue 7. P. 769-778.

148. Gibbs S. G. *Rod Pumping: Modern Methods of Design, Diagnosis and Surveillance*. Publisher: Author, 2012. 660 p. ISBN-13: 978-0-984-9661-0-3.

149. Romero O. J., Almeida P. Numerical simulation of the sucker-rod pumping system // *INGENIERIA E INVESTIGACION*. DECEMBER 2014. VOL. 34. No. 3. P. 4-11.

150. Ковшов В. Д., Сидоров М. Е., Светлакова С. В. Моделирование динамограммы станка-качалки. Нормальная работа насоса // *Нефтегазовое дело*. 2004. № 2. С. 75-81.

151. Knapp R. M. *A Dynamic Investigation of Sucker-Rod Pumping*. KU ScholarWorks The University of Kansas Theses and Dissertations Collection B.S. University of Kansas, 1963. 47 p.

152. Wang G. W., Rahman S. S., Yang G. Y. An improved model for the sucker rod pumping system // *11th Australasian Fluid Mechanics Conference*. University of Tasmania, Hobart, Australia, 14-18 Decemder, 1992. P. 1137-1140.

153. An Approach to the Design Calculation of Sucker Rod Pumping Systems in Coalbed Methane Wells / LIU Xinfu, QI Yaoguang, LI Yanxiang, LIU Chunhua // *CHINESE JOURNAL OF MECHANICAL ENGINEERING*. 2011. Vol. 24. P. 1-10.

154. Takacs G. *Sucker-Rod Pumping Manual*. PennWell Corporation, 2003. 395 p.

155. Вассерман И. Н. Продольные колебания упругих стержневых систем с граничными условиями, определяемыми многозначными соотношениями : автореф. дис. ... канд. физ.-мат. наук : 01.02.04. Пермь, 1999. 18 с.

156. Norton J. R. Dynamic Loads in Sucker-Rods // *Petroleum Engineer*. April 1960. B-33-B-41.

157. Qingyou Liu, Yufa He, Hailan Wang. Computer Simulation on Working Action of the Sucker Rod Pumping System. URL: <http://www.paper.edu.cn/index.php/default/scholar/downpaper/liuqingyou-200705-18> (Last accessed: 01.01.2017).

158. Агамалов Г. Б. Особенности механизированной добычи нефти из глубоких скважин // Нефтегазовое дело. 2009. Том 7. №2. С. 64-67.

159. Грудз В. Я., Наследник С. В. Методология разработки математической модели для исследования конструкций станка-качалки при добыче углеводородных соединений // Системы. Методы. Технологии Методология разработки. 2014. № 1 (21). С. 51-56.

160. Газаров А. Г. Разработка методов снижения износа штангового насосного оборудования в наклонно направленных скважинах : автореф. дис. ... канд. техн. наук : 05.02.13. Уфа, 2004. 16 с.

161. Xing M. An improved numerical simulation research for plunger pump in the condition of Newtonian fluid // Journal of Measurements in engineering. March 2016, volume 4, issue 1. P. 32-42.

162. Modelica and the Modelica Association. URL: <https://modelica.org> (Last accessed: 01.01.2017).

163. OpenModelica User's Guide. Release v1.12.0. URL: <https://www.openmodelica.org/doc/OpenModelicaUsersGuide/OpenModelicaUsersGuide-v1.12.0.pdf> (Last accessed: 01.03.2019).

164. EcosimPro Modelling and Simulation Software. URL: http://www.ecosimpro.com/wp-content/uploads/2015/02/ecosimpro_brochure_presentation_en.pdf (Last accessed: 01.01.2017).

165. MapleSim User's Guide. Maplesoft, 2013. 254 p.

166. Fritzson P. A. Principles of object oriented modeling and simulation with Modelica 3.3: a cyber-physical approach. 2nd edition. Wiley-IEEE Press, 2014. 1256 p.

167. Tiller M. Introduction to Physical Modeling with Modelica. Springer, 2001. 345 p.

168. Копей В. Б., Біленко П. О. Реалізація декларативного програмування в Python за допомогою математичних бібліотек SymPy та SciPy // Молодь: освіта, наука, духовність : тези доповідей XIII Всеукр. наук. конф., м. Київ, 12–14 квітня 2016 р. у II част. Ч. II. Київ : Університет «Україна», 2016. С. 227-229.

169. SymPy 1.0 documentation. URL: <http://docs.sympy.org> (Last accessed: 01.01.2017).
170. SciPy: open source scientific tools for Python / Jones E., Oliphant E., Peterson P., et al. <http://www.scipy.org> (Last accessed: 01.05.2019).
171. Добрынин С. Компьютерное моделирование методом подвижных клеточных автоматов. Saarbrucken Germany : LAP LAMBERT Academic Publishing, 2011. 132 p. ISBN 978-3-8443-5954-1.
172. Тоффоли Т., Марголюс Н. Машины клеточных автоматов: Пер. с англ. Москва : Мир, 1991. 280 с.
173. Карпов Ю. Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5. Санкт-Петербург : БХВ-Петербург, 2006. 390 с.
174. Энциклопедия кибернетики / Отв. ред. Глушков В. М. Т. 1. Киев : Главная редакция Украинской Советской Энциклопедии, 1974. 608 с.
175. Wolfram S. A New Kind of Science. Wolfram Media, Inc, 2002. 1197 p.
176. Аладьев В. З. Классические однородные структуры. Клеточные автоматы. Fultus™Books, 2009. 535 с.
177. Поликарпова Н. И., Шалыто А. А. Автоматное программирование. Санкт-Петербург : СПбГУ ИТМО, 2008. 167 с.
178. Копей В. Б. Скінченно-елементний аналіз муфтового різьбового з'єднання насосних штанг // Розвідка та розробка нафтових і газових родовищ. 2003. №2(7). С. 54-58.
179. Копей В. Б., Копей Б. В., Найда А. М. Определение остаточного ресурса трубопровода с дефектами после их ремонта композитными бандажами // Управление качеством в нефтегазовом комплексе. 2007. №4. С. 26-28.
180. Копей Б. В., Копей В. Б. Застосування засобів тривимірного моделювання для проектування штангообертача // Розвідка та розробка нафтових і газових родовищ. 2003. № 4(9). С. 99-101.
181. Копей В. Б., Петрина Ю. Д. Моделювання циклічної довговічності евольвентної зубчастої передачі зі зношеною шестернею // Сучасні міждисциплінарні дослідження: історія, сьогодення, майбутнє: результати

одинадцятій міжнародній конференції : збірник наукових праць (12 червня 2015р) / відп. ред. Приходько М.М., Тонких С.В. Київ : «Аграр Медіа Груп», 2015. С. 45-51.

182. Копей В. Б. Принципи побудови моделей евольвентних зубчастих передач для SolidWorks Simulation // Матеріали IV Міжнародного інтернет-симпозіуму "Сучасні проблеми інженерної механіки", Луцьк-2015. Луцьк, 2015. URL: <http://konf2015.web44.net/materials.htm>.

183. Копей В. Б., Панчук А. Г., Воробець М. І. Система автоматизованого проектування ексцентрико-циклоїдальних передач // Теорія і практика розвитку наукових знань (частина IV) : матеріали II Міжнародної науково-практичної конференції м. Київ, 28-29 грудня 2017 року. Київ : МЦНД, 2017. С. 29-31.

184. Abaqus 6.14 theory guide. Providence, RI : Dassault Systemes Simulia Corp., 2014. 1174 p.

185. Манилык Т., Ильин К. Практическое применение программного комплекса ABAQUS в инженерных задачах. Версия 6.5. Москва : МФТИ, ТЕСИС, 2006. 67 с.

186. Dhondt G. The Finite Element Method for Three-Dimensional Thermomechanical Applications. Wiley, 2004. 362 p.

187. Копей В. Б. Використання вільного програмного забезпечення для розробки системи скінченно-елементного аналізу різьбових з'єднань нафтогазового обладнання // Матеріали міжнародної науково-технічної конференції "Машини, обладнання і матеріали для нарощування вітчизняного видобутку та диверсифікації постачання нафти і газу" ПМ – 2016, м. Івано-Франківськ, 16-20 травня 2016 р. Івано-Франківськ : ІФНТУНГ, 2016. С. 277-280.

188. Копей В. Б., Панчук В. Г., Присяжнюк П. М. Принципи побудови системи скінченно-елементного моделювання композиційних матеріалів нафтогазового обладнання // Матеріали міжнародної науково-технічної конференції "Машини, обладнання і матеріали для нарощування вітчизняного видобутку та диверсифікації постачання нафти і газу" ПМ – 2016, 16-20 травня 2016 р. м. Івано-Франківськ. Івано-Франківськ : ІФНТУНГ, 2016. С. 248-251.

189. OPEN CASCADE. URL: <http://www.opencascade.com> (Last accessed: 2019-05-01).

190. Paviot T. pythonOCC, 3D CAD/CAE/PLM development framework for the Python programming language. URL: <http://www.pythonocc.org> (Last accessed: 2019-05-01).

191. Копей В. Б. Ядро геометричного моделювання Open CASCADE Technology для Python-програмістів: Методичні вказівки для самостійної роботи. Івано-Франківськ : ІФНТУНГ, 2017. 47 с.

192. FreeCAD: параметрический 3D CAD моделер с открытым исходным кодом. URL: <https://www.freecadweb.org> (Last accessed: 01.01.2019).

193. Копей В. Б., Скальський Р. В. Принципи розробки прикладних САПР на основі FreeCAD мовою Python // Молодь: освіта, наука, духовність : тези доповідей XII Всеукр. наук. конф., м. Київ, 21–22 квітня 2015 р. У II част., ч. II. Київ : Університет «Україна», 2015. С. 255-257.

194. Копей В. Б., Угринчук Р. В. Можливості FreeCAD для швидкої розробки прикладних програм, що працюють з геометричними моделями // Науково-практичний семінар "Графічна освіта у закладах вищої освіти: стан та перспективи" : збірник тез доповідей (м. Івано-Франківськ, 19-21 вересня 2018 р.). Івано-Франківськ : ІФНТУНГ, 2018. С. 29-32.

195. Geuzaine C., Remacle J.-F. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities // International Journal for Numerical Methods in Engineering. 2009. Volume 79, Issue 11. P. 1309-1331.

196. Копей В. Б. Застосування системи CAD/FEA для розрахунку і оптимізації різьбових з'єднань нафтогазового обладнання // Розвідка та розробка нафтових і газових родовищ. 2009. № 3(32). С. 43-49.

197. Onysko O., Kopei V., Panchuk V. Theoretical Investigation of the Tapered Thread Joint Surface Contact Pressure in the Dependence on the Profile and the Geometric Parameters of the Threading Turning Tool // INNOVATIVE IDEAS IN SCIENCE 2018 (08-09 November 2018, Baia Mare, Romania) : Book of Abstracts.

Baia Mare : Editura Casei Corpului Didactic Baia Mare "Maria Montessori", 2018. P. 84.

198. Копей В. Б. Обґрунтування доцільності збільшення довжини розвантажувальної канавки ніпеля насосної штанги // Технологічний аудит та резерви виробництва (Спецвипуск. Матеріали науково-практичної конференції "Наукові підсумки 2012 р.", м. Харків, 2012 р.). 2012. № 6/2 (8). С. 7-8.

199. Трощенко В. Т., Сосновский Л. А. Сопротивление усталости металлов и сплавов. Справочник. Киев : Наукова думка, 1987. 1311 с.

200. Бураго Н. Г., Журавлев А. Б., Никитин И. С. Модели многоосного усталостного разрушения и оценка долговечности элементов конструкций // Известия Российской академии наук. Механика твердого тела. 2011. № 6. С. 22-33.

201. Трощенко В. Т. Рассеянное усталостное повреждение металлов и сплавов. Сообщение 3. Деформационные и энергетические критерии // Проблемы прочности. 2006. № 1. С. 5-31.

202. fe-safe 6 User Manual. Volume 2. Fatigue Theory Reference Manual. Safe technology limited, 2006. 264 p.

203. Серенсен С. В., Когаев В. П., Шнейдерович Р. М. Несущая способность и расчеты деталей машин на прочность. Москва : Машиностроение, 1975. 488 с.

204. Расчет на прочность деталей машин: Справочник / И. А. Биргер, Б. Ф. Шорр, Г. Б. Иосилевич. 3-е изд., перераб. и доп. Москва : Машиностроение, 1979. 702 с.

205. Sines G. Behavior of metals under complex static and alternating stresses // Metal Fatigue / eds. G. Sines, J. L. Waisman. New York : McGraw-Hill, 1959. P. 145-169.

206. Metal Fatigue in Engineering / Ralph I. Stephens, Ali Fatemi, Robert R. Stephens, Henry O. Fuchs. John Wiley & Sons, 2000. 496 p.

207. Kandil F. A., Brown M. W., Miller K. J. Biaxial low cycle fatigue fracture of 316 stainless steel at elevated temperatures // Book 270. London : The Metals Society, 1982. P. 203-210.

208. fe-safe 6 User Manual. Volume 1. User Guide. Safe technology limited, 2014. 408 p.
209. Сиратори М., Миёси Т., Мацусита Х. Вычислительная механика разрушения: Пер. с японск. Москва : Мир, 1986. 334 с.
210. Морозов Е. М. ANSYS в руках инженера: Механика разрушения. Изд. 2-е, испр. Москва : ЛЕНАНД, 2010. 456 с.
211. Meinhard Kuna. Finite Elements in Fracture Mechanics. Theory—Numerics—Applications. Springer, 2010. 464 p.
212. Saaksvuori A., Immonen A. Product lifecycle management. Third Edition. Berlin, Heidelberg : Springer-Verlag, 2008. 268 p.
213. Копей В. Б. Абстрактна модель інформаційної системи підтримки життєвого циклу виробу // Прикарпатський вісник НТШ. Число. 2017. №2(38). С. 71-96.
214. Норенков И. П. Основы автоматизированного проектирования: Учеб. для вузов. 2-е изд., перераб. и доп. Москва : Изд-во МГТУ им. Н. Э. Баумана, 2002. 336 с.
215. Информационно-вычислительные системы в машиностроении CALS-технологии / Ю. М. Соломенцев, В. Г. Митрофанов, В. В. Павлов, Л. В. Рыбаков. Москва : Наука, 2003. 292 с.
216. Информационная поддержка жизненного цикла изделий машиностроения : принципы, системы и технологии CALS/ИПИ : учеб. пособие для студ. высш. учеб. заведений / А. Н. Ковшов, Ю. Ф. Назаров, И. М. Ибрагимов, А. Д. Никифоров. Москва : Издательский центр «Академия», 2007. 304 с.
217. Bertalanffy L. von. General System theory: Foundations, Development, Applications. 1st ed. New York : George Braziller, Inc., 1968. 289 p.
218. NATO CALS Handbook. Version 2. June 2000. Brussel: NATO CALS Office, 2000. 342 p.
219. Судов Е. В., Левин А. И. Концепция развития CALS-технологий в промышленности России. Москва : НИЦ CALS-технологий «Прикладная логистика», 2002. 131 с.

220. Stark J. Product Lifecycle Management: in 2 volumes. Third Edition. Geneva: Springer, 2016.
221. Боулдинг К. Общая теория систем - скелет науки // Исследования по общей теории систем. Москва : Прогресс, 1969. С. 106-124.
222. Системный анализ и принятие решений: Словарь-справочник : учеб. пособие для вузов / Под ред. В. Н. Волковой, В. Н. Козлова. Москва : Высш. шк., 2004. 616 с.
223. Флейшман Б. С. Элементы теории потенциальной эффективности сложных систем. Москва : Советское радио, 1971. 224 с.
224. Берталанфи Л. Общая теория систем — критический обзор // Исследования по общей теории систем: Сборник переводов / Общ. ред. и вст. ст. В. Н. Садовского и Э. Г. Юдина. Москва : Прогресс, 1969. С. 23-82.
225. Принципы материалистической диалектики как теории познания / Ответственный редактор В. А. Лекторский. Москва : Наука, 1984. 304 с.
226. Богданов А. А. Тектология (Всеобщая организационная наука). В 2-х кн. Москва : Экономика, 1989.
227. Плотинский Ю. М. Модели социальных процессов: Учебное пособие для высших учебных заведений. Изд. 2-е, перераб. и доп. Москва : Логос, 2001. 296 с.
228. Большая российская энциклопедия. URL: <https://bigenc.ru> (Last accessed: 01.01.2017).
229. Мещеряков Б., Зинченко В. Большой психологический словарь. Санкт-Петербург : прайм-ЕВРОЗНАК, 2004. 672 с.
230. Фрейд З. По ту сторону принципа удовольствия. Психология бессознательного. Перевод с нем. А. М. Боковой. Москва : Просвещение, 2006. 335 с.
231. Юнг К. Г. Психологические типы / под ред. В. Зеленского; пер. С. Лорие. Санкт-Петербург : Азбука, 2001.
232. Аугустинавичюте А. Соционика. Москва : Чёрная белка, 2008. 568 с.

233. Ермак В. Д. Классическая соционика. Системная концепция теории информационного метаболизма психики. Москва : Чёрная белка, 2009. 472 с.

234. Месарович М., Мако Д., Такахара И. Теория иерархических многоуровневых систем. Москва : Мир, 1973. 344 с.

235. Блехман И. И. Синхронизация в природе и технике. Москва : Наука. Главная редакция физико-математической литературы, 1981. 352 с.

236. Моделирование и прогнозирование мировой динамики / В. А. Садовничий, А. А. Акаев, А. В. Коротаев, С. Ю. Малков. Москва : ИСПИ РАН, 2012. 359 с.

237. Турчин П. В. Историческая динамика: На пути к теоретической истории. Пер. с англ. / Под общ. ред. Г. Г. Малинецкого, А. В. Подлазова, С. А. Боринской. Предисл. Г. Г. Малинецкого. Изд. 2-е. Москва : Издательство ЛКИ, 2010. 368 с.

238. Коротаев А. В., Малков А. С., Халтурина Д. А. Законы истории. Математическое моделирование исторических макропроцессов. Демография, экономика, войны. Москва : КомКнига, 2005. 344 с.

239. Howe N., Strauss W. The Fourth Turning: What the Cycles of History Tell Us About America's Next Rendezvous with Destiny. New York : Broadway Books, 1997.

240. Букалов А. В. О четырех эволюционных стадиях развития и законе сменяемости квадр // СМиПЛ. 1995. N 1.

241. Кун Т. Структура наукових революцій. Київ : Port-Royal, 2001. 228 с.

242. Хьюбнер К. Критика научного разума / Пер. с нем. Москва : ИФРАН, 1994. 326 с.

243. Костов С. В. Богданомика: микрокосм А. А. Богданова. 3-е изд., перераб. и доп. Новосибирск: Новосибирский издательский дом, 2015. 784 с.

244. Щедровицкий Г. П. Избранные труды. Москва : Шк.Культ.Полит., 1995. 800 с.

245. Адамар Ж. Исследование психологии процесса изобретения в области математики. Москва : Сов. радио, 1970. 152 с.

246. Тихомиров О. К. Психология мышления. Москва : Изд. центр "Академия", 2008. 288 с.

247. Альтшуллер Г. С., Шапиро Р. Б. О психологии изобретательского творчества // Вопросы психологии. 1956. № 6. С. 37-49.

248. ДСТУ 3278-95. Система розроблення та поставлення продукції на виробництво. Основні терміни і визначення. Введ. 1995–12–27. Київ : Держстандарт України, 1995. 63 с.

249. Р 50.1.031-2001. Информационные технологии поддержки жизненного цикла продукции. Терминологический словарь. Часть 1. Стадии жизненного цикла продукции. Рекомендации по стандартизации. Введ. 2001–06–01. Москва : Изд-во стандартов, 2001. 32 с.

250. ГОСТ Р ИСО/МЭК 15288—2005. Информационная технология. Системная инженерия. Процессы жизненного цикла систем. Введ. 2005–12–29. Москва : Стандартинформ, 2005. 57 с.

251. ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств. Введ. 2010–11–30. Москва : Стандартинформ, 2011. 105 с.

252. Батоврин В. К., Бахтурин Д. А. Управление жизненным циклом технических систем : серия докладов (зеленых книг) в рамках проекта «Промышленный и технологический форсайт Российской Федерации» / ред. И. С. Мацкевич, М. С. Липецкая; Фонд «Центр стратегических разработок «Северо-Запад». Санкт-Петербург, 2012. Вып. 1. 59 с.

253. Systems Engineering Principles and Practice / A. Kossiakoff, W. N. Sweet, S. J. Seymour, S. M. Biemer. 2-е изд. Hoboken, New Jersey: A John Wiley & Sons, 2011. 599 p.

254. Огурцов А. Н. Введение в синергетику : учебное пособие. Харьков : НТУ "ХПИ", 2013. 208 с.

255. Турчин В. Ф. Феномен науки: Кибернетический подход к эволюции. Изд. 2-е. Москва : ЭТС. 2000. 368 с.

256. Уемов А. И. Системный подход и общая теория систем. Москва : Мысль, 1978. 272 с.
257. Эшби У. Р. Введение в кибернетику. Москва : Издательство иностранной литературы, 1959. 432 с.
258. Черняк Ю. И. Системный анализ в управлении экономикой. Москва : Экономика, 1975. 191 с.
259. Волкова В. Н., Денисов А. А. Теория систем : учеб. пособие. Москва : Высш. шк., 2006. 511 с.
260. Растрингин Л. А., Граве П. С. Кибернетика как она есть. Москва : Молодая гвардия, 1975. 208 с.
261. Гаврилов А. В. Гибридные интеллектуальные системы: Монография. Новосибирск : Изд-во НГТУ, 2002. 142 с.
262. Колесников А. В., Кириков И. А., Листопад С. В. Гибридные интеллектуальные системы с самоорганизацией: координация, согласованность, спор. Москва : ИПИ РАН, 2014. 189 с.
263. Wooldridge M. An Introduction to Multiagent Systems. Chichester: John Wiley & Sons Ltd, 2002. 366 p.
264. Shoham Y., Leyton-Brown K. Multiagent systems: algorithmic, game-theoretic, and logical foundations. Cambridge University Press, 2008. 504 p.
265. Kravari K., Bassiliades N. A survey of agent platforms // Journal of Artificial Societies and Social Simulation. 2015. 18(1). P. 11. DOI: 10.18564/jasss.2661.
266. Shoham Y. Agent-oriented programming // Artificial Intelligence. 1993. 60. P. 51-92. DOI: 10.1.1.123.5119.
267. Karasev V. O., Sukhanov V. A. Product lifecycle management using multi-agent systems models // Procedia Computer Science. 2017. 103. P. 142-147. DOI: 10.1016/j.procs.2017.01.034.
268. Fortineau V., Paviot T., Lamouri S. Automated business rules and requirements to enrich product-centric information // Computers in Industry. 2019. 104. P. 22-33. DOI: 10.1016/j.compind.2018.10.001.

269. Van Rossum G., Drake Jr F. L. Python reference manual. Amsterdam : Centrum voor Wiskunde en Informatica Amsterdam, 1995.

270. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд.: Пер. с англ. / Буч Г., Максимчук Р. А., Энгл М. У., Янг Б. Дж., Коналлен Д., Хьюстон К. А. Москва : ООО "И.Д. Вильямс", 2008. 720 с.

271. Hunter J. D. Matplotlib: A 2D graphics environment // Computing in Science & Engineering. 2007. 9(3). P. 90-95.

272. Scikit-learn: machine learning in Python / Pedregosa et al. // JMLR. 2011. 12. P. 2825-2830. URL: <http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html> (Last accessed: 10.02.2019).

273. pyDatalog. URL: <https://sites.google.com/site/pydatalog> (Last accessed: 01.05.2019).

274. McKinney W. Data structures for statistical computing in Python // Proc. of the 9th Python in Science Conf. ed S van der Walt and J Millman, 2010. P. 51-56.

275. Hagberg A. A., Schult D. A., Swart P. J. Exploring network structure, dynamics, and function using NetworkX // Proc. of the 7th Python in Science Conf (SciPy2008) ed Gael Varoquaux, Travis Vaught et al., Pasadena, CA USA, Aug 2008. 2008. P. 11–15.

276. Ettiienne M. B., Vester S., Villadsen J. Implementing a multi-agent system in Python with an auction-based agreement approach // Programming Multi-Agent Systems. ProMAS 2011. Lecture Notes in Computer Science, ed Dennis L, Boissier O et al. vol. 7217. Berlin, Heidelberg : Springer, 2012. P. 185-196. DOI: 10.1007/978-3-642-31915-0_11.

277. osBrain – A general-purpose multi-agent system module written in Python. URL: <https://github.com/opensistemas-hub/osbrain> (Last accessed: 01.05.2019).

278. Python Agent DEvelopment framework. URL: <https://pade.readthedocs.io> (Last accessed: 01.05.2019).

279. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: навчальний посібник. Запоріжжя : ЗНТУ, 2008. 341 с.

280. Рыбина Г. В. Основы построения интеллектуальных систем. Москва : Финансы и статистика, ИНФРА-М, 2010. 432 с.

281. Гаврилова Т. А., Хорошевский В. Ф. Базы знаний интеллектуальных систем: учебник. Санкт-Петербург : Питер, 2000. 384 с.

282. Джарратано Д., Райли Г. Экспертные системы: принципы разработки и программирование; пер. с англ. Москва : ООО «И.Д. Вильямс», 2007. 1152 с.

283. OWL Web Ontology Language. Overview: W3C Recommendation 10 February 2004. W3C. URL: <http://www.w3.org/TR/owl-features> (Last accessed: 01.01.2017).

284. Resource Description Framework (RDF): Concepts and Abstract Syntax: W3C Recommendation 10 February 2004. W3C. URL: <http://www.w3.org/TR/rdf-concepts> (Last accessed: 01.01.2017).

285. Baader F. The Description Logic Handbook. New York : Cambridge University Press, 2003. 573 p.

286. A Practical Guide To Building OWL Ontologies Using The Protege-OWL Plugin and CO-ODE Tools / Matthew Horridge, Holger Knublauch, Alan Rector, Robert Stevens, Chris Wroe. Edition 1.0. Manchester: The University Of Manchester, 2004. URL: <http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf> (Last accessed: 01.01.2017).

287. SPARQL Query Language for RDF: W3C Recommendation 15 January 2008. W3C. URL: <http://www.w3.org/TR/rdf-sparql-query> (Last accessed: 01.01.2017).

288. SWRL: A Semantic Web Rule Language Combining OWL and RuleML: W3C Member Submission 21 May 2004. W3C. URL: <http://www.w3.org/Submission/SWRL> (Last accessed 01.01.2017).

289. Копей В. Б., Семанишин Л. М. Застосування мови програмування Python для побудови баз знань та експертних систем // Восточно-Европейский журнал передовых технологий. 2011. № 6/2(54). С. 62-67.

290. Копей В. Б. Статистичні моделі відмов колон насосних штанг // Матеріали II Міжнародної науково-технічної конференції "Машини, обладнання і матеріали для нарощування вітчизняного видобутку нафти і газу PGE – 2018", м. Івано-Франківськ, 24-27 квітня 2018 р. Івано-Франківськ : ІФНТУНГ, 2018. С. 310-313.

291. Копей В. Б. Прогнозування частоти відмов колон насосних штанг за допомогою ансамблів дерев рішень // Комплексне забезпечення якості технологічних процесів та систем (КЗЯТПС – 2018) : матеріали тез доповідей VIII Міжнародної науково-практичної конференції, м. Чернігів, 10–12 травня 2018 р.: у 2-х т. / Чернігівський національний технологічний університет [та ін.]; відп. за вип.: Єрошенко Андрій Михайлович [та ін.]. Т. 2. Чернігів : ЧНТУ, 2018. С. 198-200.

292. Ивахненко А. Г. Индуктивный метод самоорганизации моделей сложных систем. Киев : Наук. думка, 1981. 296 с.

293. GMDH Shell Documentation. URL: <https://gmdhsoftware.com/docs> (Last accessed: 01.01.2017).

294. Копей В. Б. Імітаційна модель свердловинної штангової насосної установки на основі абстрактних автоматів // Тези доповідей Міжнародної науково-технічної конференції "Нафтогазова енергетика-2017", м. Івано-Франківськ, 15-19 травня 2017 р. Івано-Франківськ : ІФНТУНГ, 2017. С. 365-366.

295. Копей В. Б. Імітаційна модель свердловинної штангової насосної установки на основі абстрактних автоматів // Розвідка та розробка нафтових і газових родовищ. 2017. №3(64). С. 40-49.

296. Пановко Я. Г. Внутреннее трение при колебаниях упругих систем. Москва : Гос. изд. физ.-мат. лит., 1960. 196 с.

297. Вибрации в технике : справочник. В 6-ти т. / Ред. совет: В. Н. Челомей (пред.). Москва : Машиностроение, 1981. Т. 6. Защита от вибрации и ударов / Под ред. К. В. Фролова. 1981. 456 с.

298. Orban F. Damping of materials and members in structures // 5th International Workshop on Multi-Rate Processes and Hysteresis (MURPHYS 2010).

Journal of Physics: Conference Series. 2011. 268. 012022 : IOP Publishing. URL: <http://iopscience.iop.org/1742-6596/268/1/012022>.

299. Рязанцев А. О. Разработка метода виброакустической диагностики глубиннонасосных штанг в процессе эксплуатации : автореф. дис. ... к.т.н. : 05.04.07. УГНТУ. Уфа, 2000. 22 с.

300. Мельникова Е. Ю. Разработка критериев диагностирования насосно-компрессорных труб виброакустическим методом в промышленных условиях : автореф. дис. ... к.т.н. : 05.02.13. УГНТУ. Уфа, 2002. 22 с.

301. Kopey V. Development of model of sucker-rod pumping system by using Maplesim™ software // International scientific and technical conference "Oil and Gas Power Engineering 2013" : abstracts, Ivano-Frankivsk, October 7-11, 2013. Ivano-Frankivsk : IFNTUOG, 2013. P. 116-118.

302. Modelon's Hydraulics Library. URL: <http://www.modelon.com/products/modelica-libraries/hydraulics-library> (Last accessed: 01.01.2017).

303. Архипов К. И., Попов В. И., Попов И. В. Справочник по станкам-качалкам. Альметьевск : ТатАСУнефть, 2000. 146 с.

304. Obrigewitsch M., Lapis T. Early detection of waxy deposits in beam pump wells. URL: <https://autoelect.com/products/oilgas/roc/papers/wax/index.shtml> (Last accessed: 01.01.2017).

305. Якимов С. Б., Клусов А. А., Баринов А. А. Линейный привод ШГН. Первый опыт применения в России // ТЕРРИТОРИЯ НЕФТЕГАЗ. № 8. август 2013. С. 48-54.

306. СОУ 11.1-00135390-010:2011. Організація і процедура управління роботами, які виконують бригади поточного і капітального ремонту свердловин [Чинний від 2011–06–23]. Київ : ПАТ «Укрнафта», 2011. 94 с.

307. Комп'ютерна програма "Модуль для компонентно-орієнтованого моделювання динамічних систем та приклади його застосування для побудови моделей колони насосних штанг" : свідоцтво про реєстрацію авторського права на

твір № 73098 / Копей Володимир Богданович. Дата реєстрації 25.07.17 // Бюл. "Авторське право і суміжні права" № 46/2017. С. 154.

308. SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers / Hindmarsh A. C., Brown P. N., Grant K. E., Lee S. L., Serban R., Shumaker D. E., Woodward C. S. // ACM Transactions on Mathematical Software. 2005. 31(3). P. 363-396. Also available as LLNL technical report UCRL-JP-200037.

309. Petzold L. Description of DASSL: A differential/algebraic system solver. SAND82-8637. 1982.

310. Andersson C, Fuhrer C, Akesson J. Assimulo: A unified framework for ODE solvers // Mathematics and Computers in Simulation. 2015. 116. P. 26-43. DOI: 10.1016/j.matcom.2015.04.007.

311. Копей В. Б. Компонентно-орієнтоване моделювання кінематики механізмів мовою Python на прикладі механізму верстата-гойдалки // Тези доповідей Міжнародної науково-технічної конференції "Нафтогазова енергетика-2017", м. Івано-Франківськ, 15-19 травня 2017 р. Івано-Франківськ : ІФНТУНГ, 2017. С. 362-364.

312. Копей В. Б. Принципи побудови тривимірної параметричної моделі верстата-гойдалки в SolidWorks® 2012 // Всеукраїнський науково-практичний семінар "Графічна освіта у ВНЗ: стан та перспективи" : збірник тез доповідей, м. Івано-Франківськ, 19-20 вересня 2013 р. Івано-Франківськ : ІФНТУНГ, 2013. С. 87-89.

313. Копей В. Б. Розробляння та аналіз параметричних скінченно-елементних моделей різьбових з'єднань в Abaqus® // Нафтогазова енергетика. 2010. № 1(12). С. 31-36.

314. Копей В. Б. Скінченно-елементний аналіз та оптимізація різьбових з'єднань // Тези доповідей XI Міжнародної науково-технічної конференції "Прогресивна техніка і технологія - 2010", м. Київ, 18-21 травня 2010 р. Київ : НТУУ "КПІ", 2010. С. 99.

315. Копей В. Б. Скінченно-елементний аналіз та оптимізація різьбових з'єднань // Вісник Національного технічного університету України "Київський політехнічний інститут". Серія машинобудування. 2010. № 58. С. 42-47.

316. Копей В. Б. Науково-технічна розробка "Сценарії Abaqus/CAE для побудови та аналізу скінченно-елементних моделей різьбових з'єднань" // Інтелектуальний продукт вчених, винахідників і раціоналізаторів Прикарпаття, 2011 : каталог перспективних винаходів, корисних моделей, промислових зразків і раціоналізаторських пропозицій "Галицьких кмітливців" / Довід. вид. Редакційна колегія: В. В. Попович, Б. І. Середюк, Л. М. Шляхтич, Т. В. Тиховська, В. М. Когуч. Івано-Франківськ : Обласна організація товариства винахідників і раціоналізаторів України, 2012. С. 20-21.

317. Биргер И. А., Мавлютов Р. Р. Соппротивление материалов: Учебное пособие. Москва : Наука, 1986. 560 с.

318. FreeCAD Documentation. URL: http://www.freecadweb.org/wiki/Main_Page (Last accessed 01.01.2019).

319. Копей В. В., Михайлюк В. В., Копей В. Б. Моделювання різьб насосних штанг методом скінченних елементів // Анотації міжнародної науково-практичної конференції молодих вчених "Техніка і прогресивні технології у нафтогазовій інженерії", м. Івано-Франківськ, 16-20 вересня 2008 р. Івано-Франківськ : Факел, 2008. С. 17.

320. Копей В. Б. Дослідження впливу геометричних параметрів муфтового різьбового з'єднання насосних штанг на напруження у впадинах різьби ніпеля // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2010. № 2(24). С. 81-85.

321. Якухин В. Г., Ставров В. А. Изготовление резьбы: Справочник. Москва : Машиностроение, 1989. 192 с.

322. Копей В. В., Михайлюк В. В., Копей В. Б. Оптимізація затягнення різьб у процесі згвинчування насосних штанг // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2008. № 2(18). С. 32-39.

323. Копей В. Скінченно-елементне моделювання та оптимізація параметрів муфтового різьбового з'єднання насосних штанг за критерієм втомної міцності // Комп'ютерні технології: наука і освіта : тези доповідей V Всеукраїнської науково-практичної конференції, м. Івано-Франківськ, 29.09-3.10.2010 р. Київ : Університет "Україна", 2010. С. 109-112.

324. Інструкція по розрахунку моментів згвинчування насосних штанг : Керівний документ / Б. В. Копей, В. В. Михайлюк, А. П. Джус, В. Б. Копей. Івано-Франківськ: ІФНТУНГ, 2009. 18 с.

325. Копей Б. В., Копей В. Б., Лисканич М. В. Моделювання вібрацій замкового різьбового з'єднання методом кінцевих елементів // Збірник праць 3-ї Міжнародної науково-технічної конференції (12-19 грудня 2004 р. - Хургада, Єгипет). Хмельницький : ХНУ, 2004. С. 71-75.

326. Kopey V., Kopey V., Lyskanych M. Tool-joint thread modeling by finite element method // *Wiernictwo Nafta Gaz*. 2005. r.22/1. P. 201-204.

327. Kopey V., Kopey V. Finite element method of tool-joint thread simulation // *Recueil des resumes. 3rd International symposium on hydrocarbons and chemistry*, 27-29 March, 2006, Ghardaïa, Algeria. Boumerdes : University of Boumerdes, 2006. P. 54-55.

328. Саркисов Г. М. Расчет бурильных и обсадных колонн. Москва : Недра, 1971. 205 с.

329. Abaqus 6.14 analysis user's guide. Volume 2: analysis. Providence, RI : Dassault Systems Simulia Corp., 2014. 1489 p.

330. Курушин М. И., Курушин А. М., Барманов И. С. Демпфирование вибраций изделий силами трения в резьбовых соединениях // *Известия Самарского научного центра Российской академии наук*. 2012. Т. 14, №6. С. 70-76.

331. Копей В. Б. Автоматизоване проектування з'єднання тіла склопластикової насосної штанги зі сталевую головкою // *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. 2011. №5. С. 142-147.

332. Копей В. Б. Моделювання гармонічного осьового навантажування пресового з'єднання склопластикової насосної штанги // *Соціально-економічний*

розвиток в умовах глобалізації : матеріали XLIX Міжнародної науково-практичної конференції, м. Чернівці, 29-30 листопада 2016 р. Т. 1. Київ : Науково-видавничий центр "Лабораторія думки", 2016. С. 7-10.

333. Розробка наукових основ створення з'єднань з металополімерних композитних матеріалів та керування їх зносо-фрикційними та втомними властивостями : Звіт про НДР (заключний) / Д. О. Вольченко, Б. В. Копей, О. І. Вольченко та ін. / ІФНТУНГ. 2017. 115 с. Д-4-15Ф. № держреєстрації 0115U002279.

334. Копей В. Б. Моделювання втомної міцності ніпеля склопластикової насосної штанги // Матеріали II-ї Міжнародної науково-практичної конференції "Теоретичні та практичні аспекти розвитку науки", м. Київ, 29 – 30 листопада 2016 р. Ч. 2. Київ : МЦНД, 2016. С. 22-24.

335. Метод зміцнення насосних штанг полімерною стрічкою / Копей Б. В., Онищук О. О., Онищук С. Ю., Копей В. Б. // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2009. № 1(19). С. 92-97.

336. Копей Б. В., Стеліга І. І., Копей В. Б. Комплексне зміцнення насосних штанг металізаційними покриттями і склопластиковою ізоляцією // Анотації міжнародної науково-практичної конференції молодих вчених "Техніка і прогресивні технології у нафтогазовій інженерії", м. Івано-Франківськ, 16-20 вересня 2008 р. Івано-Франківськ : Факел, 2008. С. 23.

337. Найда А. М., Копей Б. В., Копей В. Б. Експериментальна оцінка ефективності бандажів для зміцнення пошкоджених трубопроводів // Анотації міжнародної науково-практичної конференції молодих вчених "Техніка і прогресивні технології у нафтогазовій інженерії", м. Івано-Франківськ, 16-20 вересня 2008 р. Івано-Франківськ : Факел, 2008. С. 36.

338. Копей Б. В., Копей В. Б., Найда А. М. Скінчено-елементний аналіз зміцнення труби з експлуатаційними дефектами склопластиковим бандажем // Матеріали Міжнародної конференції "Підвищення якості, надійності та

довговічності технічних систем і технологічних процесів", м. Шарм Ель Шейх (Єгипет), 4-11 грудня 2005р. Хмельницький : ХНУ, 2005. С. 33-39.

339. Копей Б. В., Копей В. Б. Методика визначення товщини бандажів композитних підсилювальних при ремонті трубопроводів з експлуатаційними дефектами. Керівний документ КД – 02070855. 003. Івано-Франківськ : ІФНТУНГ, 2005. 14 с.

340. Пат. UA 60506A, МПК F16L57/00,58/02. Спосіб підвищення ресурсу, міцності і довговічності трубопроводу / Копей Б. В., Максимук О. В., Щербина Н. М., Копей В. Б., Стеліга І. І. № 2002118792; заявл. 06.11.2002; опубл. 15.10.2003, Бюл. №10.

341. Справочник по коэффициентам интенсивности напряжений / Под ред. Ю. Мураками. Москва : Мир, 1990. Т. 1, 2. 1013 с.

342. Courtney T. Mechanical Behavior of Materials. Long Grove, IL : Waveland Press, 2005. 733 p. ISBN 1-57766-425-6.

343. Intra-Laminar Fracture Toughness of Glass Fiber Reinforced Polymer By Using Theory, Experimentation and FEA / P. Firojkhani, K. Tanpure, A. Dawale, S. Patil // IOP Conf. Ser.: Mater. Sci. Eng. 2018. vol.346. 012070. DOI: 10.1088/1757-899X/346/1/012070.

344. Mandell J. F., McGarry F. J. Fracture behavior of fiberglass reinforced plastics suitable for hull materials. Report (Massachusetts Institute of Technology. Sea Grant Program); no. MITSG 75-25. 1976. URL: <https://repository.library.noaa.gov/view/noaa/9615> (Last accessed: 10.02.2019).

345. Ратич Л. В., Федорович Я. Т. Циклическая коррозионная трещиностойкость материалов и долговечность насосных штанг // ФХММ. 1988. № 6. С. 95-100.

346. Daoud O. E. K., Cartwright D. J., Carney M. Strain-energy release rate for a single-edge-cracked circular bar in tension // J. of Strain Analysis. 1978. 13, № 2. P. 83-89.

347. Методика оценки вязкости разрушения круглого проката и высокопрочной проволоки / О. Н. Романив, Л. П. Лазько, И. Н. Панько, Р. В. Ризничук // ФХММ. 1981. № 6. С. 64-68.

348. Paris P. C., Sih G. C. Stress Analysis of Cracks // Fracture Toughness and Testing and its Applications. STP 381. Philadelphia : American Society for Testing and Materials, 1965. P. 30-81. DOI: <https://doi.org/10.1520/STP26584S>.

349. Копей Б. В., Стеліга І. І., Копей В. Б. Втомні характеристики склопластикових та зміцнених склопластиковою ізоляцією сталевих насосних штанг при асиметричному навантаженні // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2009. № 3(21). С. 73-78.

350. Гальмування росту тріщин в насосних штангах полімерними та металополімерними композитними покриттями / Копей Б. В., Онищук С. Ю., Стеліга І. І., Копей В. Б. // Анотації Міжнародної науково-технічної конференції "Нафтогазова енергетика: проблеми і перспективи". Івано-Франківськ, 20-23 жовтня 2009 р. Івано-Франківськ : ІФНТУНГ, 2009. С. 66.

351. Копей В. Б. Скінченно-елементне моделювання та аналіз втомної міцності насосних штанг в зоні скруглення між піделеваторним буртом і тілом штанги // Тези доповідей міжнародної науково-практичної конференції молодих учених та студентів "Техніка і прогресивні технології у нафтогазовій інженерії - 2012", м. Івано-Франківськ, 5-7 листопада 2012 р. Івано-Франківськ : ІФНТУНГ, 2012. С. 123-126.

352. Копей В. Б., Петрина Ю. Д. Протектор для насосних штанг // Інтелектуальний продукт вчених і винахідників Прикарпаття - 2013 : щорічний каталог найвагоміших винаходів, корисних моделей, промислових зразків і раціоналізаторських пропозицій Галицьких Кмітливців. Довід. вид. / Редакційна колегія: Б. І. Середюк, Л. М. Шляхтич, Т. В. Тиховська, В. М. Когуч. Івано-Франківськ : ПП Корольчук, 2014. С. 35-36.

353. Рейнин Г. Тайны типа. Модели. Группы. Признаки. Москва : Чёрная белка, 2009. 304 с.

354. Организация систем управления созданием и развитием технической продукции : методические рекомендации / М. М. Четвертаков и др. Ленинград : ЦНИИ «Румб», 1981. 96 с.

355. Juran's Quality Control Handbook, 4th ed. / Juran J.M., Gryna F.M. (eds). New York : McGraw-Hill, 1988. 1774 p.

356. Урманцев Ю. А. Симметрия природы и природа симметрии (Философские и естественнонаучные аспекты). Москва : Мысль, 1974. 229 с.

357. Копей В. Б. Алгоритм інтелектуальної системи на основі міждисциплінарних досліджень загальносистемних закономірностей // Комп'ютерне моделювання та оптимізація складних систем (КМОСС-2018) : матеріали IV Міжнародної науково-технічної конференції, м. Дніпро, 1-2 листопада 2018 р. / Міністерство освіти і науки України, Державний вищий навчальний заклад "Український державний хіміко-технологічний університет". Дніпро : Баланс-клуб, 2018. С. 246-248.

358. Якушев А. И., Мустаев Р. Х., Мавлютов Р. Р. Повышение прочности и надежности резьбовых соединений. Москва : Машиностроение, 1979. 215 с.

359. Мочернюк Д. Ю. Исследование и расчет резьбовых соединений труб, применяемых в нефтедобывающей промышленности. Москва : Недра, 1970. 137 с.

360. Ковалев С. Ф. Герметичность и прочность конических резьбовых соединений труб нефтяного сортамента. Москва : Недра, 1965. 170 с.

361. Билык С. Ф. Герметичность и прочность конических резьбовых соединений труб нефтяного сортамента. Москва : Недра, 1981. 237 с.

362. Копей В. Б. Створення експертної системи з проблем надійності і довговічності різьбових з'єднань в редакторі онтологій Protege-OWL 3.5 // Збірка наукових праць за матеріалами Міжнародної науково-практичної конференції "Наукові дослідження сучасності. Випуск 4", м. Київ, 30 травня 2012 р. Частина 1. Київ : НАІРІ, 2012. С. 99-101.

363. Копей В. Б. Створення експертної системи з проблем надійності і довговічності різьбових з'єднань мовою Python // Збірка наукових праць за

матеріалами Міжнародної наукової конференції "Наука - XXI століття. Випуск 2", м. Київ, 27 червня 2012 р. Київ : НАІРІ, 2012. С. 117-122.

364. Комп'ютерна програма "Програма для побудови баз знань з проблем надійності і довговічності штангових свердловинних насосних установок" ("SuckerRodPumpingUnitKB") : свідоцтво про реєстрацію авторського права на твір № 42157 / Копей Володимир Богданович. Дата реєстрації 08.02.2012 // Каталог державної реєстрації. Вип.16/2012. С. 42.

365. Копей В. Б. Експертна система з проблем надійності та довговічності різьбових з'єднань // Інтелектуальний продукт вчених, винахідників і раціоналізаторів Прикарпаття, 2012 : каталог перспективних винаходів, корисних моделей, промислових зразків і раціоналізаторських пропозицій Галицьких кмітливців / Довід. вид. Редакційна колегія: В. В. Попович, В. П. Петренко, Б. І. Середюк, Л. М. Шляхтич, Т. В. Тиховська, В. М. Когуч. Івано-Франківськ : Обласна організація товариства винахідників і раціоналізаторів України, 2013. С. 34-36.

366. Копей В. Б., Шпук Т. В. Принципи побудови мовою Python програми-аутлайнера з можливостями експертної системи // Тези доповідей ІХ Всеукраїнської наукової конференції студентів і молодих вчених "Молодь: освіта, наука, духовність". Ч. 3. Київ : Університет «Україна», 2012. С. 225-227.

367. Building a framework for predictive science / McKerns M. M., Strand L., Sullivan T., Fang A., Aivazis M. A. G. // Proc. of the 10th Python in Science Conf. 2011. URL: <http://arxiv.org/pdf/1202.1056> (Last accessed: 10.02.2019).

368. Копей В. Б. Компонент для декларативного програмування в Python // Інформаційні та моделюючі технології / Матеріали Всеукраїнської науково-практичної конференції ІМТ-2015. Черкаси : ПП Нечитайло О.Ф., 2015. С. 23.

369. Копей В. Б. Простий генератор інтерактивних документів Jupyter Notebook з розміченого програмного коду Python // Вісник Університету "Україна", серія "Інформатика, обчислювальна техніка та кібернетика". 2019. №1(22). С. 161-165. DOI: 10.36994/2707-4110-2019-1-22-31.

370. Копей В. Б. Інтерфейс з Datalog Educational System для мови програмування Python // Проблеми і перспективи розвитку науки в умовах євроінтеграції : матеріали XX Міжнародної науково-практичної конференції "Проблеми і перспективи розвитку науки в умовах євроінтеграції", Чернівці, 29-30 апреля 2015 р. Т. 1. Київ : Науково-видавничий центр "Лабораторія думки", 2015. С. 10-12.

371. The PageRank citation ranking: bringing order to the Web. Technical Report. / Page L., Brin S., Motwani R., Winograd T. Stanford InfoLab, 1999. 17 p. URL: <http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf> (Last accessed: 10.02.2019).

372. Gansner E. R., North S. C. An open graph visualization system and its applications to software engineering // Software-practice and experience. 2000. 30(11). P. 1203-1233.

373. Investigation of the contact stresses on the end faces of the drill pipe connection that include the tapered thread manufactured by the cutter lathe tool with the specially modified cutting edge / Onysko O., Medvid I., Kopey V., Panchuk V., Voitenko P. // International Conference of Applied Science ICAS 2019 : Book of Abstracts, May 9-11, 2019, Hunedoara, Romania. Timisoara : Editura EUROBIT, 2019. P. 42-43.

374. Яворський М. В., Копей В. Б. Застосування систем тривимірного параметричного моделювання для розмірного аналізу // Молода наука. Технологія машинобудування : збірник наукових праць Всеукраїнської науково-технічної конференції студентів і молодих вчених / за заг. ред. С. В. Ковалевського, д-ра техн. наук, проф. Краматорськ : ДДМА, 2016. С. 182-184.

375. Гаврильців С. Я., Копей В. Б. Принципи побудови САПР зубчатих коліс та зуборізних інструментів в середовищі SolidWorks® // Матеріали міжвузівської науково-практичної конференції молодих вчених і студентів "Нові орбіти сучасного машинобудування. Питання новітніх технологій та обладнання в машинобудуванні", м. Кривий Ріг, 29 квітня 2008 р. Кривий Ріг : Видавничий центр КТУ, 2008. С. 78-79.

376. Копей В. Б. Розробка програмних компонентів мовою Python та їх використання в IPython Notebook // Вісник Університету "Україна", серія "Інформатика, обчислювальна техніка та кібернетика". 2017. №1(19). С. 208-214.

377. Копей В. Б., Яремчук А. В. Застосування мови програмування Python для автоматизованого добування знань з онлайн-баз патентів // Наукова Україна. Збірник матеріалів Всеукраїнської студентської наукової конференції з міжнародною участю, 25 травня 2015 р. Дніпропетровськ : «SeKum Software», 2015. С. 194-197.

378. Kopei V. B., Onysko O. R., Panchuk V. G. Principles of development of product lifecycle management system for threaded connections based on the Python programming language // J. Phys.: Conf. Ser. 2020. 1426. DOI: 10.1088/1742-6596/1426/1/012033.

Івано-Франківський національний технічний університет нафти і газу
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

Копей Володимир Богданович

УДК 622.276.054:[004.89+004.94]

ДИСЕРТАЦІЯ

Науково-методологічні основи автоматизованого проектування обладнання
штангової свердловинної насосної установки


05.05.12 – машини нафтової та газової промисловості

Частина 2

Додатки

Подається на здобуття наукового ступеня доктора технічних наук

Дисертація містить результати власних досліджень. Використання ідей,
результатів і текстів інших авторів мають посилання на відповідне джерело

 В. Б. Копей

Науковий консультант Петрина Юрій Дмитрович, д-р техн. наук, професор

ЗМІСТ

	с.
ДОДАТОК А Список публікацій здобувача.....	432
ДОДАТОК Б Статистичні моделі відмов штангових колон	441
ДОДАТОК В Модель ШСНУ на основі абстрактних автоматів	451
ДОДАТОК Г Модель ШСНУ мовою Modelica	473
ДОДАТОК Д Графіки циклічних навантажень на колони ШН і НКТ	485
ДОДАТОК Е Пакет для моделювання кулькового клапана методами обчислювальної гідродинаміки.....	487
ДОДАТОК Є ruscodyn – пакет для компонентно-орієнтованого акаузального моделювання мовою Python	490
ДОДАТОК Ж Компонентно-орієнтована модель верстата-качалки для симуляції кінематики	509
ДОДАТОК З Python-класи для перебудови параметричної моделі верстата- качалки у SOLIDWORKS	512
ДОДАТОК И Моделі РЗ для Abaqus/CAE	523
ДОДАТОК І Пакет ThreadsOSS – прикладна САПР різьбових з'єднань	567
ДОДАТОК Й Пакет для геометричного моделювання різьб з відхиленнями за допомогою FreeCAD API.....	590
ДОДАТОК К Результати моделювання замкового РЗ	597
ДОДАТОК Л Результати моделювання РЗ гладких НКТ умовним діаметром 114 мм (ГОСТ 633-80) з пружною моделлю матеріалу	599
ДОДАТОК М Input-файли Abaqus для відтворення гармонічного і динамічного аналізу	601
ДОДАТОК Н VBA-макрос для обчислення коефіцієнта запасу втомної міцності за критерієм Сайнса в SOLIDWORKS Simulation	603
ДОДАТОК О Варіанти конструкції пресового з'єднання полімерного стержня зі сталевою оболонкою	605
ДОДАТОК П Макрос Abaqus/CAE для розрахунку з'єднання полімерного стержня зі сталевою оболонкою	606

ДОДАТОК Р Система автоматизованого проектування металополімерних з'єднань на основі вільного програмного забезпечення	612
ДОДАТОК С Програма для обчислення КІН за результатами моделювання МСЕ	621
ДОДАТОК Т Макрос Abaqus/CAE для оптимізації конструкції ШН за критерієм втомної міцності з fe-safe	626
ДОДАТОК У Засоби оптимізації конструкції протектора	632
ДОДАТОК Ф Абстрактна модель ІС	636
ДОДАТОК Х Код експертної системи з проблем надійності та довговічності різьбових з'єднань.....	640
ДОДАТОК Ц Приклад PLM системи різьбових з'єднань.....	661
ДОДАТОК Ч Акти впровадження результатів роботи	680

ДОДАТОК А

Список публікацій здобувача

Праці, в яких опубліковані основні наукові результати дисертації

1. Experimental study of the reinforcement of damaged steel pipe by composite bandage / Kopey B., Rozgonjuk V., **Kopey V.**, Maksymuk O., Scherbina N., Nayda A. // *Wiertnictwo Nafta Gaz*. 2004. r.21/1. P. 125-134.
2. Finite-element analysis of the tubing thread / Kopey B., **Kopey V.**, Bebnarz S., Savula S. // *Wiertnictwo Nafta Gaz*. 2006. r.23/2. P. 681-685.
3. Оптимізація товщини композитних бандажів при ремонті трубопроводів з дефектами / Копей Б. В., **Копей В. Б.**, Максимук О. В., Щербина Н. М., Найда А. М. // *Науковий вісник Івано-Франківського національного технічного університету нафти і газу*. 2007. № 2(16). С. 101-107.
4. Копей Б. В., Зінченко Ю. С., **Копей В. Б.** Аналіз поломок насосних штанг в промислових умовах // *Науковий вісник Івано-Франківського національного технічного університету нафти і газу*. 2008. № 2(18). С. 49-56.
5. **Копей В. Б.** Аналіз способів підвищення ресурсу муфтового різьбового з'єднання насосних штанг // *Розвідка та розробка нафтових і газових родовищ*. 2008. № 4(29). С. 66-72.
6. Копей Б. В., Кузьмін О. О., **Копей В. Б.** Механічні методи зняття відкладень парафіну та асфальто-смолистих речовин з поверхні свердловинного обладнання // *Нафтогазова енергетика*. 2008. № 3(8). С. 10-14.
7. Сучасні методи боротьби з корозією глибинного обладнання ШСНУ / Копей Б. В., Онищук О. О., Онищук С. Ю., **Копей В. Б.** // *Нафтогазова енергетика*. 2008. № 2(7). С. 13-16.
8. **Копей В. Б.** Застосування системи CAD/FEA для розрахунку і оптимізації різьбових з'єднань нафтогазового обладнання // *Розвідка та розробка нафтових і газових родовищ*. 2009. № 3(32). С. 43-49.

9. Копей Б. В., Михайлюк В. В., **Копей В. Б.** Моделювання різьб насосних штанг методом скінченних елементів // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2009. № 2(20). С. 61-67.

10. Копей Б. В., Кузьмін О. О., **Копей В. Б.** Розроблення з'єднань склопластикових порожнистих насосних штанг та визначення навантажень на них // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2010. № 1(23). С. 77-83.

11. **Копей В. Б.** Розробляння та аналіз параметричних скінченно-елементних моделей різьбових з'єднань в Abaqus® // Нафтогазова енергетика. 2010. № 1(12). С. 31-36.

12. **Копей В. Б.** Скінченно-елементний аналіз та оптимізація різьбових з'єднань // Вісник Національного технічного університету України "Київський політехнічний інститут". Серія машинобудування. 2010. № 58. С. 42-47.

13. **Копей В. Б.** Дослідження впливу геометричних параметрів муфтового різьбового з'єднання насосних штанг на напруження у впадинах різьби ніпеля // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2010. № 2(24). С. 81-85.

14. **Копей В. Б.**, Петрина Ю. Д. Принципи розробки бази знань з проблем надійності і довговічності різьбових з'єднань // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2010. № 4(26). С. 66-69.

15. **Копей В. Б.** Автоматизоване проектування з'єднання тіла склопластикової насосної штанги зі сталеву головою // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2011. №5. С. 142-147.

16. Використання явища резонансу для комплектування колони насосних штанг / Олійник А. П., Копей Б. В., Зінченко Ю. С., **Копей В. Б.** // Розвідка та розробка нафтових і газових родовищ. 2011. №1 (38). С. 69-75.

17. **Копей В. Б.**, Палійчук І. І. Застосування мови програмування Python для побудови баз знань з проблем надійності і довговічності штангових

свердловинних насосних установок // Нафтогазова енергетика. 2011. №2(15). С. 12-18.

18. **Копей В. Б.** Імітаційна модель свердловинної штангової насосної установки на основі абстрактних автоматів // Розвідка та розробка нафтових і газових родовищ. 2017. №3(64). С. 40-49.

19. **Копей В. Б.,** Копей Б. В., Кузьмін О. О. Принципи побудови моделі свердловинної штангової насосної установки для середовища Maplesoft MapleSim 7 // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2017. №2(43). С. 42-52.

20. **Копей В. Б.** Абстрактна модель інформаційної системи підтримки життєвого циклу виробу // Прикарпатський вісник НТШ. Число. 2017. №2(38). С. 71-96.

21. **Kopei V. B.,** Onysko O. R., Panchuk V. G. Computerized system based on FreeCAD for geometric simulation of the oil and gas equipment thread turning // IOP Conf. Ser.: Mater. Sci. Eng. 2019. 477:012032. DOI: 10.1088/1757-899X/477/1/012032. (*Scopus*)

22. **Kopei V. B.,** Onysko O. R., Panchuk V. G. Component-oriented acausal modeling of the dynamical systems in Python language on the example of the model of the sucker rod string // PeerJ Computer Science. 2019. 5:e227. DOI: 10.7717/peerj-cs.227. (*Scopus, Q1*)

23. **Копей В. Б.,** Онисько О. Р., Жигуц Ю. Ю. Обґрунтування застосування двоопорних різьбових з'єднань пустотілих насосних штанг // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2019. №1(46). С. 7-15. DOI: 10.31471/1993-9965-2019-1(46)-7-15.

24. **Kopei V.,** Onysko O., Panchuk V. The application of the uncorrected tool with a negative rake angle for tapered thread turning // Ivanov V. et al. (eds) Advances in Design, Simulation and Manufacturing II. DSMIE 2019. Lecture Notes in Mechanical Engineering. Cham : Springer, 2020. P. 149-158. DOI: 10.1007/978-3-030-22365-6_15. (*Scopus*)

25. **Kopei V. B.**, Onysko O. R., Panchuk V. G. Principles of development of product lifecycle management system for threaded connections based on the Python programming language // J. Phys.: Conf. Ser. 2020. 1426:12033. DOI: 10.1088/1742-6596/1426/1/012033. (*Scopus*)

26. Onysko O. R., **Kopey V. B.**, Panchuk V. G. Theoretical investigation of the tapered thread joint surface contact pressure in the dependence on the profile and the geometric parameters of the threading turning tool // IOP Conf. Ser.: Mater. Sci. Eng. 2020. 749:012007. DOI: 10.1088/1757-899X/749/1/012007. (*Scopus*)

Праці, які засвідчують апробацію матеріалів дисертації

27. **Копей В. Б.** Аналіз способів підвищення ресурсу муфтового різьбового з'єднання насосних штанг // Анотації Міжнародної науково-практичної конференції молодих вчених "Техніка і прогресивні технології у нафтогазовій інженерії", м. Івано-Франківськ, 16-20 вересня 2008 р. Івано-Франківськ : Факел, 2008. С. 55. (*форма участі – очна*)

28. Скінчено-елементний аналіз насосних штанг з зарізьбовими канавками / Копей В. В., **Копей В. Б.**, Петрина Ю. Д, Михайлюк В. В. // Анотації Міжнародної науково-технічної конференції "Нафтогазова енергетика: проблеми і перспективи". м. Івано-Франківськ, 20-23 жовтня 2009 р. Івано-Франківськ : ІФНТУНГ, 2009. С. 67. (*форма участі – очна*)

29. **Копей В. Б.**, Панчук А. Г. Дослідження залежності напружень в муфтовому різьбовому з'єднанні насосних штанг від характеристик матеріалів деталей з'єднання // Сучасні технології в промисловому виробництві : матеріали Всеукраїнської міжвузівської науково-технічної конференції, м. Суми, 19-23 квітня 2010 р. Ч. II. Суми : Вид-во СумДУ, 2010. С. 122-123. (*форма участі – заочна*)

30. **Копей В. Б.** Скінченно-елементний аналіз та оптимізація різьбових з'єднань // Тези доповідей XI Міжнародної науково-технічної конференції "Прогресивна техніка і технологія - 2010", м. Київ, 18-21 травня 2010 р. Київ : НТУУ "КПІ", 2010. С. 99. (*форма участі – очна*)

31. **Копей В.** Скінченно-елементне моделювання та оптимізація параметрів муфтового різьбового з'єднання насосних штанг за критерієм втомної міцності // Комп'ютерні технології: наука і освіта : тези доповідей V Всеукраїнської науково-практичної конференції, м. Івано-Франківськ, 29.09-3.10.2010 р. Київ : Університет "Україна", 2010. С. 109-112. *(форма участі – очна)*

32. **Копей В. Б.,** Панчук А. Г. Оптимізація параметрів з'єднання тіла склопластикової насосної штанги зі сталевую головкою // Инновационные технологии в машиностроении : материалы Международной научно-практической конференции, г. Запорожье, 17-21 мая 2011 г. Том 2. Запорожье : Запорожская торгово-промышленная палата, 2011. С. 58-60. *(форма участі – заочна)*

33. **Копей В. Б.,** Венгрынюк Т. П. Моделирование дефектов труб в SolidWorks® // Надежность и безопасность магистрального трубопроводного транспорта : материалы VII Междунар. науч.-техн. конф., Новополоцк, 22 – 25 ноября 2011 г. / под общ. ред. д-ра техн. наук, проф. В. К. Липского. Новополоцк : Полоц. гос. ун-т, 2011. С. 250-251. *(форма участі – заочна)*

34. **Копей В. Б.,** Петрина Ю. Д., Венгрынюк Т. П. Конечно-элементное моделирование ремонта труб с дефектами стеклопластиковыми бандажами в SolidWorks® // Надежность и безопасность магистрального трубопроводного транспорта : материалы VII Междунар. науч.-техн. конф., Новополоцк, 22 – 25 ноября 2011 г. / под общ. ред. д-ра техн. наук, проф. В. К. Липского. Новополоцк : Полоц. гос. ун-т, 2011. С. 248-250. *(форма участі – заочна)*

35. **Копей В. Б.** Створення експертної системи з проблем надійності і довговічності різьбових з'єднань в редакторі онтологій Protégé-OWL 3.5 // Збірка наукових праць за матеріалами Міжнародної науково-практичної конференції "Наукові дослідження сучасності. Випуск 4", м. Київ, 30 травня 2012 р. Частина 1. Київ : НАІРІ, 2012. С. 99-101. *(форма участі – заочна)*

36. **Копей В. Б.** Створення експертної системи з проблем надійності і довговічності різьбових з'єднань мовою Python // Збірка наукових праць за матеріалами Міжнародної наукової конференції "Наука - XXI століття. Випуск 2", м. Київ, 27 червня 2012 р. Київ : НАІРІ, 2012. С. 117-122. *(форма участі – заочна)*

37. **Копей В. Б.** Скінченно-елементне моделювання та аналіз втомної міцності насосних штанг в зоні скруглення між піделеваторним буртом і тілом штанги // Тези доповідей Міжнародної науково-практичної конференції молодих учених та студентів "Техніка і прогресивні технології у нафтогазовій інженерії - 2012", м. Івано-Франківськ, 5-7 листопада 2012 р. Івано-Франківськ : ІФНТУНГ, 2012. С. 123-126. (*форма участі – очна*)

38. **Копей В. Б.** Обґрунтування доцільності збільшення довжини розвантажувальної канавки ніпеля насосної штанги // Технологічний аудит та резерви виробництва (Спецвипуск. Матеріали науково-практичної конференції "Наукові підсумки 2012р.", м. Харків, 2012 р.). 2012. № 6/2 (8). С. 7-8. (*форма участі – заочна*)

39. **Копей В. Б.** Моделювання клапана свердловинного штангового насоса методом обчислювальної гідродинаміки в Abaqus/CAE® // Матеріали Міжнародної науково-технічної конференції "Математичне моделювання прикладних задач математики, фізики, механіки ММАР-2013", м. Харків, 5 – 25 травня 2013 р. Харків : ХНАДУ, 2013. URL: <http://files.khadi.kharkov.ua/images/Fizika.pdf> (дата звернення: 29.02.2016). (*форма участі – заочна*)

40. **Копей В.** Development of model of sucker-rod pumping system by using Maplesim™ software // International scientific and technical conference "Oil and Gas Power Engineering 2013" : abstracts, Ivano-Frankivsk, October 7-11, 2013. Ivano-Frankivsk : IFNTUOG, 2013. P. 116-118. (*форма участі – очна*)

41. **Копей В. Б.** Принципи побудови тривимірної параметричної моделі верстата-гойдалки в SolidWorks® 2012 // Всеукраїнський науково-практичний семінар "Графічна освіта у ВНЗ: стан та перспективи" : збірник тез доповідей, м. Івано-Франківськ, 19-20 вересня 2013 р. Івано-Франківськ : ІФНТУНГ, 2013. С. 87-89. (*форма участі – очна*)

42. **Копей В. Б.** Використання вільного програмного забезпечення для розробки системи скінченно-елементного аналізу різьбових з'єднань нафтогазового обладнання // Матеріали Міжнародної науково-технічної

конференції "Машини, обладнання і матеріали для нарощування вітчизняного видобутку та диверсифікації постачання нафти і газу" ІІМ – 2016, м. Івано-Франківськ, 16-20 травня 2016 р. Івано-Франківськ : ІФНТУНГ, 2016. С. 277-280. *(форма участі – очна)*

43. **Копей В. Б.** Моделювання гармонічного осьового навантажування пресового з'єднання склопластикової насосної штанги // Соціально-економічний розвиток в умовах глобалізації : матеріали XLIX Міжнародної науково-практичної конференції, м. Чернівці, 29-30 листопада 2016 р. Т. 1. Київ : Науково-видавничий центр "Лабораторія думки", 2016. С. 7-10. *(форма участі – заочна)*

44. **Копей В. Б.** Моделювання втомної міцності ніпеля склопластикової насосної штанги // Матеріали II-ї Міжнародної науково-практичної конференції "Теоретичні та практичні аспекти розвитку науки", м. Київ, 29 – 30 листопада 2016 р. Ч. 2. Київ : МЦНД, 2016. С. 22-24. *(форма участі – заочна)*

45. **Копей В. Б.** Компонентно-орієнтоване моделювання кінематики механізмів мовою Python на прикладі механізму верстата-гойдалки // Тези доповідей Міжнародної науково-технічної конференції "Нафтогазова енергетика-2017", м. Івано-Франківськ, 15-19 травня 2017 р. Івано-Франківськ : ІФНТУНГ, 2017. С. 362-364. *(форма участі – очна)*

46. **Копей В. Б., Копей Б. В., Кузьмін О. О.** Принципи побудови моделі свердловинної штангової насосної установки для середовища Maplesoft MapleSim 7 // Тези доповідей Міжнародної науково-технічної конференції "Нафтогазова енергетика-2017", м. Івано-Франківськ, 15-19 травня 2017 р. Івано-Франківськ : ІФНТУНГ, 2017. С. 364-365. *(форма участі – очна)*

47. **Копей В. Б.** Імітаційна модель свердловинної штангової насосної установки на основі абстрактних автоматів // Тези доповідей Міжнародної науково-технічної конференції "Нафтогазова енергетика-2017", м. Івано-Франківськ, 15-19 травня 2017 р. Івано-Франківськ : ІФНТУНГ, 2017. С. 365-366. *(форма участі – очна)*

48. **Копей В. Б., Воробець М. І.** Система автоматизованого проектування металополімерних з'єднань на основі вільного програмного забезпечення // Машинобудування очима молодих: прогресивні ідеї – наука – виробництво (МОМ

– 2017) : матеріали тез доповідей XVII Міжнародної науково-практичної конференції, м. Чернігів, 01 – 03 листопада 2017 р. Чернігів : ЧНТУ, 2017. С. 31-33. *(форма участі – заочна)*

49. **Копей V.**, Panchuk V., Onysko O. Component-oriented modelling of dynamical systems in Python language on the example of the model of the sucker rod string // International conference Innovative Ideas in Science 2017 : book of abstracts, Banja Luka, 02 - 03 November 2017. Banja Luka : University of Business Engineering and Management, 2017. P. 33. *(форма участі – очна)*

50. **Копей В. Б.** Статистичні моделі відмов колон насосних штанг // Матеріали II Міжнародної науково-технічної конференції "Машини, обладнання і матеріали для нарощування вітчизняного видобутку нафти і газу PGE – 2018", м. Івано-Франківськ, 24-27 квітня 2018 р. Івано-Франківськ : ІФНТУНГ, 2018. С. 310-313. *(форма участі – очна)*

51. **Копей В. Б.** Прогнозування частоти відмов колон насосних штанг за допомогою ансамблів дерев рішень // Комплексне забезпечення якості технологічних процесів та систем (КЗЯТПС – 2018) : матеріали тез доповідей VIII Міжнародної науково-практичної конференції, м. Чернігів, 10–12 травня 2018 р. : у 2-х т. / Чернігівський національний технологічний університет [та ін.]; відп. за вип.: Єрошенко Андрій Михайлович [та ін.]. Т. 2. Чернігів : ЧНТУ, 2018. С. 198-200. *(форма участі – заочна)*

52. **Копей V.**, Onysko O., Panchuk V. Computerized system based on FreeCAD for geometric simulation of thread turning of oil and gas pipes // 6th International Conference of Applied Science, May 9-11, 2018, Banja Luka, Bosnia and Herzegovina : book of abstracts. Banja Luka : University of Banja Luka, 2018. P. 108. *(форма участі – очна)*

53. **Копей В. Б.** Алгоритм інтелектуальної системи на основі міждисциплінарних досліджень загальносистемних закономірностей // Комп'ютерне моделювання та оптимізація складних систем (КМОСС-2018) : матеріали IV Міжнародної науково-технічної конференції, м. Дніпро, 1-2 листопада 2018 р. / Міністерство освіти і науки України, Державний вищий

навчальний заклад "Український державний хіміко-технологічний університет".
Дніпро : Баланс-клуб, 2018. С. 246-248. (*форма участі – заочна*)

54. **Копей В. Б.**, Онисько О. Р., Жигуц Ю. Ю. Обґрунтування застосування двоопорних муфтових різьбових з'єднань пустотілих насосних штанг // Матеріали доповідей VIII Міжнародної науково-технічної конференції "Прогресивні технології у машинобудуванні РТМЕ-2019", 4-8 лютого 2019 р., м. Івано-Франківськ-Яремче. Івано-Франківськ : ІФНТУНГ, 2019. С. 146-149. (*форма участі – очна*)

55. **Kopei V.**, Onysko O., Panchuk V. Principles of the product lifecycle management system development for threaded connections based on the Python programming language // International Conference of Applied Science ICAS 2019 : book of abstracts, May 9-11, 2019, Hunedoara, Romania. Timisoara : Editura EUROBIT, 2019. P. 42. (*форма участі – очна*)

56. **Kopei V. B.**, Kopei B. V. Harmonic axial loading analysis of the tubing threaded connection // Modern achievements of science and education : proceedings of XIV International scientific conference, September 26 - October 3, 2019, Netanya, Israel. Khmelnytskyi : KhNU, 2019. P. 29-37. (*форма участі – очна*)

Праці, які додатково відображають наукові результати дисертації

57. Комп'ютерна програма "Програма для побудови баз знань з проблем надійності і довговічності штангових свердловинних насосних установок" ("SuckerRodPumpingUnitKB") : свідоцтво про реєстрацію авторського права на твір № 42157 / **Копей Володимир Богданович**. Дата реєстрації 08.02.2012 // Каталог державної реєстрації. Вип.16/2012. С. 42.

58. Комп'ютерна програма "Модуль для компонентно-орієнтованого моделювання динамічних систем та приклади його застосування для побудови моделей колони насосних штанг" : свідоцтво про реєстрацію авторського права на твір № 73098 / **Копей Володимир Богданович**. Дата реєстрації 25.07.17 // Бюл. "Авторське право і суміжні права" № 46/2017. С. 154.

ДОДАТОК Б

Статистичні моделі відмов штангових колон

Код програм також доступний в GitHub (vkopey/RodStats: Statistical models of sucker rod strings failures. URL: <http://github.com/vkopey/RodStats>).

Таблиця Б.1 – Частина статистичних даних відмов колон ШН

N	Na me	Year	Mo nth	Pump	H	H_ fail	D_ fail	Type	H16	H19	H22	H25	Gas	Wa ter	Pro duct	Para ffin	Curv_ begin	Curv_ end	Stress
1	299	1999	12	95	815	416	25	4	0	0	0	816	222	98	4566	0			107
2	39	1999	12	70	1127	176	25	2	0	0	576	552	282	98	5027	0			94
3	514	1999	12	57	1241	0	0	5	0	232	656	248	450	97	2482	0	1030	1120	72
4	354	1999	12	57	1508	688	19	3	0	1144	208	152	181	75	2288	0			84
5	22 стр	1999	12	44	1552	752	22	8	0	480	864	168	324	74	1952	1			69
6	24 сп	1999	12	32	1080	400	22	4	0	304	720	16	514	54	564	1			37
7	30 сп	1999	12	44	1497	992	19	1	0	680	664	128	450	52	588	1			66
8	168 пд	1999	12	38	1680	416	22	1	0	0	1344	336				0			71
9	710	1999	12	57	1345	1089	22	1	0	0	768	560	279	86	2460	1			86
10	246	1999	12	44	1532	264	22	4	0	0	1344	176	101	90	3554	0			72
11	54 стр	1999	12	44	1644	430	22	8	0	240	1152	192	400	69	1757	1			74
12	51 стр	1999	12	44	1708	0	0	5	0	40	1136	472	246	72	2537	0			79
13	907	1999	12	44	1550	200	22	1	0	1032	488	16	208	72	2518	0			66
14	249	1999	12	57	1426	1320	19	1	0	200	896	288	217	85	4203	0	1080	1426	86
15	675	1999	12	57	1290	776	22	7	0	336	832	112				1	340	350	78
16	317	1999	12	95	1000	424	25	7	0	0	128	872	155	90	1210	0			127
17	235	1999	12	57	1097	656	22	8	0	312	688	88	250	97	3080	0			68
18	760	1999	12	38	1596	120	25	7	0	936	464	144		75	1396	0			60
19	60 стр	1999	12	44	1800	1280	19	2	0	480	896	488	256	63	1778	1			85
20	85 стр	1999	12	44	1732	880	22	8					468	71	631	1			
21	28	1999	12	44	1300	0	0	5					302	90	262	1	1000	1300	
22	238	1999	11	95	984	904	25	8	0	0	0	992	160	99	5482	1			127
23	820	1999	11	57	1454	720	22	4	0	0	1096	336	193	87	1750	1			90
24	244	1999	11	57	1497	480	22	2	0	48	1352	80	121	95	4970	1			90
25	824	1999	11	44	1473	1441	19	1	0	208	1104	40	223	95	2857	1			63
26	246	1999	11	44	1532	776	22	8	0	0	1352	160	101	90	3554	0			72
27	246	1999	11	44	1532	312	22	4	0	0	1352	160	101	90	3554	0			72
28	165 пд	1999	11	57	1347	568	22	1	0	0	880	392				0			82
29	40 стр	1999	11	44	1303	0	0	5	0	616	584	80				0			58
30	4 підб	1999	11	32	1838	640	19	1	0	696	440	688				0			66
31	79 підб	1999	11	32	1656	130	25	4	0	1008	448	224				0			57
32	210	1999	11	70	1192	672	19	2	0	464	208	520	217	92	6469	0			95
33	703	1999	11	44	1596	280	22	2	0	816	456	264	212	91	3655	0			69
34	1 ч	1999	11	32	1775	200	25	3	0	704	448	520	1333	74	42	0			60
35	51 стр	1999	11	44	1708	1304	22	2	0	40	1136	472	246	72	2537	0			79
36	921	1999	11	57	1501	440	22	2	0	232	832	408	269	90	9622	0			91
37	26 пд	1999	11	38	1600	696	22	2	0	712	752	272				1			69
38	916	1999	11	44	1667	360	22	2	0	608	640	376				0			74
39	141 пд	1999	11	44	1606	320	22	7	0	856	672	40	694	13	1236	0			68
40	34 пд	1999	11	32	1763	0	0	5								0			
41	176 пд	1999	11	29	1600	400	22	7	0	640	960	0				0			51
42	52 стр	1999	11	44	1676	1376	22	2	0	64	1408	152	207	65	2451	1			76
43	86 пд	1999	11	57	1335	280	25	1	0	88	672	520				0			83
44	326	1999	11	70	1264	1040	22	7	0	0	224	1032	222	94	4667	0	1185	1264	106

Лістинг Б.1 – polyfit3.py – програма для побудови статистичних моделей виду $f(h)$

```

#encoding: utf-8
from scipy.optimize import curve_fit
import numpy as np
import itertools

def score(y, ya): #r**2
    return np.corrcoef(y, ya)[0, 1]**2

def poly(x, A, P):
    """Значення полінома з коефіцієнтами A і степенями P"""
    comp=[a*x**p for a,p in zip(A,P)] # доданки полінома
    return sum(comp) # значення полінома a[0]+a[1]*x+a[2]*x**2

def pfit(x, y, P, Z):
    """Апрокс. поліномом з степенями P з врахуванням вектора занулення Z"""
    def polyz(x, *A): # поліном з коеф. A з їх зануленням вектором Z"""
        A=[a*z for a,z in zip(A,Z)] # коеф. полінома з врахув. занулення
        return poly(x, A, P)

    A, cov = curve_fit(polyz, x, y, p0=Z) # апроксимувати нелінійним
    методом найменших квадратів
    s=score(y,poly(x, A, P)) # внутрішній критерій - наскільки добре p
    описує точки x,y
    return A, s # коефіцієнти полінома і R**2

def crossValidation(x, y, P, Z):
    """Повертає узагальнений критерій"""
    # ділимо дані на групи
    A,s0=pfit(x,y,P,Z) # підгонка усього
    x1,y1=x[::2],y[::2] # непарні
    x2,y2=x[1::2],y[1::2] # парні
    A1,s01=pfit(x1,y1,P,Z) # підгонка групи 1
    s1=score(y2, poly(x2,A1,P))
    A2,s02=pfit(x2,y2,P,Z) # підгонка групи 2
    s2=score(y1, poly(x1,A2,P))
    s3=score(poly(x,A1,P), poly(x,A2,P))
    #return np.mean([s0,s3]) # або
    return np.mean([s0,s1,s2,s3])
    #return s1
    #return np.mean([s01,s1]) # найпростіший випадок

def combi(x,y,Np=4):
    """Комбінаторний алгоритм індуктивної самоорганізації моделі

```

```

x,y - координати емпіричних точок
Np - максимальна степінь полінома"""
P=range(Np+1) # степені полінома [0,1,2,3,...]
res=[] # результати
# усі можливі комбінації
for Z in itertools.product([0,1],repeat=Np+1):
    #Z - список для занулення коефіцієнтів (1,1,0,...)
    if any(Z[1:]): # окрім поліномів f=0 та f=const
        res.append([Z, crossValidation(x,y,P,Z)]) # узагальн. критерій
res.sort(key=lambda x: x[1], reverse=True) # сорт. за спаданням score
return res # відсортований список Z, score

def plot(x,y,Z,xmin=None,xmax=None):
    """Рисує дані і поліном Z"""
    Np=len(Z)-1
    P=range(Np+1)
    A,s=pfit(x, y, P, Z)
    print "A, R**2 = ", A,s
    import matplotlib.pyplot as plt
    plt.plot(x,y,'ro')
    if xmin==None: xmin=x.min()
    if xmax==None: xmax=x.max()
    x_=np.linspace(xmin,xmax,100)
    y_=poly(x_,A,P)
    plt.plot(x_,y_,'k-',linewidth=2)

    from scipy.integrate import cumtrapz, quad
    y_int = cumtrapz(y_, x_, initial=0) # знач. первісної шляхом інтегрув.
    print 'Площа під кривою=', quad(poly,xmin,xmax,args=(A,P))[0]
    plt.plot(x_,y_int,'k--',linewidth=2)
    plt.show()
    #print "integral, score=", score2(A,P)

# def score2(A,P):
#     """Додатковий критерій. Для підгонок функцій густини розподілу"""
#     from scipy.integrate import quad
#     s, err = quad(poly, 0, 1, args=(A,P)) # інтегрувати в межах
#     # s повинна бути рівна 1
#     return s, np.exp(-(s-1)**2) # повертає від 0 до 1

if __name__=='__main__': # приклад використання
    N=16 # кількість точок
    # координати емпіричних точок
    x = np.linspace(0, 10, N)
    y = np.array([0,2,1,4,5,7,6,7,8,7,9,9,8,9,9,9])
    Np=4 # степінь полінома

```

```

res=combi(x,y,Np) # список моделей
for i in res:
    print i
plot(x,y,Z=res[0][0]) # рисуємо найкращий поліном

```

Лістинг Б.2 – forSclearn3.py – програма для прогнозування частоти аварій свердловини за її параметрами (регресія)

```

# -*- coding: utf-8 -*-
"""
Задача регресії - прогнозування частоти аварій свердловини за її
параметрами. Те саме що forSclearn2.py, але для пошуку оптимальних
параметрів використовували differential_evolution
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('123_2017_part.csv',sep=';') # прочитати з файлу csv
df=df[['Pump', 'Stress', 'Gas', 'Curv', 'Water', 'H', 'Product',
'Paraffin', 'Kv', 'H19', 'H22', 'H25', 'Month']]

df=df.dropna()
#print df
X=df.drop(['Kv'], axis=1)
y=df['Kv']

from sklearn.utils import shuffle
X, y = shuffle(X, y) # випадкове перемішування даних

from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

# дані для побудови моделі і екзамену
X_, x_test, y_, y_test =
train_test_split(X,y,test_size=0.2)#,random_state=0)

def f(x):
    #n_e, m_d = int(round(x[0])), int(round(x[1]))
    #model=RandomForestRegressor(n_estimators=n_e, max_depth=m_d)

```



```

n_e, l_r, m_d = int(round(x[0])), x[1], int(round(x[2]))
model=GradientBoostingRegressor(n_estimators=n_e, learning_rate=l_r,
max_depth=m_d)
s=cross_val_score(model, X_, y_, cv=7) # перехресна перевірка
print s.mean()
return -s.mean()

from scipy.optimize import differential_evolution
#bounds = [(50, 150), (3, 5)] # границі
bounds = [(50, 150), (0.01, 0.6), (3, 5)] # границі
# закоментуйте наступні 2 рядки, якщо параметри моделі уже відомі
#res = differential_evolution(f, bounds=bounds)
#print res
##
# найкраща модель
#model = RandomForestRegressor(n_estimators=133, max_depth=5)
model = GradientBoostingRegressor(n_estimators=120, learning_rate=0.15,
max_depth=4)
model.fit(X_, y_) # найкраща модель для даних для навчання
Y_test=model.predict(x_test) # екзамен на тестових даних (Мюллер с.279)
from sklearn.metrics import r2_score
print r2_score(y_test, Y_test) # або model.score(x_test, y_test)

plt.scatter(y_test, Y_test) # реальні і прогнозовані тестові дані
plt.xticks(np.arange(1,16))
#plt.yticks(np.arange(0,17))
#plt.axis('equal')
plt.grid()
plt.show()

imp=zip(model.feature_importances_, X.columns.values)
for score, name in sorted(imp,reverse=True):
    print score, name

## виконати ці рядки з консолі, а потім виконати програму N раз для
перехресної перевірки
#yt=dict.fromkeys(range(1,16))
#for i in yt: yt[i]=[]
##
for y,Y in zip(y_test, Y_test):
    yt[y].append(Y)

# зберігає у файл csv. Потім відкрити в Excel, замінити '.' на ',',
зберегти і передати в Statistica 10 для аналізу
df = pd.DataFrame() # об'єкт DataFrame
for i in yt: # кожний стопчик

```

```

df[i] = pd.Series(yt[i]) # об'єкт Series
df.to_csv('yt_Yt.csv',sep=';',index=False,header=False,mode='w+') # увага!
режим додавання до файлу

# print np.mean(yt[1]), np.std(yt[1])
# plt.figure()
# plt.hist(yt[1]) # гістограма
# plt.show()

```

Лістинг Б.3 – forSclearn1.py – програма для прогнозування класу аварійності свердловини за її параметрами (класифікація)

```

# -*- coding: utf-8 -*-
"""
Задача класифікації - прогнозування класу аварійності свердловини за її
параметрами. Те саме що forSclearn0.py, але для пошуку оптимальних
параметрів використовували differential_evolution
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('123_2017_part.csv',sep=';') # прочитати з файлу csv
df=df[['Pump', 'Stress', 'Gas', 'Curv', 'Water', 'H', 'Product',
'Paraffin', 'H19', 'H22', 'H25', 'Month', 'Kv_Cat']]

df=df.dropna()
X=df.drop(['Kv_Cat'], axis=1)
y=df['Kv_Cat']

from sklearn.utils import shuffle
X, y = shuffle(X, y) # випадкове перемішування даних #, random_state=0
from sklearn.ensemble import
RandomForestClassifier,GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

# дані для побудови моделі і екзамєну
X_, x_test, y_, y_test =
train_test_split(X,y,test_size=0.2)#,random_state=0)

def f(x):

```

```

#n_e, m_d = int(round(x[0])), int(round(x[1]))
#model=RandomForestClassifier(n_estimators=n_e, max_depth=m_d)
n_e, l_r, m_d = int(round(x[0])), x[1], int(round(x[2]))
model=GradientBoostingClassifier(n_estimators=n_e, learning_rate=l_r,
max_depth=m_d)
s=cross_val_score(model, X_, y_, cv=7) # перехресна перевірка
print s.mean()
return -s.mean()

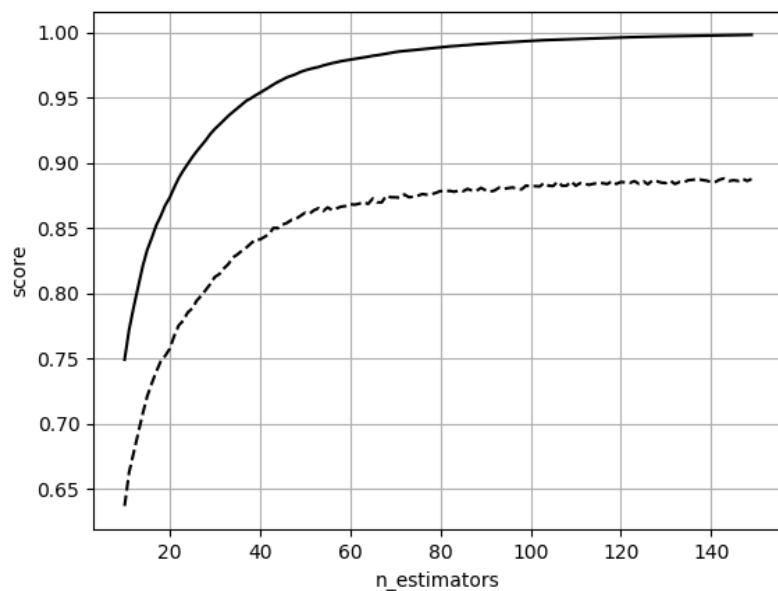
from scipy.optimize import differential_evolution
#bounds = [(50, 150), (3, 5)] # границі
bounds = [(50, 150), (0.01, 0.6), (3, 5)] # границі
# закомментуйте наступні 2 рядки, якщо параметри відомі
#res = differential_evolution(f, bounds=bounds)
#print res
##
#model = RandomForestClassifier(n_estimators=121, max_depth=5)
model=GradientBoostingClassifier(n_estimators=120,learning_rate=0.38,max_de
pth=5)
model.fit(X_,y_)

imp=zip(model.feature_importances_, X.columns.values)
for score, name in sorted(imp,reverse=True):
    print score, name

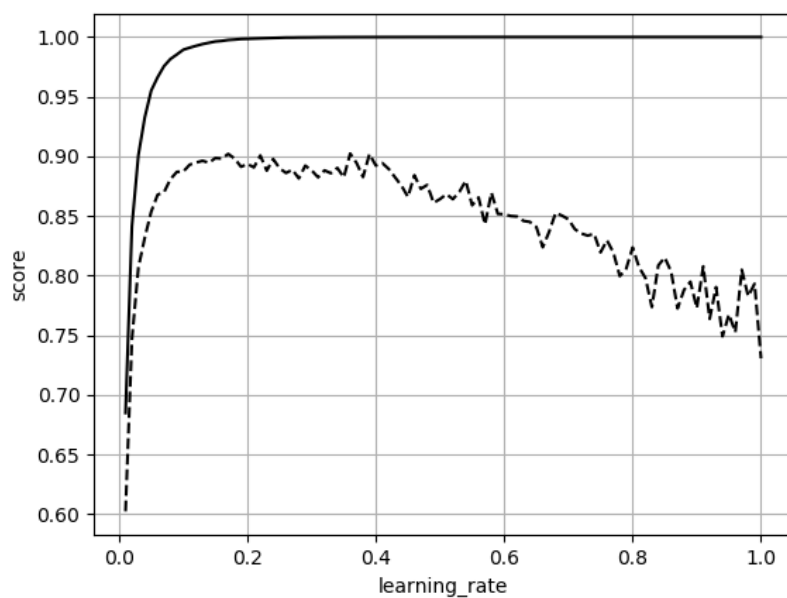
# звіт по класифікації (виконайте кілька раз і обчисліть середнє)
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
Y_test=model.predict(x_test) # екзамен на тестових даних (Мюллер с.279)
print confusion_matrix(y_test, Y_test) # матриця помилок
print classification_report(y_test, Y_test) # повний звіт по класифікації

## крива точності-повноти
# будується для різних порогових значень імовірності
from sklearn.metrics import precision_recall_curve
y_scores=model.predict_proba(x_test)[:,:1] # імовірності класу 1 тест. даних
precision, recall, thresholds = precision_recall_curve(y_test, y_scores)
plt.plot(precision, recall, 'k-')
for p,r,t in zip(precision[:-1], recall[:-1], thresholds)[:4]: # кожна
четверта імовірність
    plt.text(p,r,round(t,2))
plt.xlabel(u"Точність"), plt.ylabel(u"Повнота")
plt.show()
# середня точність класифікатора (площа під кривою точність-повнота)
from sklearn.metrics import average_precision_score
print average_precision_score(y_test, y_scores)

```



а



б

а) – для параметра $n_estimators$; б) – для параметра $learning_rate$
 Рисунок Б.1 – Криві перевірки GradientBoostingRegressor

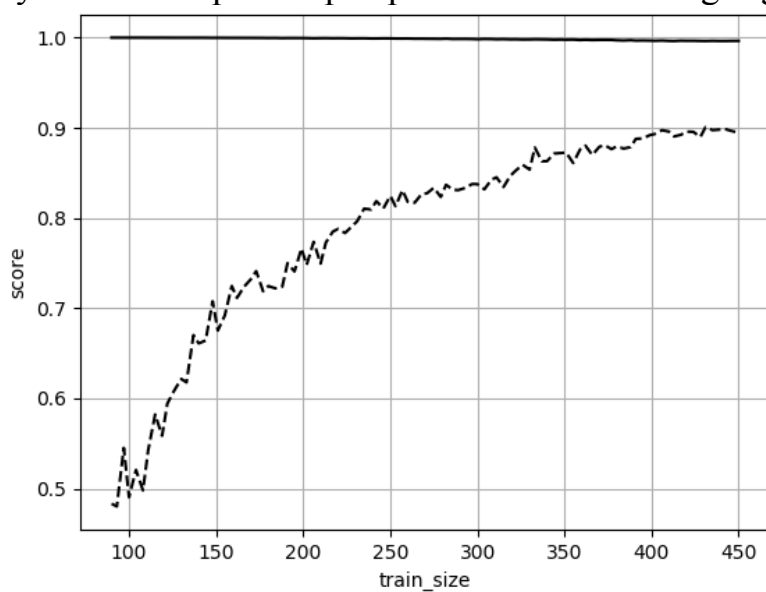
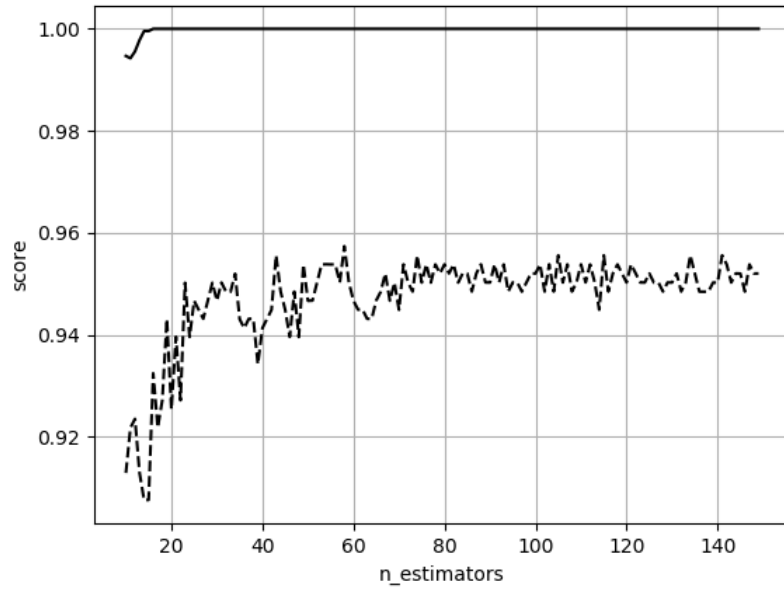
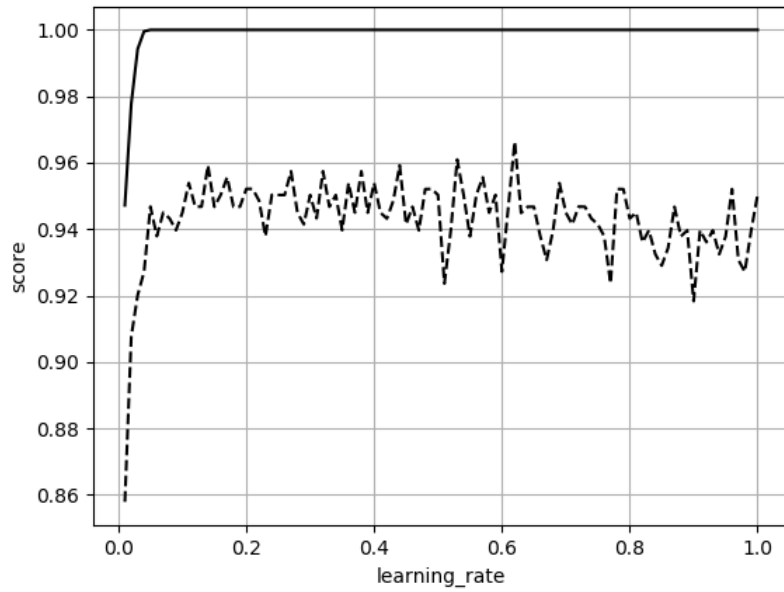


Рисунок Б.2 – Криві навчання GradientBoostingRegressor



а



б

а) – для параметра $n_estimators$; б) – для параметра $learning_rate$
 Рисунок Б.3 – Криві перевірки GradientBoostingClassifier

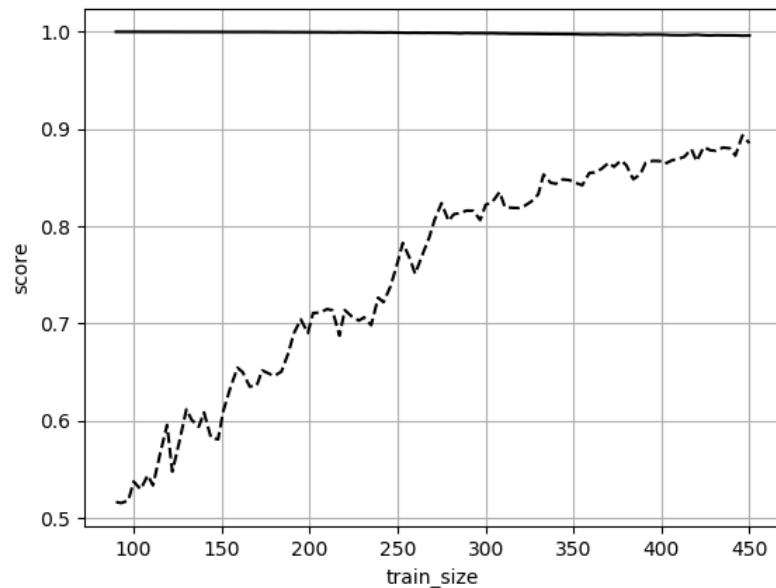
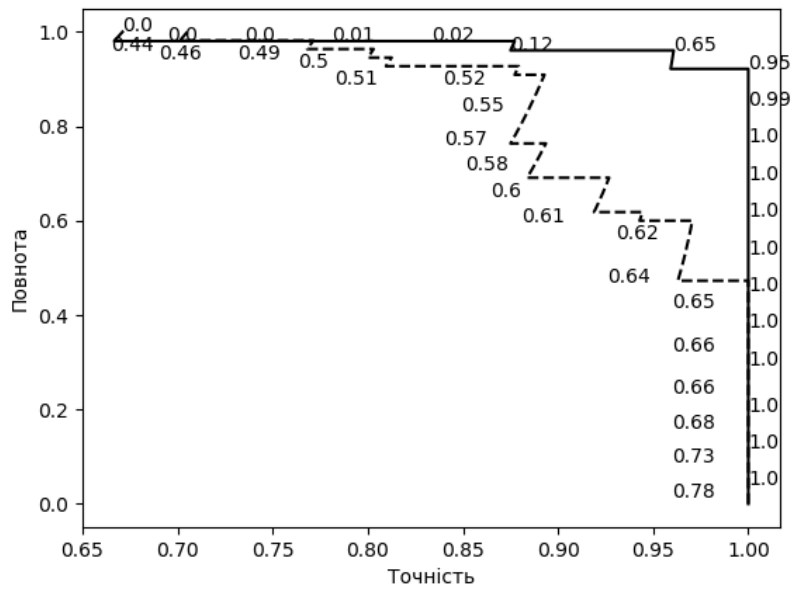


Рисунок Б.4 – Криві навчання GradientBoostingClassifier



(- -) – RandomForestClassifier; (—) – GradientBoostingClassifier

Рисунок Б.5 – Крива точність-повнота для моделей

ДОДАТОК В

Модель ШСНУ на основі абстрактних автоматів

Код програм також доступний в GitHub (vkorey/PU. URL: <http://github.com/vkorey/PU>).

Лістинг В.1 – PUmодель.py – модуль для підготовки параметрів моделей ШСНУ

```
# -*- coding: cp1251 -*-
import math
# Для моделювання порожнистої трубчатої штанги її замінюємо еквівалентною
суцільною з такими ж: зовнішнім діаметром, пружністю, масою. Але тоді у неї
буде інший модуль E і густина
def eqYoungsModulus(E,D,d):
    """Еквівалентний модуль Юнга суцільної штанги діаметром D
    з умови однаковості пружності  $k=E*A/L=Eeq*Aeq/L$ 
    суцільної (діаметром D) і трубчатої штанги (зовн. діаметром D,
    внутрішнім d і модулем Юнга E)"""
    A=math.pi*(D**2.0-d**2.0)/4.0 # площа перетину труби
    Aeq=math.pi*(D**2.0)/4.0 # площа перетину суцільної штанги
    return E*A/Aeq

def eqDensity(Ro,D,d,Roo=0.0):
    """Еквівалентна густина суцільної штанги діаметром D
    з умови однаковості маси  $m=Ro*A*L+Roo*Ao*L=Roeq*Aeq*L$ 
    суцільної (діаметром D) і трубчатої штанги (зовн. діаметром D,
    внутрішнім d, густиною тіла Ro і тіла отвору Roo)
    Якщо Roo=0 - порожнисті штанги наповнені повітрям"""
    A=math.pi*(D**2.0-d**2.0)/4.0 # площа перетину труби
    Ao=math.pi*(d**2.0)/4.0 # площа перетину отвору
    Aeq=math.pi*(D**2.0)/4.0 # площа перетину суцільної штанги
    return (Ro*A+Roo*Ao)/Aeq

class SuckerRod(object):
    """Клас насосних штанг або секції однотипних штанг. Одиниці СИ"""
    diameters=[0.016,0.019,0.022,0.025] # діаметри тіла
    length=8.0 # довжина (за замовчуванням)
    materials={"сталь":{"модуль пружності":2.1e11,
                        "густина":8000.0}, # тут густина дещо завищена для
    врахування мас муфт і головок
              "склопластик":{"модуль пружності":0.5e11,
                              "густина":2100.0},
              "стальТр22Екв":{"модуль
    пружності":eqYoungsModulus(2.1e11,0.022,0.0085),
    "густина":eqDensity(8000.0,0.022,0.0085,1000.0)},
```

```

        "склопластикТр22Екв": {"модуль
пружності": eqYoungsModulus(0.5e11, 0.022, 0.0085),
"густина": eqDensity(2100.0, 0.022, 0.0085, 1000.0)}}
    #!!! якщо Ro0=0, то порожнисті штанги є герметичними і наповнені
повітрям (але це недоцільно)
    alfa=0.0 # кут відхилення свердловини внизу секції від вертикалі (рад.)

    def __init__(self, PU, diametr, length=8.0, material="сталь",
alfa=0.0):
        """Конструктор"""
        self.PU=PU # свердловинна штангова насосна установка
        self.diametr=diametr # діаметр тіла штанги
        self.length=length # довжина штанги або секції
        self.material=material # матеріал
        self.alfa=alfa

    def area(self):
        """Площа поперечного перетину  $S=\pi*r^2$ """
        return math.pi*(self.diametr/2.0)**2

    def volume(self):
        """Об'єм, який займає штанга в рідині  $V=L*\pi*r^2$ """
        return self.length*math.pi*(self.diametr/2.0)**2

    def mass(self):
        """Маса  $m=V*ro$ """
        return self.volume()*self.materials[self.material]["густина"]

    def fluidVolume(self):
        """Об'єм рідини в НКТ навколо штанги (секції)"""
        A=self.PU.well.sectionalArea()-self.area() # увага! тут площа НКТ,
а не плунжера
        return A*self.length

    def fluidMass(self):
        """Маса рідини в НКТ навколо штанги (секції)"""
        return self.PU.well.density*self.fluidVolume()

    def weight(self):
        """Вага в рідині"""
        fluidDensity=self.PU.well.density # густина рідини
        Farh=fluidDensity*9.81*self.volume() # сила Архімеда
        w=self.mass()*9.81 # вага в повітрі
        return w-Farh

    def stiffness(self):
        """Жорсткість  $k=E*S/L$ """
        E=self.materials[self.material]["модуль пружності"]
        return E*self.area()/self.length

```



```

def hydrodynamicRodResistance(self):
    """Гідродинамічний опір 'с' штанги (секції) за Пірвердяном
    Враховується під час ходу вниз. Сила опору  $F=c*v$ """
    msh=self.PU.well.diametr/self.diametr
    Msh=1.0/((math.log(msh)*(msh**2+1)/(msh**2-1))-1) # коефіцієнт Мшт
    #  $v=2nS$  - швидкість штанг
    R=math.pi**2*self.PU.well.viscosity*self.PU.well.density*Msh
    return R*self.length

def hydrodynamicRodResistance2(self):
    """Гідродинамічний опір 'с' штанги (секції) за Андреевим
    Враховується під час ходу вниз. Сила опору  $F=c*v$ """
    m=self.diametr/self.PU.well.diametr
    R=16.9*self.PU.well.viscosity*self.PU.well.density*m**5.49
    R=R*1000 # одиниці ?
    return R*self.length

def materialRodResistance(self, sigmaA=None):
    """Еквівалентний коефіцієнт опору матеріалу, Н*с/м
    sigmaA - амплітуда напружень (Па)
    sigmaA=None - не залежить від ампл. напруж."""
    k=self.stiffness() # жорсткість стержня
    # визначаємо коефіцієнт поглинання матеріалу
    if self.material.startswith("сталь"):
        if sigmaA!=None: # якщо залежить від ампл. напруж.
            psi=6e-5*sigmaA/1000000.0
        else: # якщо не залежить від ампл. напруж.
            #psi<0.13 (метали в пружному діапазоні)
            #psi=0.25...0.5 (металічні структури)
            #psi=0.38...0.88 (металічні структури зі з'єднаннями)
            psi=0.01
    if self.material.startswith("склопластик"):
        if sigmaA!=None: # якщо залежить від ампл. напруж.
            psi=4e-4*sigmaA/1000000.0 # довідник "Вибрации в технике"
        else: # якщо не залежить від ампл. напруж.
            psi=0.08
    omega=self.PU.pumpingUnit.omega # кутова частота зовнішньої сили
    #m=self.mass() # маса стержня
    #omega=math.sqrt(k/m) # власна частота стержня
    #["Вибрации в технике" т6 с130 ф13,19]
    c=psi*k/(2*math.pi*omega) #k=E*S/L

#      #!уточнити врахування окремо опору з'єднань
#      nz=int(self.length/8) # кількість муфтових з'єднань на секції
#      kz=1000000.0 # жорсткість муфтового з'єднання ?
#      cz=0.2*kz/(2*math.pi*omega) # екв коеф опору муфтового з'єднання
#      (psi=0,006...0,4)
#      czs=1/(nz*(1/cz)) # екв коеф опору усіх з'єднань на секції
#      return 1/(1/c+1/czs) # екв коеф опору секції (матеріал+з'єднання)
return 20.0*c # увага збільшено у 20.0 (для спрощ. моделі 50)

```

```
#####
class SuckerRodString(object):
    """Клас колони насосних штанг"""
    items=[] # список штанг (секцій) [штанга1, штанга2,...](починаючи з
гирла)

    def __init__(self, PU, sections):
        """Конструктор.
        PU - ШСНУ
        sections=[(діаметр, довжина, матеріал, кут відхилення від
вертикалі(рад.),...)]"""
        self.PU=PU
        for diametr, length, material, alfa in sections: # для кожної
секції
            # додати штангу (секцію однотипних штанг)
            self.items.append(SuckerRod(PU, diametr, length, material,
alfa))

    def length(self, start=0, end=None):
        """Довжина. start, end - індекси зрізу"""
        return sum([r.length for r in self.items[start:end]])

    def xList(self):
        """Список координат нижніх кінців секцій"""
        X=[]
        x=0.0 # координата
        for r in self.items:
            x=x+r.length # додати довжину даної секції
            X.append(x)
        return X

    def height(self):
        """Висота проєкції колони штанг на вертикаль"""
        return sum([r.length*math.cos(r.alfa) for r in self.items])

    def volume(self, start=0, end=None):
        return sum([r.volume() for r in self.items[start:end]])

    def stiffness(self):
        """Сумарний коеф. жорсткості колони 1/k=1/k1+1/k2+..."""
        return 1/sum([1/r.stiffness() for r in self.items])

    def damping(self):
        """Сумарний коеф. демпфування колони 1/d=1/d1+1/d2+...
[Вибрации в технике т6 с177]"""
        return 1/sum([1/r.materialRodResistance() for r in self.items])

    def mass(self, start=0, end=None):
        return sum([r.mass() for r in self.items[start:end]])

    def massList(self):
```

```

    """Список мас секцій"""
    return [r.mass() for r in self.items]

def fluidMassList(self):
    """Список мас рідини біля вузлів"""
    return [r.fluidMass() for r in self.items]

def weight(self, start=0, end=None):
    return sum([r.weight() for r in self.items[start:end]])

def wellAngleList(self):
    """Список кутів свердловини внизу секцій"""
    return [x.alfa for x in self.items]

def secAngleList(self):
    """Повертає список кутів секцій an_i
    за кутами свердловини внизу секції alfa_i.
           an0           an1           an2
    -----0-----0-----0
           alfa0           alfa1           alfa2
    Розраховує приблизно за умови рівності довжин секцій.
    """
    angles=[self.items[0].alfa] # кут верхньої секції=куту верхньої
ділянки свердловини
    aprev=self.items[0].alfa # кут першої ділянки свердловини (вверху).
Як правило 0.
    for r in self.items[1:]: # для кожної ділянки крім першої
        a=(aprev+r.alfa)/2 # розрахувати наближено кут
        angles.append(a) # додати в список
        aprev=r.alfa # запам'ятати як попередню секцію
    return angles

def angleTop(self):
    """Кути між верхньою штангою і свердловиною"""
    AN=self.wellAngleList()
    ANS=self.secAngleList()
    return [ans-an for an,ans in zip(AN,ANS)]

def angleBottom(self):
    """Кути між свердловиною і нижньою штангою
    Останнє значення 0"""
    AN=self.wellAngleList()
    ANS=self.secAngleList()
    return [an-ans for an,ans in zip(AN[:-1],ANS[1:])] + [0.0]

def hydrodynamicRodResistanceList(self):
    """Список гідродинамічних опорів секцій"""
    return [r.hydrodynamicRodResistance() for r in self.items]

def weightForceList(self):
    """Список осьових сил від ваги секцій

```

```
(з врахуванням кута свердловини)""""
return [r.weight()*math.cos(r.alfa) for r in self.items]
```

```
class Well(object):
    """Клас свердловини"""
    diameters=[60.3-2*5.0, 73.0-2*5.5, 73.0-2*7.0, 88.9-2*6.5, 114.3-2*7.0]
    diametr=diameters[4]/1000.0 #4 діаметр НКТ (однаковий на інтервалах)
    density=1000.0 #1000.0 густина суміші [Белов]
    viscosity=0.000002 #0.000001 # кінематична в'язкість (змінна по висоті!!!)
    bulkModulus=2.2e9 # об'ємний модуль пружності

    def sectionalArea(self):
        """Площа поперечного перетину НКТ"""
        return math.pi*(self.diametr/2.0)**2
```

```
class Pump(object):
    """Клас свердловинного насоса"""
    diameters=[27, 32, 38, 44, 50, 57, 57, 63, 70, 95]
    diametr=38/1000.0 #38 діаметр плунжера
    valveDiametr=0.025 # діаметр отвору сідла клапана
    openValveResistance=0.00001 # опір відкритого клапана ???
    pistonGap=0.1e-3 #0.1e-3 зазор між плунжером і циліндром
    # далі v - швидкість плунжера, vv - швидкість в сідлі клапана !!!

    def __init__(self,PU):
        self.PU=PU # ШСНУ

    def volumeFlowRate(self,v):
        """Теоретична подача насоса (об'ємна витрата) Q=A*v"""
        return self.pistonArea()*v #self.PU.pumpingUnit.velAvg()

    def pistonArea(self):
        """Площа поперечного перетину плунжера"""
        return math.pi*(self.diametr/2.0)**2 # !!! приблизно

    def valveDischargeCoefficient(self,Re):
        """Коефіцієнт витрати клапана. Залежить від числа Рейнольдса.
[Гіматудінов]"""
        if Re<=225:
            mu=0.0846*Re**0.2872
        elif Re<=30000:
            mu=0.4
        elif Re<=300000:
            mu=0.0085*Re**0.3764
        else: mu=1.0
        return mu

    def valveLossCoefficient(self,Re):
        """Коефіцієнт витрат клапана (місцевого опору)"""
        return 1.0/self.valveDischargeCoefficient(Re)**2.0
```

```

def valveV(self,v):
    """Швидкість потоку в сідлі, v - швидкість плунжера"""
    return self.volumeFlowRate(v)/(math.pi*(self.valveDiametr/2.0)**2)

def valveRe(self,vv):
    """Число Рейнольдса, vv - швидкість потоку в сідлі"""
    nu=self.PU.well.viscosity
    Re=abs(vv)*self.valveDiametr/nu
    return Re

def dynamicPressure(self, vv):
    """Динамічний тиск, vv - швидкість потоку в сідлі"""
    ro=self.PU.well.density
    return vv*vv*ro/2

def dPvalve(self,vv):
    """Перепад тисків на клапані [Гіматудінов]
    vv - швидкість потоку в сідлі"""
    Re=self.valveRe(vv)
    dP=self.valveLossCoefficient(Re)*self.dynamicPressure(vv)
    return dP

def valveHydroResistForce(self,v):
    """Сила гідродинамічного опору клапана насоса.
    Діє під час ходу вниз і ввєрх. Направлена проти руху
    v - швидкість плунжера"""
    vv=self.valveV(v)
    return self.dPvalve(vv)*self.pistonArea() # повертає завжди додатне

def pistonFrictionForce(self):
    """Сила тертя плунжера об стінки циліндра, Н.
    Формула Сердюка. Орієнтовно 1000 Н
    Діє під час ходу вниз і ввєрх. Направлена проти руху"""
    return 1.84*self.diametr/self.pistonGap-137 # для змащування водою

class PumpingUnit():
    """Клас верстата-гойдалки"""
    stroke=3.0 #2.0 # довжина ходу точки підвіски
    n=6.5 #10.0 # кількість гойдань (подвійних ходів) за хвилину
    fi=math.pi # початковий кут (почати з верхньої точки або з середини,
    якщо колона стабілізована)
    # ланки механізму (довжина, маса, момент інерції відносно осі z, яка
    проходить через центр мас)
    # Тип: СКД8-3-4000 (дезаксіальний)
    p1=dict(L=1.29,m=1982.0,Iz=635.0) # два кривошипи
    p2=dict(L=3.0,m=499.0,Iz=728.0) # два шатуна і траверса
    p3=dict(L=2.0,m=643.0,Iz=443.0) # заднє плече балансира
    p4=dict(L=2.29,m=911.0,Iz=1106.0) # переднє плече балансира
    p5=dict(L=0.789,m=4*750.0,Iz=0.0) # противаги (L - радіус центра мас)
    Приймаємо, що маса зосереджена в точці (Iz=0.0)

```

```

# Увага! Значення за замовчуванням р5 уточнити функцією balancing
dx=1.345 # горизонтальна відстань між осями опори балансира і
тихохідного валу редуктора
dy=3.012 # 3.0, 3.035 ? # вертикальна відстань між осями
(розраховується)
r=0.84 # радіус кривошипа (залежить від ходу)
# примітка до табл. на с.42 [Архіпов] ?

def __init__(self):
    self.A=0.5*self.stroke # амплітуда
    self.ns=self.n/60.0 # частота, Гц
    self.omega=2*math.pi*self.ns # кутова частота, рад/с

def velAvg(self):
    """Середня швидкість, м/с"""
    return 2*self.stroke*self.n/60.0

def motionX(self, t):
    """Описує закон руху точки підвіски.
    Повертає координату X точки підвіски в момент часу t"""
    #!уточнити
    #v=self.A*self.omega*math.cos(self.omega*t+self.fi) # швидкість
    x=self.A*math.sin(self.omega*t+self.fi) # переміщення
    return x

class PU():
    """Клас свердловинної штангової насосної установки"""
    suckerRodString=None
    well=None
    pump=None
    outPres=100000.0 # 100000.0 тиск на гирлі в НКТ
    outTubePres=500000.0 # 500000.0 тиск в затрубному просторі
    heightDynamic = None # глибина динамічного рівня (якщо None, то за
замовчуванням)
    # зменшення heightDynamic зменшує тільки Fmax (див. Мищенко ф-ла 9.90)
    timeEnd=20.0 # кінець часу
    timeStep=0.1 # 0.01 крок часу

    def __init__(self):
        """Конструктор"""
        h=1500.0 # довжина колони
        sections=[(0.019, h/8.0, "сталь", 0.0),
                  (0.019, h/8.0, "сталь", 0.0),
                  (0.019, h/8.0, "сталь", 0.0),
                  (0.019, h/8.0, "сталь", 0.0),
                  (0.019, h/8.0, "сталь", 0.0),
                  (0.019, h/8.0, "сталь", 0.0),
                  (0.019, h/8.0, "сталь", 0.0),
                  (0.019, h/8.0, "сталь", 0.0)] # 0.2
        self.suckerRodString=SuckerRodString(PU=self,sections=sections) #
колона штанг

```

```

self.well=Well() # свердловина
self.pump=Pump(PU=self) # насос
self.pumpingUnit=PumpingUnit() # верстат-гойдалка
if self.heightDynamic==None:
    self.heightDynamic=self.suckerRodString.height()-100.0 # -100
орієнтовно
    r,m=self.balancing() # радіус і маса противаг
    self.pumpingUnit.p5['L']=r
    self.pumpingUnit.p5['m']=m

def fluidVolume(self):
    """Об'єм стовпа рідини над плунжером V=H*S"""
    return self.suckerRodString.height()*self.pump.pistonArea()

def fluidWeight(self): # не враховується інерція рідини
    """Вага рідини над плунжером w=V*ro*g.
    Діє на плунжер під час ходу вверх. Направлена вниз"""
    return self.fluidVolume()*self.well.density*9.81

def outPresForce(self):
    """Навантаження від тиску на гирлі.
    Діє на плунжер під час ходу вверх. Направлене вниз"""
    return self.pump.pistonArea()*self.outPres

def outTubePresForce(self):
    """Навантаження від тиску в затрубному просторі.
    Діє на плунжер під час ходу вверх. Направлене вверх"""
    return self.pump.pistonArea()*self.outTubePres

def outTubeFluidWeight(self):
    """Вага рідини в затрубному просторі.
    Діє на плунжер під час ходу вверх. Направлена вверх"""
    v=self.pump.pistonArea()*(self.suckerRodString.height()-
self.heightDynamic) # об'єм
    return v*self.well.density*9.81

def tubePressureFrictLossForse(self,v):
    """Навантаження від втрати тиску від тертя потоку по довжині НКТ
    Діє на плунжер під час ходу вверх. Направлене вниз. Не для моделі
Maplesim"""
    v=self.pump.volumeFlowRate(v)/self.well.sectionalArea() # середня
швидкість в НКТ
    D=self.well.diametr
    L=self.suckerRodString.length()
    nu=self.well.viscosity
    ro=self.well.density
    Re=abs(v)*D/nu # число Рейнольдса
    fL=64/Re # для ламінарного
    fT=0.316/math.pow(Re, 1.0/4) # для турбулентного (формула Блазіуса)
    if Re<=2000: # якщо ламінарний режим
        f=fL

```

```

elif Re>=4000: # якщо турбулентний режим
    f=fT
else: # в іншому випадку
    f=fL+(fT-fL)*(Re-2000)/2000
dP=(f*L/D)*v*v*ro/2 # втрати тиску
return self.pump.pistonArea()*dP # завжди додатна

def polishedRodForce(self):
    """Повертає кортеж (Fmin,Fmax) розрахований за наближеними
формулами
    Ці формули вірні, коли динамічний рівень = довжині спуску
    Тобто розраховується найбільш небезпечний випадок"""
    F1=self.suckerRodString.weight() # вага штанг в рідині
    F2=self.fluidWeight() # вага рідини
    k=(self.pumpingUnit.stroke*self.pumpingUnit.n**2)/1790
    Fmin=F1*(1-k) # формула Міллса
    Fmax=(F1+F2)*(1+k) # формула Кемлера
    return (Fmin,Fmax)

def sigmaPr(self,d):
    """Приведене напруження [Круман]"""
    A=math.pi*(d/2.0)**2 # площа перетину верхньої штанги
    Fmin,Fmax=self.polishedRodForce()
    return (Fmax-0.5625*Fmin)/A

def polishedRodForce2(self):
    """Повертає кортеж (Fmin,Fmax) розрахований за дуже наближеними
формулами"""
    F1=self.suckerRodString.weight() # вага штанг в рідині
    F2=self.fluidWeight() # вага рідини
    Fmin=F1
    Fmax=F1+F2
    return (Fmin,Fmax)

def balancing(self,Pmin=None,Pmax=None):
    """Параметри орієнтовного урівноважування [Архіпов]. Для верстатів
типу СК і СКД. Повертає радіус і сумарну масу противаг"""
    if Pmin==None and Pmax==None: # якщо не задано
        Pmin,Pmax=self.polishedRodForce() # розрахувати за наближеними
формулами
        Mp=750.0 # маса 1 противаги
        A0=52000.0 # робота сил ваги неурівноважених частин верстата
        S=self.pumpingUnit.stroke
        K_list=[2,4,6,8] # список кількості противаг
        A=Pmin*S/2+Pmax*S/2 # робота сил, що урівноважуються
        R_list=[(A-A0)/(2*9.81*Mp*k) for k in K_list] # список радіусів
противаг
        #print "Rp=", R_list
        Rmax=self.pumpingUnit.p1['L']-0.2 # максимальний радіус противаги
?уточнити
        for i,r in enumerate(R_list):

```



```

        if r<Rmax: break # вибирають радіус з мінімальною кількістю
противаг
        return r, K_list[i]*Mp

    def info(self): # друкує інформацію
        s=""
        s+="вага штанг в рідині %d Н\n"%self.suckerRodString.weight()
        s+="вага рідини %d Н\n"%self.fluidWeight()
        s+="дуже наближено Fmin=%d Fmax=%d\n"%self.polishedRodForce2()
        s+="наближено Fmin=%d Fmax=%d\n"%self.polishedRodForce()
        s+="приведене напруження
%d\n"%self.sigmaPr(d=self.suckerRodString.items[0].diametr)
        s+="діаметр плунжера %f\n"%self.pump.diametr
        s+="довжина колони %d\n"%self.suckerRodString.length()
        s+="жорсткість колони %d Н/м\n"%self.suckerRodString.stiffness()
        s+="маса колони %d кг\n"%self.suckerRodString.mass()

n0=math.sqrt(self.suckerRodString.stiffness()/self.suckerRodString.mass())
# власна частота стержня
    s+="власна частота колони %f рад/с\n"%n0
    s+="динамічний рівень %d м\n"%self.heightDynamic
    s+="вага рідини в затр просторі %f Н\n"%self.outTubeFluidWeight()
    s+="еквівалентний коефіцієнт опору матеріалу %f
\n"%self.suckerRodString.damping()
    s+="еквівалентний коефіцієнт опору матеріалу секції 1 %f
\n"%self.suckerRodString.items[0].materialRodResistance()
    # зі збільшенням довжини секції прямує до 0 (як і жорсткість)
    s+="жорсткість секції 1 %f
\n"%self.suckerRodString.items[0].stiffness()
    s+="маса секції 1 %f \n"%self.suckerRodString.items[0].mass()
    print s

if __name__ == '__main__':
    pu=PU()
    pu.info()

```

Лістинг В.2 – CAmodel.py – модель ШСНУ на основі абстрактних автоматів

```

# -*- coding: CP1251 -*-
import math, os
import PUmodel
import matplotlib.pyplot as plt
from matplotlib import rc
rc('font', family='Arial') # для підтримки Юнікоду

class Automaton(object):
    """Абстрактний автомат для моделювання руху вузла пружного стержня"""
    def __init__(self):
        """Конструктор автомата"""
        self.x=None # координата

```

```

self.x0=None # початкова координата
self.prevx=None # попередня координата
self.bc=None # гранична умова (0 - нерухомий)
self.f=0.0 # зовнішня сила
self.fs=0.0 # вага секції знизу
self.left=None # верхній сусід
self.right=None # нижній сусід
self.delta=None # початкова відстань до нижнього сусіда
(rSection.length)
self.k=None # коефіцієнт жорсткості секції знизу
(rSection.stiffness())
self.c=0.0 # коефіцієнт опору матеріалу секції знизу
self.m=1.0 # маса секції знизу
self.a=0.0 # прискорення
self.v=0.0 # швидкість
self.prevv=0.0 # попередня швидкість
self.eps=0.0 # нев'язка
self.dx=1.0 # крок наближення
self.area=None # площа поперечного перетину секції знизу
(rSection.area())
self.domain=None # об'єкт класу CA_model
self.rSection=None # секція знизу
self.alfa=0.0 # кут відхилення свердловини від вертикалі біля
автомата

def rule(self):
    """Правило поведінки автомата"""

    if self.bc!=None: # якщо задана гранична умова
        self.x=self.bc # визначити x

    if self.domain.t!=0.0: # якщо час не 0
        self.v=self.velocity() # розрахувати швидкість
        self.a=self.acceleration() # розрахувати прискорення
    else: self.v=self.a=0.0

    if self.bc!=None: # якщо задана гранична умова
        return # вийти (не потрібно знаходити x)

    Fk=self.rightForce()-self.leftForce() # сили пружності
    Fv=self.rightForce2()-self.leftForce2() # сили демпфування
    Fa=self.dynamicForce() # сила інерції
    # зовнішні сили

F=self.fs+self.f+self.pistonForce()+self.frictionForce()+self.hydrodynamicR
esistanceForce()

# рівновага усіх сил біля автомата: m*a+c*v+k*u=F
# якщо сила>0, то вона направлена вниз
# Напрямок осі x:
# [0]---[1]---[2]--->+x,+v,+a,+f,нижній

```

```

# обчислити нев'язку
self.eps=Fa+Fv+Fk-F
#print "eps=",self.eps

# якщо нев'язка відрізняється від 0
if self.eps>0:
    self.x=self.x-self.dx # збільшити x на малу величину
if self.eps<0:
    self.x=self.x+self.dx # зменшити x на малу величину

def velocity(self):
    return (self.x-self.prevx)/self.domain.dt # швидкість

def acceleration(self):
    return (self.v-self.prevv)/self.domain.dt # прискорення

def leftForce(self):
    """Повертає значення сили пружності секції зверху"""
    if self.left: # якщо є лівий сусід
        dLeftBegin=self.left.delta # початкова довжина пружини зверху
        dLeft=self.x-self.left.x # довжина пружини зверху
        fLeft=(dLeftBegin-dLeft)*self.left.k # сила пружності
    else:
        fLeft=0.0
    return fLeft

def leftForce2(self):
    """Повертає значення сили демпфування секції зверху"""
    if self.left: # якщо є верхній сусід
        vrel=self.v-self.left.v # відносна швидкість
        fLeft=-self.left.rSection.materialRodResistance(None)*vrel
    else:
        fLeft=0.0
    return fLeft

def rightForce(self):
    """Повертає значення сили пружності секції знизу"""
    if self.right: # якщо є нижній сусід
        dRightBegin=self.delta # початкова довжина пружини знизу
        dRight=self.right.x-self.x # довжина пружини знизу
        fRight=(dRightBegin-dRight)*self.k # сила пружності
    else:
        fRight=0.0
    return fRight

def rightForce2(self):
    """Повертає значення сили демпфування секції знизу"""
    if self.right: # якщо є нижній сусід
        vrel=self.right.v-self.v # відносна швидкість
        fRight=-self.rSection.materialRodResistance(None)*vrel
    else:

```

```

        fRight=0.0
    return fRight

def dynamicForce(self):
    """Сила інерції секції  $F=m*a$ """
    return self.m*self.a

def dynamicFluidForce(self):
    """Сила інерції рідини. Діє на плунжер"""

k=self.domain.PU.pump.pistonArea()/self.domain.PU.well.sectionalArea()
a=self.a*k # прискорення рідини менше прискорення плунжера у k раз
return -a*sum(self.domain.PU.suckerRodString.fluidMassList())

def normalForce(self):
    """Сума нормальних сил у вузлі від ваги, верхнього і нижнього
натягу"""
    F1=self.f*math.tan(self.alfa) # нормальна сила від ваги (f - осьова
сила від ваги)
    F2=F3=0.0
    if self.left:
        F2=self.leftForce()*math.sin(self.an1) # нормальна сила від
верхнього натягу
    if self.right:
        F3=self.rightForce()*math.sin(self.an2) # нормальна сила від
нижнього натягу
    return F1+abs(F2+F3) # якщо нехтувати зазором між стінкою НКТ і
вузлом

def frictionForce(self):
    """Сила тертя секції"""
    # сила тертя на інтервалі
    #Ft=C*(Psh*math.sin(a1)-F12*math.sin(a12-a1)) #
    #Ft=C*(Psh*math.sin(a1)+2*F12*math.tan(a2/2-a1/2)) #  $a_{12}-a_1=a_2/2-$ 
a1/2
    #Ft=C*(Psh*math.sin(a1)+F12*(a2-a1)) # формула Песляка

    v=self.v # швидкість
    Fn=self.normalForce() # нормальна сила # для перевірки напрямку
введіть тут 1000.0
    fc=0.16 #0.25 # коефіцієнт тертя ковзання (сталь-сталь зі
змащуванням)
    #F=-fc*abs(Fn)*math.tanh(v/0.01) # спрощена модель

    Fc=abs(Fn)*fc # сила тертя Кулона
    fmax=0.2 # коефіцієнт тертя ковзання (сталь-сталь з малим
змащуванням)
    Fmax=abs(Fn)*fmax # максимальна сила тертя ковзання
    vs=0.1 # коеф. швидкості ковзання Штрибека
    n=1.0 # степінь згасання (Decay exponent)
    # сумарна сила тертя

```

```

F=-math.tanh(v/0.01)*(Fc+(Fmax-Fc)*math.exp(-(abs(v)/vs)**n))
return F

def hydrodynamicResistanceForce(self):
    """Сила гідродинамічного опору секції"""
    v=self.v # швидкість
    if v<=0: # якщо хід вверх

k=self.domain.PU.pump.pistonArea()/self.domain.PU.well.sectionalArea()
    vu=v-v*k # швидкість штанг відносно рідини
    F=-self.c*vu*10.0 #увага!!! збільшено
    # F=0.0 # якщо швидкості штанг і рідини рівні
    else: # якщо хід вниз
        F=-self.c*v*10.0 #увага!!! збільшено
    return F*math.tanh(abs(v)/0.01) # згладжуємо біля 0

def pistonForce(self):
    """Сума зовнішніх сил на плунжері насоса"""
    v=self.v # швидкість
    F=0.0 # для усіх вузлів крім останнього
    if self.right!=None: return F # якщо не останній вузол (насос)

    if v<0: # якщо рух плунжера вгору
        # направлені вниз:
        F+=self.domain.PU.fluidWeight()
        F+=self.domain.PU.outPresForce()
        F+=abs(self.domain.PU.tubePressureFrictLossForse(v)) # тут abs
        F+=self.dynamicFluidForce()
        # направлені вверх:
        F+=-self.domain.PU.outTubePresForce()
        F+=-self.domain.PU.outTubeFluidWeight()
    if v!=0:
        # наступні сили діють під час РУХУ (!) вгору і вниз
        # і направлені проти руху плунжера
        F+=-math.copysign(1,
v)*abs(self.domain.PU.pump.valveHydroResistForce(v))
        F+=-math.copysign(1,
v)*self.domain.PU.pump.pistonFrictionForce() # модель тертя плунжера

    F=F*math.tanh(abs(v)/0.01) # ф-ція згладжування біля 0 (F=тах, якщо
v=0.01)
    #або F=F*(1.0-math.exp(v/0.01))
    return F

def rightStress(self):
    """Повертає значення напруження в пружині справа (внизу)"""
    if self.right:
        return self.rightForce()/self.area
    else:
        return 0.0

```

```

class CA_model(object):
    """Система автоматів (пружний стержень)"""
    def __init__(self,PU):
        """Створює систему автоматів за об'єктом класу PU (ШЧНУ)
        Схема системи автоматів:
                секція1                                секція2
        0-----0-----0
        m0=0      d0      m1      d1      m2
        c0=0      k0      c1      k1      c2
        f0      a0      f1      a1      f2
        alfa0=0  ans0    alfa1    ans1    alfa2
        Перший автомат (0) допоміжний.
        """

        self.PU=PU # ШЧНУ
        rs=PU.suckerRodString.items # список секцій
        n=len(rs)+1 # кількість вузлів (автоматів)
        # Перший автомат (0) допоміжний.
        X=[0.0]+PU.suckerRodString.xList() # список координат вузлів
        M=[0.0]+PU.suckerRodString.massList() # маса вузлів
        Fs=[0.0]+PU.suckerRodString.weightForceList() # список сил від ваги
секцій
        C=[0.0]+PU.suckerRodString.hydrodynamicRodResistanceList() #
гідродинамічні опори секцій

        AN=[0.0]+PU.suckerRodString.wellAngleList()
        #ANS=PU.suckerRodString.secAngleList()
        AN1=[None]+pu.suckerRodString.angleTop()
        AN2=[0.0]+pu.suckerRodString.angleBottom()

        self.dt=self.PU.timeStep # крок часу
        self.T=self.timeList(PU.timeEnd, PU.timeStep) # список значень часу
[t0,t1,t2,...]
        self.t=self.T[0] # поточне значення часу (0.0)

        BCdict={} # словник граничних умов (див. self.BCList)
        # закон руху точки підвіски
        BCdict[0]=[PU.pumpingUnit.motionX(t) for t in self.T]
        self.BC=CA_model.BCList(BCdict, n, self.T, default=None) # список
історії граничних умов [[bc0,bc1,bc2,...],...]

        F=[0.0]*n # список інших зовнішніх сил (якщо потрібно)
        Fdict=dict(zip(range(n),F)) # словник сил (див. self.BCList)
        self.F=CA_model.BCList(Fdict, n, self.T, default=0.0) # список
історії зовнішніх сил [[f0,f1,f2,...],...]

        self.p=[Automaton() for i in range(n)] # список автоматів

        # для кожного автомата, крім останнього, визначити властивості
        for i in range(n-1):
            self.p[i].rSection=rs[i]

```

```

self.p[i].k=rs[i].stiffness()
self.p[i].delta=rs[i].length
self.p[i].area=rs[i].area()

# задаємо інші властивості усіх автоматів
for i in range(n):
    self.p[i].bc=self.BC[0][i]
    self.p[i].f=self.F[0][i]
    self.p[i].fs=Fs[i]
    self.p[i].m=M[i]
    self.p[i].c=C[i]
    self.p[i].alfa=AN[i]
    self.p[i].x=X[i]
    self.p[i].x0=X[i]
    self.p[i].domain=self
    self.p[i].an1=AN1[i]
    self.p[i].an2=AN2[i]

for i in range(n): # задаємо сусідів
    if i!=0: # якщо не перший
        self.p[i].left=self.p[i-1] # визначити верхнього сусіда
    if i!=len(self.p)-1: # якщо не останній
        self.p[i].right=self.p[i+1] # визначити нижнього сусіда

# словник історій результатів
# наприклад, список історії значень x [[x0,x1,x2,...],...]

self.history={'T':self.T,'X':[],'V':[],'A':[],'F1':[],'Fr':[],'Sr':[]}

def run(self):
    """Еволюція системи автоматів в конкретний момент часу. Одна з
    можливих реалізацій. Повільна, але не потребує сторонніх алгоритмів."""
    eps=0.01*len(self.p) # похибка розрахунку
    e=eps+1 # сума нев'язок
    eprev=eps+1 # попередня сума нев'язок
    # початковий крок наближення (залежить від delta)
    dx=min([a.delta for a in self.p if a.delta!=None])/10.0 # !!!
    Змінювати в межах 10-100
    niter=0 # кількість ітерацій (для оптимізації алгоритму)
    while e>eps: # поки сума нев'язок > eps
        for a in self.p: # для кожного автомата
            a.dx=dx # змінити крок наближення
            a.rule() # застосувати правило поведінки автомата
        e=sum([abs(a.eps) for a in self.p]) # сума нев'язок
        #print e
        if e>=eprev: # якщо сума нев'язок >= попередньої
            dx=dx*0.935 # зменшити крок !!! Змінювати в межах 0.1-0.99
            #print "dx=",dx
        eprev=e # запам'ятати суму нев'язок
        niter=niter+1
    #print niter

```

```

#         a=self.p[1]
#         print 'lf=',a.leftForce()
#         print 'rf=',a.rightForce()
#         print 'lf2=',a.leftForce2()
#         print 'lvrel=',a.v-a.left.v
#         print 'rf2=',a.rightForce2()
#         print 'rvrel=',a.right.v-a.v

def runDynamic(self):
    """Динамічна еволюція системи автоматів"""
    # початкові значення координат, швидкостей і прискорень
    for a in self.p: # для кожного автомата
        a.prevx=a.x # попередня координата
        a.prevv=0.0 # попередня швидкість
    for i,t in enumerate(self.T): # для кожного значення часу
        self.t=t # поточне значення часу
        print "solve t=",t
        for a,f,bc in zip(self.p, self.F[i], self.BC[i]): # для кожного
автомата
            a.f=f # визначити силу в момент часу t
            a.bc=bc # визначити граничну умову в момент часу t
            self.run() # знайти урівноважений стан
            for a in self.p: # для кожного автомата
                a.prevx=a.x # запам'ятати поточні x та v
                a.prevv=a.v
            self.appendHistory()# запам'ятати історію
            self.writeHistory()

def appendHistory(self):
    """Додає в списки історії список значень для поточного часу"""
    self.history['X'].append([a.x for a in self.p])
    self.history['V'].append([a.v for a in self.p])
    self.history['A'].append([a.a for a in self.p])
    self.history['F1'].append([a.leftForce() for a in self.p])
    self.history['Fr'].append([a.rightForce() for a in self.p])
    self.history['Sr'].append([a.rightStress() for a in self.p])

def draw(self):
    """Рисую положення автоматів псевдографікою (в текстовому
режимі)"""
    s=""
    prev=0
    for a in self.p:
        s+="-"*int(round(a.x-prev)-1)+"0"
        prev=a.x # координата попереднього автомата
    print s
    #time.sleep(1)

def drawPlot(self, itemIndexes, key, x0scale=1, toFile=True):
    """Рисую залежність величини x[i] від t, де i - індекс автомата

```



```

itemIndexes - список індексів автоматів
key - ключ історії x
x0scale - масштаб відстаней між першими точками кривих по осі X
(1...0):
    1 - реальний, 0 - відстані рівні 0.
    Крива для 0-го вузла не зміщується.
    Використ. для покращення візуалізації переміщень вузлів,
    якщо вони малі відносно відстаней між вузлами"""
plt.figure()
plt.gca().invert_yaxis() # обернути вісь y
T=self.history['T'] # список значень часу
x00=self.history[key][0][0] # x 0-го вузла для t=0
for i in itemIndexes: # для кожного індексу
    x0i=self.history[key][0][i] # x i-го вузла для t=0
    x0=(1-x0scale)*(x0i-x00) # величина зміщення по осі X
    X=[x[i]-x0 for x in self.history[key]] # спис. істор. вузла "i"
    plt.plot(T, X, 'k-') # (T, X, 'r-', T, X,'ro')
plt.grid(True)
plt.xlabel(u't, с') # надпис осі x
plt.ylabel(key) # надпис осі y
if toFile:
    fileName='CAmodelPlot{0}{1}.png'.format(itemIndexes,key)
    plt.savefig(fileName)
else:
    plt.show()

def drawDynamometerCard(self, itemIndexes, keyX, keyY, signY=-1,
toFile=True, tstart=None, tend=None):
    """Рисує динамограму - залежність Y(X), наприклад,
сила(переміщення)"""
    plt.figure()
    plt.gca().invert_xaxis() # обернути вісь x
    plt.grid(True)
    plt.xlabel(u"S, м") #надпис осі x
    plt.ylabel(u"F, Н") #надпис осі y
    for i in itemIndexes: # для кожного вузла
        # індекси початкового і кінцевого часу в історії
        tstarti=self.history['T'].index(tstart) if tstart in self.T
else None
        tendi=self.history['T'].index(tend)+1 if tend in self.T else
None
        X=[x[i] for x in self.history[keyX][tstarti:tendi]] # список
історії X вузла "i"
        x0=self.history[keyX][0][i] # початкове значення
        X=[x-x0 for x in X] # відносно початкової координати
        Y=[signY*y[i] for y in self.history[keyY][tstarti:tendi]] #
список історії Y вузла "i"
        plt.plot(X, Y, 'k-')
    if toFile:
        fileName='CAmodelDynCard{0}{1}.png'.format(keyX,keyY)
        plt.savefig(fileName)

```

```

else:
    plt.show()

def writeHistory(self):
    """Зберігає історію в бінарний файл"""
    import pickle
    f = open('CAmodelHistory.pkl', 'wb')
    pickle.dump(self.history, f) # зберегти історію
    f.close()

def readHistory(self):
    """Читає історію з бінарного файлу"""
    import pickle
    fileName='CAmodelHistory.pkl'
    if not os.path.isfile(fileName): # якщо файлу немає
        print "No file "+fileName
        return False # вийти, повернувши False
    f = open(fileName, 'rb')
    self.history=pickle.load(f) # завантажити історію
    f.close()
    print "Warning! Old results"
    return True

def writeCSV(self,itemIndexes,key):
    """Записує значення історій у файл CSV.
    itemIndexes - список індексів автоматів
    key - ключ історії"""
    import csv
    csv_file=open("RodHistory_"+key+".csv", "wb")
    writer = csv.writer(csv_file,delimiter = ';')
    for time,data in zip(self.T,self.history[key]):
        X=[data[i] for i in itemIndexes] # лише дані за вказаними
індексами
        writer.writerow([time]+X) # записати рядок
    csv_file.close()

def info(self):
    """Виводить значення усіх атрибутів автоматів"""
    attrs=["rSection","k","delta","area","bc","x","prevx","v","prevv",\
        "a","m","alfa","an1","an2","c","eps","f"]
    for attr in attrs:
        print attr+"=", [a.__getattr__(attr) for a in self.p]
    print "leftForce=", [a.leftForce() for a in self.p]
    print "rightForce=", [a.rightForce() for a in self.p]

@staticmethod
def timeList(timeEnd,timeStep):
    """Повертає список значень часу"""
    nTimeSteps=int(timeEnd/timeStep)
    return [x*timeStep for x in range(nTimeSteps+1)]

```

```

@staticmethod
def BCList(BC, n, timeList, default=None):
    """
    Повертає список граничних умов (або сил) для автоматів 1,2,... у
    вигляді:
    [[bc0,bc1,...],...],
    де перший список відповідає першому значенню часу.

    BC - словник граничних умов, де ключі - індекси КА, а значеннями
    можуть бути:
    1.Дійсні (постійне значення): 0.0
    2.Словники пар час-значення: {0.0:1.0,0.05:2.0}
    3.Списки історії значень:
    [1.0,2.0,3.0,4.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0]
    або: [math.sin(x) for x in range(nt)]
    n - кількість автоматів
    timeList - список значень часу
    default - значення списків за замовчуванням"""

    nt=len(timeList) # кількість значень часу
    res=[] # результат
    for i in range(nt):
        res.append([default]*n) # заповнюємо res різними(!) списками

    for k in BC: # для кожного заданого вузла
        if type(BC[k]).__name__=='dict': # якщо задана для конкретних
інтервалів часу
            i=0
            prev=None # попередній
            for t in res: # для кожного значення часу
                if BC[k].has_key(timeList[i]):
                    t[k]=BC[k][timeList[i]]
                    prev=t[k]
                else:
                    t[k]=prev
            i=i+1
        elif type(BC[k]).__name__=='list': # якщо задана для кожного
інтервалу часу
            i=0
            for t in res: # для кожного значення часу
                t[k]=BC[k][i]
                i=i+1
        else: # якщо константа
            for t in res: # для кожного значення часу
                t[k]=BC[k]

    #print res
    return res

def resultsCAmodel(self):
    """Виводить результати автоматної моделі"""

```

```
if self.history['X']==[]: # якщо немає результатів
    oldHistExist=self.readHistory() # прочитати старі результати
    if not oldHistExist: return

tend=self.T[-1]
T=1.0/self.PU.pumpingUnit.ns # період
tstart=tend-T # тільки останній період
# знайти найближче значення до tstart у списку self.T
dt=[abs(t-tstart) for t in self.T]
tstart=self.T[dt.index(min(dt))]
self.drawDynamometerCard([0,1,2,3,4,5,6,7], "X", "Fr",
tstart=tstart, tend=tend)
self.drawPlot([0,1,2,3,4,5,6,7,8], 'X', x0scale=0.001)
self.drawPlot([0,1,2,3,4,5,6,7], 'Sr')

if __name__ == '__main__':
    pu=PUmodel.PU()
    model=CA_model(pu)
    #model.info()
    #model.run() # статика
    #model.info()

    model.runDynamic() # динаміка
    #model.info()
    model.resultsCAmodel()
```

ДОДАТОК Г

Модель ШСНУ мовою Modelica

Код також доступний в GitHub (vkorey/PU. URL: <http://github.com/vkorey/PU>).

Лістинг Г.1 – Main.mo - Modelica-код без анотацій

```

model Main
  import Modelica.Constants.inf;
  import Pi=Modelica.Constants.pi;
  import pi=Modelica.Constants.pi;

  inner parameter Modelica.SIunits.Density rhoFluid = FP1.rhoFluid;
  inner parameter Modelica.SIunits.BulkModulus EFluid = FP1.EFluid;
  inner parameter Modelica.SIunits.KinematicViscosity nuFluid =
  FP1.nuFluid;
  public inner Maplesoft.Multibody.World
  world(gravityDir=Maplesoft.Multibody.Selectors.UnitVector.negY,
  gravityAcc=9.81);
  public Modelica.Blocks.Sources.Sine S1(amplitude=1.5,
  freqHz=.108333333333, phase=3.14159265359, offset=0, startTime=200) ;
  public Modelica.Mechanics.Translational.Sources.Position
  P1(useSupport=false, exact=true, f_crit=50) ;
  public Modelica.Mechanics.Translational.Components.Fixed F3(s0=0) ;
  public Maplesoft.Hydraulics.Basic.HydraulicCylinder
  HC1(A=0.113411494795e-2) ;
  public Maplesoft.Hydraulics.Basic.CheckValve CV2(Ropen=178.945936903,
  Gclosed=0.10e-11, Startclosed=false) ;
  public Maplesoft.Hydraulics.Basic.CheckValve CV1(Ropen=178.945936903,
  Gclosed=0.10e-11, Startclosed=true) ;
  public Maplesoft.Hydraulics.Basic.AtmosphericPressure AP1(P=.0) ;
  public Maplesoft.Hydraulics.Basic.CircularPipe CP1(D=.1003, L=1500.0,
  epsilon=0.15e-4, ReL=2000, ReT=4000, rho=rhoFluid, nu=nuFluid) ;
  public Maplesoft.Hydraulics.Basic.AtmosphericPressure AP2(P=100000.0) ;
  public inner Maplesoft.Hydraulics.FluidProperties FP1(rhoFluid=1000.0,
  EFluid=800000000.0, nuFluid=0.2e-5) ;
  public MapleSimStandaloneSubsystem n0 ;
  public Maplesoft.Hydraulics.Basic.FluidInertia FI1(A=0.761764747263e-2,
  L=1500.0, rho=rhoFluid) ;
  public Maplesoft.Mechanical.Basic.TransFriction TF1(fs=702.75, fc=562.2,
  d=0, vs=.1, n=1, v0=0.1e-1) ;
  public MapleSimStandaloneSubsystem n1 ;
  public MapleSimStandaloneSubsystem n2 ;
  public MapleSimStandaloneSubsystem n3 ;
  public MapleSimStandaloneSubsystem n4 ;

```

```

public MapleSimStandaloneSubsystem n5 ;
public MapleSimStandaloneSubsystem n6 ;
public MapleSimStandaloneSubsystem n7 ;
public Maplesoft.Hydraulics.Basic.FixedPressure FP2(P=0.13234e8) ;
equation
  connect(S1.y, P1.s_ref) ;
  connect(CV2.portA, AP1.portA) ;
  connect(CV2.portB, CV1.portA) ;
  connect(F3.flange, HC1.flange_b) ;
  connect(HC1.portA, CV2.portB) ;
  connect(P1.flange, n0.b1) ;
  connect(CP1.portB, FI1.portA) ;
  connect(FI1.portB, AP2.portA) ;
  connect(HC1.flange_a, TF1.flange_b) ;
  connect(TF1.flange_a, F3.flange) ;
  connect(n1.b1, n0.a1) ;
  connect(n2.b1, n1.a1) ;
  connect(n3.b1, n2.a1) ;
  connect(n3.a1, n4.b1) ;
  connect(n4.a1, n5.b1) ;
  connect(n5.a1, n6.b1) ;
  connect(n6.a1, n7.b1) ;
  connect(n7.a1, HC1.flange_a) ;
  connect(FP2.portB, CP1.portA) ;
  connect(CV1.portB, FP2.portA) ;
end Main;
model VerticalPipe
  extends Maplesoft.Icons.CustomComponent;
  parameter Real z = 0 "z";
  parameter Real rho = 1000 "rho";
  parameter Real g = 9.81 "g";
  Real Pin;
  Real Qin;
  Real dP;
  Real Pout;
  Real Qout;
  Maplesoft.Hydraulics.Interfaces.Port_a portA;
  Maplesoft.Hydraulics.Interfaces.Port_a portB;
equation
  Qin = Qout;
  dP = rho * g * z;
  Pin - Pout = dP;
  portA.p = Pin;
  portA.q = Qin;
  portB.p = Pout;
  portB.q = -Qout;
end VerticalPipe;
model MapleSimStandaloneSubsystem
  import Modelica.Constants.inf;
  import Pi=Modelica.Constants.pi;
  import pi=Modelica.Constants.pi;

```

```

    public Modelica.Mechanics.Translational.Components.Mass
M1(m=425.29310548, L=0) ;
    public Maplesoft.Sensors.SpeedSensor SS1(dimension="Velocity",
fromUnit="m/s", toUnit="m/s", scale=1.0, shift=.0) ;
    public Modelica.Mechanics.Translational.Sources.Force
F2(useSupport=false) ;
    public Modelica.Blocks.Math.BooleanToReal BTR2(realTrue=1.0,
realFalse=.0) ;
    public Modelica.Blocks.Math.Gain G2(k=46.9936711616) ;
    public Modelica.Blocks.Logical.LessThreshold LT1(threshold=0) ;
    public Modelica.Mechanics.Translational.Sources.ConstantForce
CF1(useSupport=false, f_constant=-3650.60969416) ;
    public Modelica.Blocks.Math.Product P3 ;
    public Modelica.Mechanics.Translational.Interfaces.Flange_b b1 ;
    public Modelica.Mechanics.Translational.Interfaces.Flange_a a1 ;
    public Maplesoft.Mechanical.Basic.TransFriction TF1(fs=0., fc=0.,
d=46.9936711616, vs=.1, n=1, v0=0.1e-1) ;
    public Modelica.Mechanics.Translational.Components.Fixed F1(s0=0) ;
    public Modelica.Blocks.Math.Gain G3(k=100) ;
    public Modelica.Blocks.Math.Tanh T1 ;
    public Modelica.Blocks.Math.Product P1 ;
    protected Modelica.Blocks.Interfaces.RealOutput RO_1 ;
    public Maplesoft.Sensors.ForceSensor FS1(dimension="Force",
fromUnit="N", toUnit="N", scale=1.0, shift=.0) ;
    public Maplesoft.Sensors.ForceSensor FS2(dimension="Force",
fromUnit="N", toUnit="N", scale=1.0, shift=.0) ;
    public Modelica.Blocks.Math.Gain G1(k=.0) ;
    public Modelica.Blocks.Math.Gain G4(k=.0) ;
    public Modelica.Blocks.Math.Add A1(k1=1, k2=1) ;
    public Modelica.Blocks.Math.Abs abs1_1 ;
    public Modelica.Blocks.Math.Add A2(k1=1, k2=-1) ;
    public Modelica.Mechanics.Translational.Components.SpringDamper
SD1(s_nominal=0.10e-3, c=317552.185425, d=14849.8907517, s_rel0=0) ;
    //public outer Maplesoft.Hydraulics.FluidProperties FP1;
equation
    connect(M1.flange_a, SS1.F) ;
    connect(CF1.flange, M1.flange_a) ;
    connect(LT1.y, BTR2.u) ;
    connect(BTR2.y, G2.u) ;
    connect(F2.flange, M1.flange_a) ;
    connect(F1.flange, TF1.flange_a) ;
    connect(G3.y, T1.u) ;
    connect(P1.y, F2.f) ;
    connect(TF1.flange_b, M1.flange_a) ;
    connect(SS1.RO, LT1.u) ;
    connect(P3.u2, G2.y) ;
    connect(G3.u, RO_1) ;
    connect(SS1.RO, RO_1) ;
    connect(RO_1, P3.u1) ;
    connect(P1.u1, T1.y) ;

```

```

connect(P3.y, A2.u1) ;
connect(P1.u2, A2.y) ;
connect(A2.u2, abs1_1.y) ;
connect(A1.y, abs1_1.u) ;
connect(G4.y, A1.u2) ;
connect(G1.y, A1.u1) ;
connect(FS1.R0, G1.u) ;
connect(FS1.F1, b1) ;
connect(FS2.F1, M1.flange_a) ;
connect(FS2.R0, G4.u) ;
connect(FS2.F2, a1) ;
connect(SD1.flange_b, FS1.F2) ;
connect(M1.flange_b, SD1.flange_a) ;
end MapleSimStandaloneSubsystem;

```

Лістинг Г.2 – MAPLESIMmodel.py – модель ШЧНУ для Maplesim 7

```

# -*- coding: cp1251 -*-
import math, os
import matplotlib.pyplot as plt
from matplotlib import rc
rc('font', family='Arial') # для підтримки Юнікоду
import PUmudel, maplepy

# Бажано задавати SD1_s_rel0. Визначити його можна шляхом моделювання
непрацюючої установки (задавши S1_T0>0).

class Maplesim_model():
    """Модель ШЧНУ у Maplesim
    Потребує файлу моделі PUmudel.msim"""
    pu=None # об'єкт класу PUmudel.PU
    ms=None # об'єкт класу maplepy.MapleInterface4Maplesim

    def prepareParams(self):
        """Підготовлює параметри Maplesim моделі"""
        pu=self.pu # ШЧНУ
        rs=pu.suckerRodString.items # список секцій
        n=len(rs) # кількість секцій

        M=pu.suckerRodString.massList() # маса секцій
        M2=pu.suckerRodString.fluidMassList() # маса рідини біля секцій
        C2=pu.suckerRodString.hydrodynamicRodResistanceList()
        W=pu.suckerRodString.weightForceList()
        C=[r.stiffness() for r in rs]
        D=[r.materialRodResistance() for r in rs]
        v=pu.pumpingUnit.velAvg()

        AN=pu.suckerRodString.wellAngleList() #список кутів alfa (довж. n)

```



```

ANS=pu.suckerRodString.secAngleList() # кути відх. секцій від верт.
AN1=pu.suckerRodString.angleTop()
AN2=pu.suckerRodString.angleBottom()

#         for p in 'M M2 C2 W C D v AN ANS AN1 AN2'.split():
#         print p, eval(p) # надрукувати

# підготувати параметри для моделі Maplesim
modelParams={'S1_amplitude':pu.pumpingUnit.A, # параметри верстата
'S1_freqHz':pu.pumpingUnit.ns,
'S1_phase':pu.pumpingUnit.fi,
'HC1_A':pu.pump.pistonArea(), # парам. гідравл. част.
'CV1_Ropen':pu.pump.valveLossCoefficient(v),
'CV2_Ropen':pu.pump.valveLossCoefficient(v),
'CP1_D':pu.well.diametr, # ?
'CP1_L':pu.suckerRodString.length(),
#Замість VerticalPipe1 можна використовувати FP2_P
#'VerticalPipe1_z':pu.suckerRodString.height(),
'AP1_P':0.0, # тиск на вході
'AP2_P':pu.outPres, # тиск на гирлі
# гідростатичний тиск, який діє на плунжер під час
ходу вверх
# враховується також гідростат. тиск в затр. просторі,
який діє з другої сторони плунжера
'FP2_P':pu.fluidWeight()/pu.pump.pistonArea()-
pu.outTubeFluidWeight()/pu.pump.pistonArea()-pu.outTubePres,
'rhoFluid':pu.well.density,
'nuFluid':pu.well.viscosity,
# середня площа перетину труби
'FI1_A':
(sum(M2)/pu.well.density)/pu.suckerRodString.length(),
# ввести FI1_L=0, якщо не моделюється інерція рідини
'FI1_L':pu.suckerRodString.length(),
'TF1_fs':1.25*pu.pump.pistonFrictionForce(),
'TF1_fc':pu.pump.pistonFrictionForce()
}
#W[0]+=15000.0 # !!! сила тертя в ущільненні
for i in range(n): # для кожної секції ni (i=0, 1, 2...)
    modelParams['n'+str(i)+'_M1_m']=M[i]
    # !!! увага тут і нижче коефіцієнти демпфування збільшено
    modelParams['n'+str(i)+'_G2_k']=10*C2[i] # узгодити з TF1_d!!!
гасить вібрації під час ходу вниз
    modelParams['n'+str(i)+'_CF1_f_constant']=-W[i] # увага знак
    modelParams['n'+str(i)+'_TF1_fs']=0.2*W[i]*math.tan(AN[i]) #
сила тертя від ваги max
    modelParams['n'+str(i)+'_TF1_fc']=0.16*W[i]*math.tan(AN[i]) #
сила тертя від ваги
    modelParams['n'+str(i)+'_TF1_d']=10*C2[i] # коеф. в'язкого
тертя (узгодити з G2_k!!!) гасить вібрації
    modelParams['n'+str(i)+'_G1_k']=0.16*math.sin(AN1[i]) # множник
сили тертя від верхнього натягу

```

```

        modelParams['n'+str(i)+'_G4_k']=0.16*math.sin(AN2[i]) # множник
сили тертя від нижнього натягу
        modelParams['n'+str(i)+'_SD1_c']=C[i]
        modelParams['n'+str(i)+'_SD1_d']=D[i] # повинен бути більшим,
оскільки існує ще конструкційне демпфування

#         for p in sorted(modelParams.keys()):
#             print p, modelParams[p]

self.modelParams=modelParams

def createMaplesimModel(self, execute=False, resFileName=None):
    """Створює Maplesim модель. Виконує її, якщо execute=True"""
    ms=maplepy.MapleInterface4Maplesim()
    ms.path=os.getcwd().replace('\\', '/')
    ms.filenameMaplesim="PUmodel.msim"
    ms.paramDictMaplesim=self.modelParams # {} якщо не передавати
нічого
    #print ms.getCode()
    if execute: ms.execute=True
    if resFileName: ms.resultCSVfile=resFileName
    ms.runMaple()
    self.ms=ms

def drawDynamometerCard(self, S0=1.0):
    """Створює файл-рисунок з динамограмою
Першою пробою в PUmodel.msim має бути проба Length і Force
на місці динамометра (верхня штанга)
S0 - площа поперечного перетину верхньої штанги (якщо S0=1,
виводить силу)"""
    results=self.ms.readCSVfile(5) # список результатів

    # рисує тільки останній період
    T=1.0/self.modelParams['S1_freqHz'] # період
    tend=results[-1][0] # кінцевий час
    tstart=tend-T # початок останнього періоду

    f=[] # сила або напруження
    s=[] # переміщення
    f1=[]
    s1=[]
    for res in results: # для кожного часу
        if res[0]>=tstart: # починати з часу tstart
            f.append(res[1]/S0) # !!!напруження
            s.append(res[2]) # переміщення
            f1.append(res[3])
            s1.append(res[4])
    plt.plot(s, f, 'k-')
    plt.plot(s1, f1, 'b-')
    plt.grid(True)
    plt.xlabel(u"S, м") #надпис осі x

```

```

if S0==1.0:
    plt.ylabel(u"F, Н") #надпис осі у
else:
    plt.ylabel("s, Pa") #надпис осі у
plt.savefig("MaplesimDynCard.png")
return min(f),max(f),min(s1),max(s1),min(f1),max(f1)

def drawDynamometerCardMulti(self,
filenames=[],S0L=[],freqL=[],styleL=[]):
    """Створює файл-рисунок з динамограмою для багатьох файлів-
результатів
    див. документацію drawDynamometerCard"""
    ms=maplepy.MapleInterface4Maplesim()
    ms.path=os.getcwd().replace('\\', '/')
    for name,S0,freq,style in zip(filenames,S0L,freqL,styleL):
        ms.resultCSVfile=name
        results=ms.readCSVfile(3) # список результатів

        # рисує тільки останній період
        T=1.0/freq # період
        tend=results[-1][0] # кінцевий час
        tstart=tend-T # початок останнього періоду

        f=[] # сила або напруження
        s=[] # переміщення

        delta=0
        #if name=='13.csv': delta=0 # !!!поправка практичної
динамограми, якщо потрібно

        for res in results: # для кожного часу
            if res[0]>=tstart: # починати з часу tstart
                f.append(res[1]/S0-delta) # !!!напруження
                s.append(res[2]) # переміщення
        plt.plot(s, f, style)
    plt.grid(True)
    plt.xlabel(u"S, м") #надпис осі x
    plt.ylabel(u"F, Н") #надпис осі у
    plt.savefig("MaplesimDynCard.png")

def optimize(self):
    """Розраховує модель для різних значень її параметрів"""
    parList=[i/10.0 for i in [1,2,3,4,5,6,7,8,9,10]] # список значень
параметра
    fl=open("ampl-freq.csv", "w")
    self.pu=PUmodel.PU()
    for par in parList: # для кожного значення
        self.pu.pumpingUnit.ns=par # змінити значення параметра
        self.prepareParams()
        # створює файли результатів x_MaplesimResults.csv

```

```

        #self.createMaplesimModel(True,
str(par)+'_MaplesimResults.csv')
        self.createMaplesimModel(True)
        fl.write("%f;%f;%f;%f;%f;%f\n"%self.drawDynamometerCard())
        fl.flush()
    fl.close()

class Maplesim_model2(Maplesim_model): # успадковує Maplesim_model
    """Спрощена модель ШЧУ у Maplesim
    Потребує файлу моделі PUmodel.msim"""

    def prepareParams(self):
        """Підготовлює параметри Maplesim моделі"""
        pu=self.pu # ШЧУ

        M=pu.suckerRodString.mass() # маса колони
        W=pu.suckerRodString.weight() # вага колони
        WF=pu.fluidWeight() # вага рідини
        C=pu.suckerRodString.stiffness() # жорсткість колони
        D=pu.suckerRodString.damping()# еквівалентний коефіцієнт опору
        #D=50*D # увага, збільшено в 50! (або прийняти psi=0.5)
        #D=psi*math.sqrt(C*M)/(2*math.pi) # ?

        # підготувати параметри для моделі Maplesim
        modelParams={'S1_amplitude':pu.pumpingUnit.A, # параметри верстата-
гойдалки
                    'S1_freqHz':pu.pumpingUnit.ns,
                    'S1_phase':pu.pumpingUnit.fi,
                    'SD1_c':C, # параметри гідравлічної частини
                    'SD1_d':D,
                    'M1_m':M,
                    'CF1_f_constant':-W,
                    'G1_k':-WF
                    }

        for p in sorted(modelParams.keys()):
            print p, modelParams[p]

        self.modelParams=modelParams

class Maplesim_model3(Maplesim_model): # успадковує Maplesim_model
    """Модель верстата-гойдалки у Maplesim
    Потребує файлу моделі PUmodel.msim"""

    def prepareParams(self):
        """Підготовлює параметри Maplesim моделі"""
        pu=self.pu # ШЧУ
        # підготувати параметри для моделі Maplesim
        modelParams={'p1_L':pu.pumpingUnit.p1['L'], # параметри ланок
                    'p1_m':pu.pumpingUnit.p1['m'],
                    'p1_Iz':pu.pumpingUnit.p1['Iz'],

```

```

        'p2_L':pu.pumpingUnit.p2['L'],
        'p2_m':pu.pumpingUnit.p2['m'],
        'p2_Iz':pu.pumpingUnit.p2['Iz'],
        'p3_L':pu.pumpingUnit.p3['L'],
        'p3_m':pu.pumpingUnit.p3['m'],
        'p3_Iz':pu.pumpingUnit.p3['Iz'],
        'p4_L':pu.pumpingUnit.p4['L'],
        'p4_m':pu.pumpingUnit.p4['m'],
        'p4_Iz':pu.pumpingUnit.p4['Iz'],
        'p5_L':pu.pumpingUnit.p5['L'], # радіус противаги
        'p5_m':pu.pumpingUnit.p5['m'], # маса противаги
        'p5_Iz':pu.pumpingUnit.p5['Iz'],
        'dx':pu.pumpingUnit.dx, # відстані між осями
        'dy':pu.pumpingUnit.dy,
        'r_k':pu.pumpingUnit.r, # радіус кривошипа
        'const1_k':-pu.pumpingUnit.omega # кутова швидкість
кривошипів, рад/с
        # знак "-" для від'ємного дезаксіалу (Архіпов с.15)
        # тоді хід ввєрх довший
        # ще бажано FF1_teta=pi/2
    }

    for p in sorted(modelParams.keys()):
        print p, modelParams[p]

    self.modelParams=modelParams

if __name__ == '__main__':
    model=Maplesim_model() # !!! визначити вірний клас
    #model.optimize()

    model.pu=PUmodel.PU()
    print "Fmin,Fmax=", model.pu.polishedRodForce()
    #print "Rp=", model.pu.balancing()
    model.prepareParams()

    model.createMaplesimModel(execute=1)
    #model.drawDynamometerCard(S0=math.pi*0.019**2/4) #-
math.pi*0.0085**2/4)
    #model.drawDynamometerCard()
    # іншим способом побудови динамограм є збереження maplesim plots у .csv
    #model.drawDynamometerCardMulti(['1.csv', '2.csv', '3.csv'], [326.0e-
6, 326.0e-6, 326.0e-6], [0.166666, 0.166666, 0.166666], ['k-', 'k--', 'k:'])

    #f=model.modelParams['S1_freqHz']
    model.drawDynamometerCardMulti(['testAPI.csv'], [1.0], [0.10833], ['k-'])

#model.drawDynamometerCardMulti(['2_.csv', '1_.csv', '2.csv', '1.csv'], [1.0, 1.
0, 1.0, 1.0], [0.10833, 0.10833, f, f], ['k--', 'k--', 'k-', 'k-'])

```

Лістинг Г.3 – maplepy.py – Python-інтерфейс до Maplesim

```

# -*- coding: CP1251 -*-
class MapleInterface(object):
    """Інтерфейс до Maple"""
    mapleExePath=r"c:\Program Files\Maple 18\bin.win\smaple.exe"
    codeTemplate="" # шаблон коду Maple
    def getCode(self):
        """Повертає код Maple"""
        return ""
    def runMaple(self):
        """Виконує код Maple"""
        import subprocess,tempfile,os
        tf=tempfile.NamedTemporaryFile(delete=False, suffix='.mpl') #
тимчасовий файл .mpl
        tf.write(self.getCode()) # записати код
        tf.close()
        subprocess.Popen(self.mapleExePath+' '+tf.name).communicate() #
виконати код Maple
        os.unlink(tf.name) # видалити тимчасовий файл

class MapleInterface4Maplesim(MapleInterface):
    """Інтерфейс до Maplesim"""
    path="c:/9" # шлях до файлів моделі та результатів (розділювач тільки
/)
    filenameMaplesim="testAPI.msim" # файл моделі Maplesim
    # приклад доступу до параметру SD1_c підсистеми n: n_SD1_c
    paramDictMaplesim={"SD1_c":1, "SD1_d":0.1} # словник параметрів моделі
Maplesim
    execute=False # чи виконувати відразу симуляцію
    resultCSVfile="testAPI.csv" # файл результатів CSV
    # шаблони коду Maple
    codeTemplate1=r""""A :=MapleSim:-
LinkModel('filename'="{filenameMaplesim}"):
A:-SetParameters([{setParams}]):
""""
    codeTemplate2=r""""simData:=A:-Simulate(output = datapoint):
ExportMatrix("{resultCSVfile}", simData, target=delimited, delimiter=";"):
""""
    def paramString(self,paramDictMaplesim):
        """Повертає рядок параметрів MapleSim"""
        paramList=[k+" = "+str(v) for k,v in paramDictMaplesim.iteritems()]
# перетворити в список
        return ", ".join(paramList) # об'єднати в рядок

    def getCode(self):
        """Повертає код Maple"""
        setParams=self.paramString(self.paramDictMaplesim)
        filenameMaplesim=self.path+'/'+self.filenameMaplesim
        if self.execute: # якщо відразу виконувати симуляцію
            allCodeTemplate=self.codeTemplate1+self.codeTemplate2

```

```

        resultCSVfile=self.path+'/'+self.resultCSVfile
        code=allCodeTemplate.format(filenameMaplesim=filenameMaplesim,
setParams=setParams, resultCSVfile=resultCSVfile)
    else:

code=self.codeTemplate1.format(filenameMaplesim=filenameMaplesim,
setParams=setParams)
    return code

def readCSVfile(self,nc=2):
    """Читає файл результатів CSV та повертає список з результатами
у вигляді [[t1,x1,y1,...],[t2,x2,y2,...],...]
nc - кількість змінних (мінімум 2)"""
    import csv
    resultCSVfile=self.path+'/'+self.resultCSVfile
    csv_file=open(resultCSVfile, "rb")
    reader=csv.reader(csv_file,delimiter = ';')
    resultList=[]
    for row in reader: # для кожного рядка
        oneResult=[] # список результатів з одного рядка
        for i in range(nc): # для кожної змінної
            oneResult.append(float(row[i])) # додати значення змінної
        resultList.append(oneResult) # додати список результатів з
одного рядка
    csv_file.close()
    return resultList

# class MapleInterface4Maplesim2(MapleInterface4Maplesim):
#     """Інтерфейс до Maplesim. Для моделей з підсистемами"""
#     paramDictMaplesim={'SD1_c':2} # параметри моделі
#     paramDictMaplesim2={'n':{'SD1_d':0.2}} # параметри підсистем моделі
#     CodeTemplate3="""A:-SetSubsystemName("{subSystemName}"):
# A:-SetParameters([{setParams}]):
# """
#
#     def getSubCode(self):
#         """Повертає код Maple для вводу параметрів підсистем"""
#         codeList=[]
#         for sname in self.paramDictMaplesim2:
#             setParams=self.paramString(self.paramDictMaplesim2[sname])
#
subCode=self.CodeTemplate3.format(subSystemName=sname,setParams=setParams)
#             codeList.append(subCode)
#         return '\n'.join(codeList)
#
#     def getCode(self):
#         """Повертає код Maple"""
#         setParams=self.paramString(self.paramDictMaplesim)
#         filenameMaplesim=self.path+'/'+self.filenameMaplesim
#         if self.execute: # якщо відразу виконувати симуляцію

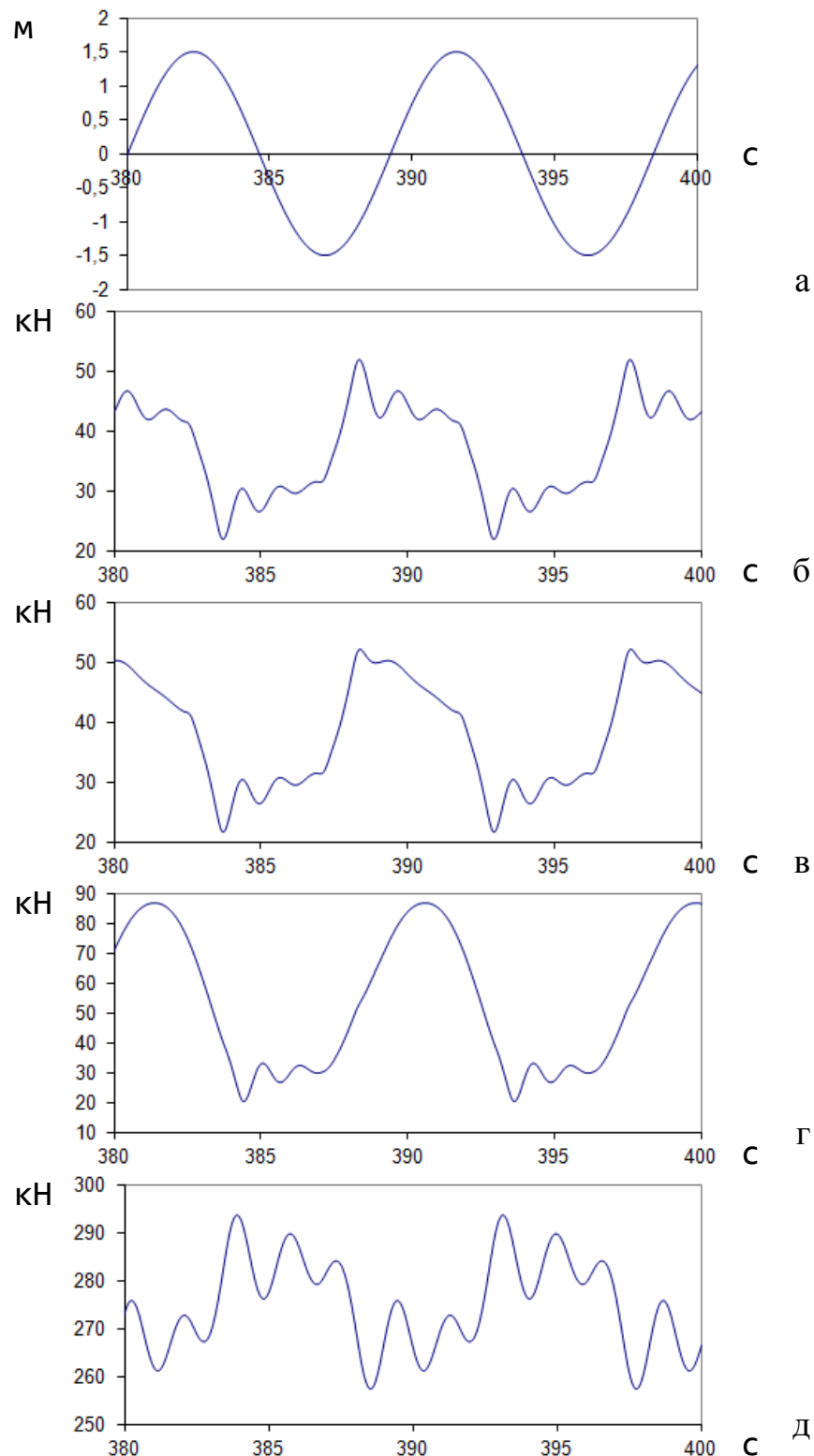
```

```
#
allCodeTemplate=self.codeTemplate1+self.getSubCode()+self.codeTemplate2
#         resultCSVfile=self.path+'/'+self.resultCSVfile
#
code=allCodeTemplate.format(filenameMaplesim=filenameMaplesim,
setParams=setParams, resultCSVfile=resultCSVfile)
#         else:
#         allCodeTemplate=self.codeTemplate1+self.getSubCode()
#
code=allCodeTemplate.format(filenameMaplesim=filenameMaplesim,
setParams=setParams)
#         return code

if __name__ == '__main__':
    ms=MapleInterface4Maplesim()
    print ms.getCode()
    #ms.runMaple()
    #print ms.readCSVfile(3)
```

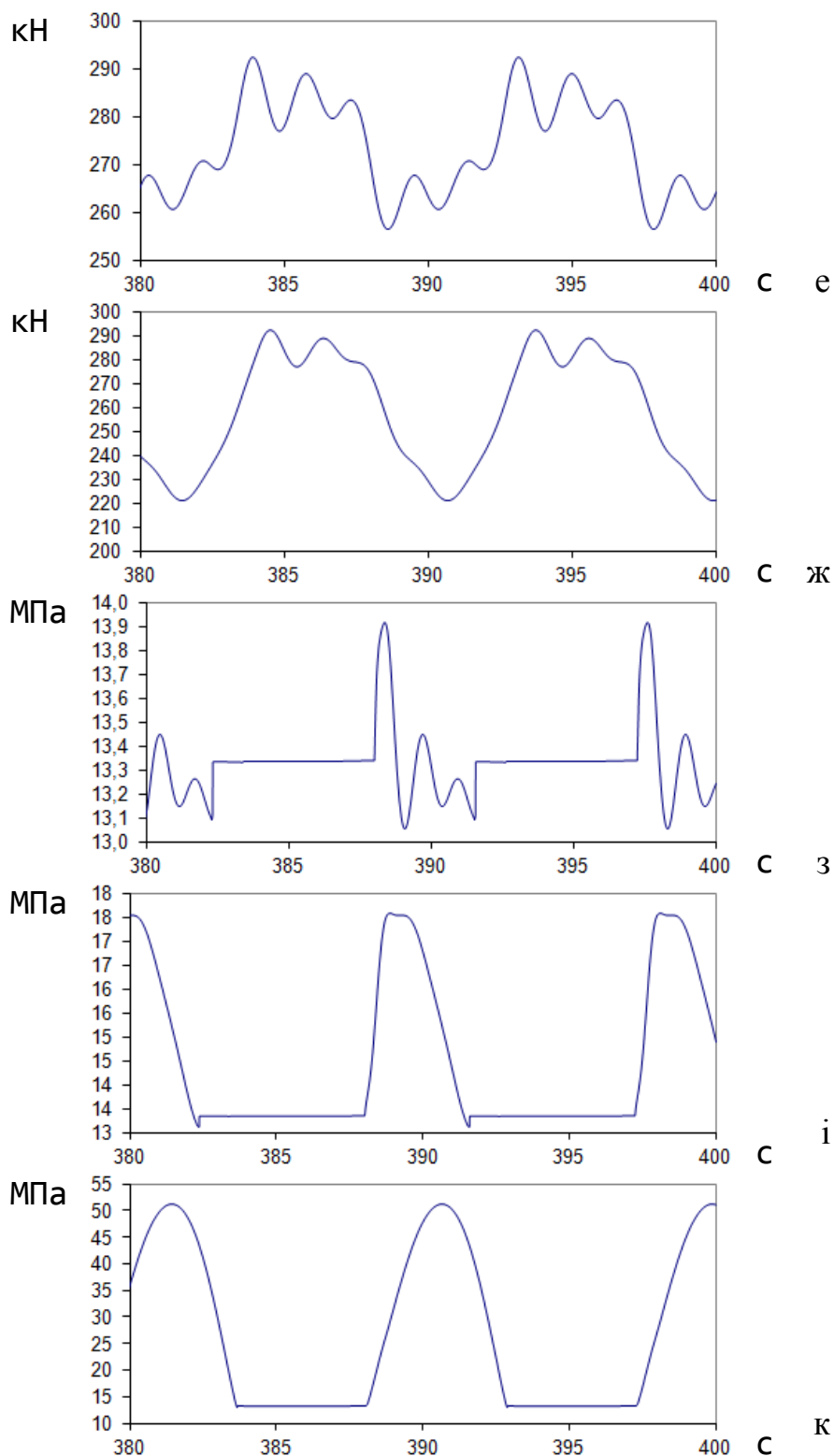

ДОДАТОК Д

Графіки циклічних навантажень на колони ШН і НКТ



а – переміщення полірованого штока; б, в, г – сила у верхній частині колони ШН;
 д – сила у верхній частині колони НКТ; г – $d=10$ мм; в – $d=20$ мм

Рисунок Д.1, аркуш 1 – Залежності параметрів від часу симуляції в умовах відсутності (б, д, з) та наявності СПУ на довжині 500 м, що зменшили діаметр отвору НКТ до значення d



е, ж – сила у верхній частині колони НКТ; з, і, к – тиск у нижній частині колони НКТ; ж, к – $d=10$ мм; е, і – $d=20$ мм

Рисунок Д.1, аркуш 2

ДОДАТОК Е

Пакет для моделювання кулькового клапана методами обчислювальної гідродинаміки

Код також доступний в GitHub (vkorey/BallValveAbaqus: Abaqus scripts for modelling of ball valve. URL: <http://github.com/vkorey/BallValveAbaqus>). Вміст пакету:

- pickleIPC.py – бібліотека для взаємодії між процесами;
- server.py – сервер для паралельних задач;
- script.py – сценарій AbaqusCAE.

Лістинг Е.1 – pickleIPC.py

```
# -*- coding: CP1251 -*-
"""Бібліотека для IPC. Містить функції, які дозволяють процесам
обмінюватись об'єктами python через сокети та тимчасові файли"""
import os, pickle, tempfile

def writeSocket(_socket, data):
    """Відсилає об'єкт python (data) через сокет"""
    f = _socket.makefile('wb') #,buffer_size # створити файл
    pickle.dump(data, f, pickle.HIGHEST_PROTOCOL) # законсервувати дані
    f.close() # закрити файл

def readSocket(_socket):
    """Повертає об'єкт python отриманий з сокета"""
    f = _socket.makefile('rb') #,buffer_size # створити файл
    data = pickle.load(f) # розконсервувати дані з файлу
    f.close() # закрити файл
    return data

def writeTempFile(data, tmpFileName="data4AbaqusScript.tmp"):
    """Записує об'єкт (data) у тимчасовому файлі в тимчасовій папці"""
    name=os.path.join(tempfile.gettempdir(),tmpFileName)
    f=open(name, "wb") # відкрити бінарний файл для запису
    pickle.dump(data,f) # законсервувати дані у файлі
    f.close() # закрити файл

def readTempFile(tmpFileName="data4AbaqusScript.tmp"):
    """Повертає об'єкт шляхом читання тимчасового файлу у тимчасовій папці"""
    name=os.path.join(tempfile.gettempdir(),tmpFileName)
    f=open(name, "rb") # відкрити бінарний файл для читання
    data=pickle.load(f) # розконсервувати дані з файлу
    f.close() # закрити файл
    return data
```

Лістинг Е.2 – server.py

```

# -*- coding: CP1251 -*-
import csv,pickleIPC,subprocess,datetime

csv_file=open("results.csv", "wb") # CSV файл результатів
writer = csv.writer(csv_file,delimiter = ';') # об'єкт для запису у файл
for h in [0.04, 0.05]: # для значень h у списку
    pickleIPC.writeTempFile(h) # записати дані в тимчасовий файл
    print datetime.datetime.now().isoformat() # час початку
    print "Abaqus CAE started. Please wait"
    # виконує скрипт в Abaqus та чекає завершення
    AbaqusPath=r"d:\SIMULIA\Abaqus\6.12-3\code\bin\abq6123.exe"
    subprocess.Popen(AbaqusPath+' cae noGUI=script.py').communicate()
    print datetime.datetime.now().isoformat() # час завершення
    print "Abaqus CAE finished"
    data=pickleIPC.readTempFile() # прочитати дані, які повернув скрипт
    writer.writerow(data) # записати у файл CSV
    csv_file.flush() # очистити буфер
csv_file.close() # закрити файл CSV

```

Лістинг Е.3 – script.py

```

# -*- coding: CP1251 -*-
from part import *
from material import *
from section import *
from optimization import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

def set_values(part,feature,par):
    '''Присвоює значення параметрам. Приклад:
    par={'h1':0.0002,'h2':0.00004}
    set_values(part='A1',feature='Shell planar-1',par=par)'''
    p=model.parts[part] # деталь
    f=p.features[feature] # елемент
    s=model.ConstrainedSketch(name='__edit__', objectToCopy=f.sketch) #
    тимчасовий ескіз
    p.projectReferencesOntoSketch(filter=COPLANAR_EDGES, sketch=s,
    upToFeature=f) # спроекувати
    for k,v in par.iteritems(): # для всіх параметрів
        s.parameters[k].setValues(expression=str(v)) # установити значення

```

```

f.setValues(sketch=s) # установити ескіз
del s # знищити
p.regenerate() # регенерувати деталь

def readODB_set2(set,step,var,pos=NODAL):
    '''Читає результати з останнього фрейму кроку на заданій множині
    set - множина, step - крок, var - змінна:
    ('S','Pressure'), ('U','Magnitude'), ...
    pos - позиція: NODAL - для вузлів,INTEGRATION_POINT - для елементів
    Приклад: readODB_set2(set='Cont',step='Step-1',var=('S','Mises'))
    '''
    if pos==NODAL:
        s=odb.rootAssembly.nodeSets[set.upper()] # множина вузлів
    if pos==INTEGRATION_POINT:
        s=odb.rootAssembly.elementSets[set.upper()] # множина елементів
    fo=odb.steps[step].frames[-
1].fieldOutputs[var[0]].getSubset(region=s,position=pos) # дані
    #openOdb(r'C:/Temp/Model-1.odb').steps['Step-
1'].frames[4].fieldOutputs['CPRESS'].getSubset(position=NODAL,
region=openOdb(r'C:/Temp/Model-
1.odb').rootAssembly.nodeSets['CONT']).values[0].data
    res=[] # список результатів
    for v in fo.values: # для кожного вузла/елемента
        # додати до списку результатів
        if var[1]=='Pressure': res.append(v.press)
        if var[0]=='U' and var[1]=='Magnitude': res.append(v.magnitude)
        if var[1]=='U1': res.append(v.data.tolist()[0])
        if var[1]=='U2': res.append(v.data.tolist()[1])
        if var[0]=='PRESSURE': res.append(v.data)
    return res # повертає список значень

import pickleIPC,shutil,os
os.chdir('C:/Abaqus') # визначити робочий каталог
openMdb(pathName='C:/Abaqus/testCFD.cae') # відкрити модель
model=mdb.models['Model-1'] # створити об'єкт моделі
h=pickleIPC.readTempFile() # прочитати дані, які передав сервер
# надати значення параметру 'h' та перебудувати геометрію
#!'Solid revolve-1'
set_values(part='Part-1',feature='Solid extrude-1',par={'h':h})
model.parts['Part-1'].generateMesh() # генерувати сітку елементів
mdb.jobs['Model-1'].submit() # виконати задачу
mdb.jobs['Model-1'].waitForCompletion() # чекати завершення
# зберегти файл результатів .odb під унікальною назвою
shutil.copyfile(r'C:/Abaqus/Model-1.odb',
r'C:/Abaqus/results_'+str(h)+'.odb')
odb=openOdb(r'C:/Abaqus/Model-1.odb') # відкрити базу даних результатів
results=readODB_set2(set='bot',step='Step-1',var=('PRESSURE','')) #
отримати результати
pickleIPC.writeTempFile([h, sum(results)/len(results)]) # записати дані для
передачі їх серверу
odb.close() # закрити базу даних результатів

```

ДОДАТОК Є

ruscodyn – пакет для компонентно-орієнтованого акаузального моделювання мовою Python

Код також доступний в GitHub (vkorey/ruscodyn: Component-oriented acausal modeling of the dynamical systems in Python language. URL: <http://github.com/vkorey/ruscodyn>). Вміст пакету:

- Ruscodyn.mo – моделі штангової колони (мова Modelica, для порівняння);
- ruscodyn.py – компоненти та розв’язувач (метод Ейлера);
- ruscodynDAE.py – компоненти та розв’язувач (DAE);
- main1.py – модель вільних коливань колони (метод Ейлера);
- trapComponents.py – компоненти (метод трапецій);
- main1T.py – модель вільних коливань колони (метод трапецій);
- main1DAE.py – модель вільних коливань колони (DAE);
- main1Sym.py – модель вільних коливань колони (аналітична);
- main2s.py – односекційна модель процесу відкачування (метод Ейлера);
- main2.py – двосекційна модель процесу відкачування (метод Ейлера);
- main2V.py – двосекційна модель обриву колони (метод Ейлера, події);
- main2sDAE.py – односекційна модель процесу відкачування (DAE);
- main2DAE.py – двосекційна модель процесу відкачування (DAE);
- eliminate.py – скорочення системи рівнянь для SymPy.

Для встановлення просто помістіть усі модулі в одну папку. Вимоги:

- Python 3.7 (рекомендується) або Python 2.7;
- numpy-1.16.4+mkl;
- scipy-1.2.2 (необов'язково);
- mpmath-1.1.0;
- sympy-1.4;
- Assimulo-2.9 (для ruscodynDAE);
- matplotlib-2.2.4 (необов'язково, для графіків);

– OpenModelica 1.12 (для Pycodyn.mo).

Приклад використання (Python):

```
d:\WinPython32_37\python-3.7.4\python.exe main1.py
```

Приклад використання (OpenModelica):

```
loadModel(Modelica)
loadFile("Pycodyn.mo")
simulate(Pycodyn.Oscillator, stopTime=10)
plot(mass1.s)
simulate(Pycodyn.Pumping, stopTime=100)
plotParametric(motion1.flange.s, spring1.flange_a.f)
```

Лістинг Є.1 – Pycodyn.mo

```
1 within;
2 package Pycodyn
3 extends Modelica.Icons.Package;
4
5 connector Flange // клас-connector
6   Real s; // змінна (переміщення в точці з'єднання рівні)
7   flow Real f; // змінна (сума сил в точці з'єднання рівна нулю)
8 end Flange;
9
10 model Fixed // клас-model
11   parameter Real s0=0; // параметр зі значенням за замовчуванням
   (постійний у часі)
12   Flange flange; // об'єкт класу Flange
13 equation // рівняння моделі
14   flange.s = s0;
15 end Fixed;
16
17 partial model Transl // клас-model
18   Flange flange_a; // об'єкт класу Flange
19   Flange flange_b; // об'єкт класу Flange
20 end Transl;
21
22 model Mass // клас-model
23   extends Transl; // успадкування класу Transl
24   parameter Real m(min=0, start=1); // параметр
25   Real s; // змінна
26   Real v(start=0); // змінна з початковою умовою
27   Real a(start=0); // змінна з початковою умовою
28 equation // рівняння моделі
29   v = der(s);
30   a = der(v);
31   m*a = flange_a.f + flange_b.f;
32   flange_a.s = s;
33   flange_b.s = s;
```

```

34 end Mass;
35
36 model SpringDamper // клас-model
37   extends Transl; // успадкування класу Transl
38   parameter Real c(final min=0, start=1); // параметр
39   parameter Real d(final min=0, start=1); // параметр
40   Real s_rel(start=0); // змінна
41   Real v_rel(start=0); // змінна
42   Real f; // змінна
43 equation // рівняння моделі
44   f = c*s_rel+d*v_rel;
45   s_rel = flange_b.s - flange_a.s;
46   v_rel = der(s_rel);
47   flange_b.f = f;
48   flange_a.f = -f;
49 end SpringDamper;
50
51 model Motion // клас-model
52   parameter Real A=2.1/2;
53   parameter Real n=6.4/60;
54   Flange flange; // об'єкт класу Flange
55 equation // рівняння моделі
56   flange.s = A*sin(6.283185307179586*n*time);
57 end Motion;
58
59 model Force // клас-model
60   Real f(start=0);
61   Flange flange; // об'єкт класу Flange
62 equation // рівняння моделі
63   flange.f = f;
64 end Force;
65
66 model Oscillator // клас-model
67   Mass mass1(s(start=-1), v(start=0), m=3961.0); // об'єкт з початковими
    умовами
68   SpringDamper spring1(c=44650.0, d=2120.7); // об'єкт
69   Fixed fixed1(s0=0); // об'єкт
70 equation // додаткові рівняння
71   // створює систему рівнянь (див. коментарі класу Flange)
72   connect(fixed1.flange, spring1.flange_a);
73   connect(spring1.flange_b, mass1.flange_a);
74 end Oscillator;
75
76 model Pumping // клас-model
77   Mass mass1(s(start=-0.9), v, m=3961.0); // об'єкт з початковими
    умовами
78   SpringDamper spring1(c=44650.0, d=2120.7); // об'єкт
79   Motion motion1(A=2.1/2, n=6.4/60);
80   Force force1;
81 equation // додаткові рівняння
82   connect(motion1.flange, spring1.flange_a);

```



```

83 connect(spring1.flange_b, mass1.flange_a);
84 connect(mass1.flange_b, force1.flange);
85 algorithm
86 if mass1.v <= 0 then
87     force1.f:=34687.0;
88 else
89     force1.f:=34687.0+18499.0*tanh(abs(mass1.v )/0.01);
90 end if;
91 end Pumping;
92
93 end Pycodyn;

```

Лістинг Є.2 – pycodyn.py

```

# -*- coding: utf-8 -*-
from sympy import *
import math, time
import matplotlib.pyplot as plt

def (d,name): # повертає значення за назвою символу
    for k in d:
        if repr(k)==name: return d[k]

dt=0.1 # крок часу
#dt=Symbol('dt') # тільки для отримання рівнянь в символній формі

class Translational1D(object):
    """Базовий клас механічних поступальних 1D компонентів"""
    def __init__(self, name, args):
        self.name=name # назва компонента
        for k,v in args.items(): # для кожної пари ключ-значення
            if k in ['name','self']: continue # окрім name і self
            if v==None: # якщо значення є None
                # створити символну змінну з назвою name+'_'+k
                self.__dict__[k]=Symbol(name+'_'+k)
            elif type(v) in [float,Float]: # якщо значення є дійсним числом
                self.__dict__[k]=Number(v) # створити константу
        self.eqs=[] # список рівнянь
        self.pins=[] # список фланців

    def pinEqs(self,pindex,pins):
        eqs=[] # список рівнянь фланця
        f=Number(0) # сума сил на фланцях інших компонентів
        for pin in pins: # для кожного фланця інших компонентів
            # додати рівняння, що описують рівність на фланці:
            eqs.append(Eq(self.pins[pindex]['x'], pin['x'])) # переміщень
            eqs.append(Eq(self.pins[pindex]['xp'], pin['xp'])) # переміщень
        в час t-dt
        f+=pin['f'] # додати до суми сил
        eqs.append(Eq(self.pins[pindex]['f'], -f)) # рівність нулю суми сил
        на фланці

```

```
return eqs
```

```
class Mass(Translational1D):
```

```
    """Маса, зосереджена у точці, що рухається поступально"""
```

```
    def __init__(self, name, m=1.0, x=None, xp=None, v=None, vp=None,
a=None, f1=None, f2=None):
```

```
        Translational1D.__init__(self, name, locals()) # виклик
конструктора базового класу
```

```
        # система рівнянь
```

```
        self.eqs=[Eq(self.m*self.a, self.f1+self.f2),
                Eq(self.a, (self.v-self.vp)/dt),
                Eq(self.v, (self.x-self.xp)/dt)]
```

```
        self.pins=[dict(x=self.x, xp=self.xp, f=self.f1),
                dict(x=self.x, xp=self.xp, f=self.f2)] # два фланця
```

```
class SpringDamper(Translational1D):
```

```
    """Поступальні 1D пружина і демпфер, з'єднані паралельно"""
```

```
    def __init__(self, name, c=1.0, d=0.1, x1=None, x2=None, x1p=None,
x2p=None, vrel=None, f1=None, f2=None):
```

```
        Translational1D.__init__(self, name, locals())
```

```
        # система рівнянь
```

```
        self.eqs=[Eq(self.c*(self.x2-self.x1)+self.d*self.vrel, self.f2),
                Eq(-self.f2, self.f1),
                Eq(self.vrel, (self.x2-self.x2p)/dt-(self.x1-
```

```
self.x1p)/dt)]
```

```
        self.pins=[dict(x=self.x1, xp=self.x1p, f=self.f1),
                dict(x=self.x2, xp=self.x2p, f=self.f2)] # два фланця
```

```
class Force(Translational1D):
```

```
    """1D сила, точка прикладення якої рухається поступально"""
```

```
    def __init__(self, name, f=None, x=None, xp=None):
```

```
        Translational1D.__init__(self, name, locals())
```

```
        self.pins=[dict(x=self.x, xp=self.xp, f=-self.f)] # один фланець
```

```
class System(object):
```

```
    """Система компонентів, з'єднаних фланцями"""
```

```
    def __init__(self, els, eqs):
```

```
        self.els=els # список компонентів
```

```
        self.elsd=dict([(e.name,e) for e in els]) # те саме, але словник
```

```
        self.eqs=[] # список рівнянь системи
```

```
        for e in self.els: # для кожного компонента
```

```
            self.eqs+=e.eqs # з'єднати з рівняннями компонента
```

```
        self.eqs=self.eqs+eqs # з'єднати з додатковими рівняннями
```

```
    def solveN(self, eqs): # розв'язує систему алг. рівнянь за доп. scipy
```

```
        import scipy.optimize
```

```
        eqs0=[]
```

```
        vrs=set()
```

```
        for e in eqs:
```

```
            eq=e.lhs-e.rhs
```

```

    eqs0.append(eq)
    for a in eq.atoms():
        if a.is_Symbol:
            vrs.add(a)
vrs=list(vrs)
f=lambdify([vrs], eqs0, 'numpy')
goals=[0.0 for i in vrs]
sol=scipy.optimize.root(f, goals, method='lm')
d=dict(zip(vrs,sol.x))
return d

def solve(self, ics): # розв'язує систему алг. рівнянь в момент часу t
    eqs=[e.subs(ics) for e in self.eqs] # підстановка початкових умов
    eqs=[e for e in eqs if e not in (True,False)] # відкинути усі
вироджені рівняння
    # розв'язати систему за допомогою:
    #sol=nsolve(eqs) # SymPy (повільно) #або solve
    sol=self.solveN(eqs) # SciPy (швидше)
    sol.update(ics) # оновити словник словником поч. умов ics
    return sol

def solv(self, preState): # розв'язує шляхом підстановки у вираз symPy
    state=preState.copy()
    for k in self.ceqs: # поточні рівняння
        state[k]=self.ceqs[k].subs(preState).evalf()
        #assert type(state[k]) in [float,Float]
    return state

def solvN(self, preState): # те саме, але за доп. lambda-функції
# використовуйте Python 3.7 для швидшого виконання
    state=preState.copy()
    ls=dict([(repr(a),state[a]) for a in self.vrsp]) # словник аргум.
    res=self.ceqsf(**ls) # виклик lambda-функції
    for a,v in zip([i[0] for i in self.ceqsi], res):
        state[a]=v # оновити state
    return state

def createCurEqs(self, fnBC): # створити поточні 'швидкі рівняння'
    eqs=Tuple(*self.eqs)
    vrs={i for i in eqs.atoms(Symbol) if repr(i)[-1]!='p'} # змінні без
'p'
    vrsbc=set(fnBC.vrs)
    vrs=vrs-vrsbc # невідомі змінні на поточному кроці
    self.ceqs=solve(eqs,vrs) # поточні вирази
    self.vrsp={i for i in eqs.atoms(Symbol) if repr(i)[-1]=='p'} #
змінні з 'p'
    self.vrsp.update(vrsbc) # відомі змінні на поточному кроці
    self.ceqsi=self.ceqs.items() # впорядковані вирази
    self.ceqsf=lambdify(self.vrsp,[i[1] for i in self.ceqsi],'numpy') #
поточна lambda-функція

```

```

def solveDyn(self, state, timeEnd, fnBC):
    # розв'язує динамічну задачу
    t=0.0 # змінна часу
    T=[] # список значень часу
    Res=[] # список результатів

    self.createCurEqs(fnBC)
    ics={} # для self.solve()
    start = time.time()
    while t<timeEnd:
        for k in state: # попередні значення "xp=x"...
            if repr(k)[-1]=='p':
                state[k]=byName(state,repr(k)[: -1])
                ics[k]=byName(state,repr(k)[: -1]) # для self.solve()

        state.update(fnBC(state, t)) # оновити граничні умови
        #state=self.solv(state) # шляхом виразу sympy (повільно)
        state=self.solvN(state) # шляхом виразу numpy (швидше)

        # ics.update(fnBC(state, t)) # оновити граничні умови
        # state=self.solve(ics) # шляхом scipy.optimize.root (повільно,
для частих подій)

        print(t)
        T.append(t)
        Res.append(state) # зберегти результати
        t+=dt # збільшити значення часу

        self.event(state) # обробник подій
    end = time.time()
    print('simulation time',end-start)
    return T,Res

def event(self, state): # обробник подій
    pass

```

Лістинг Є.3 – rucodynDAE.py

```

# -*- coding: utf-8 -*-
import numpy as np
from sympy import *
t=Symbol('t')

class Translational1D(object):
    """ Базовий клас механічних поступальних 1D компонентів """
    def __init__(self, name, args):
        self.name=name # назва компонента
        for k,v in args.items(): # для кожної пари ключ-значення
            if k in ['name','self']: continue # окрім name і self
            if v==None: # якщо значення є None
                # створити символічну змінну з назвою name+'_'+k

```

```

        self.__dict__[k]=Symbol(name+'_'+k)
        elif type(v) in [float,Float]: # якщо значення є дійсним числом
            self.__dict__[k]=Number(v) # створити константу
self.eqs=[] # список рівнянь
self.pins=[] # список фланців

def pinEqs(self,pindex,pins):
    eqs=[] # список рівнянь фланця
    f=Number(0) # сума сил на фланцях інших компонентів
    for pin in pins: # для кожного фланця інших компонентів
        # додати рівняння, що описують рівність на фланці:
        eqs.append(Eq(self.pins[pindex]['x'], pin['x'])) # переміщень
        f+=pin['f'] # додати до суми сил
    eqs.append(Eq(self.pins[pindex]['f'], -f)) # рівність нулю суми сил
на фланці
    return eqs

class Mass(Translational1D):
    """Маса, зосереджена у точці, що рухається поступально"""
    def __init__(self, name, m=1.0, x=None, v=None, a=None, f1=None,
f2=None, Dx=None, Dv=None):
        Translational1D.__init__(self, name, locals()) # виклик
конструктора базового класу
        # система рівнянь
        self.eqs=[Eq(self.m*self.a, self.f1+self.f2),
            Eq(self.a, self.Dv),
            Eq(self.v, self.Dx)]
        self.pins=[dict(x=self.x, f=self.f1),
            dict(x=self.x, f=self.f2)] # два фланця

class SpringDamper(Translational1D):
    """Поступальні 1D пружина і демпфер, з'єднані паралельно"""
    def __init__(self, name, c=1.0, d=0.1, x1=None, x2=None, vrel=None,
f1=None, f2=None, Dx1=None, Dx2=None):
        Translational1D.__init__(self, name, locals())
        # система рівнянь
        self.eqs=[Eq(self.c*(self.x2-self.x1)+self.d*self.vrel, self.f2),
            Eq(-self.f2, self.f1),
            Eq(self.vrel, self.Dx2-self.Dx1)]

        self.pins=[dict(x=self.x1, f=self.f1),
            dict(x=self.x2, f=self.f2)] # два фланця

class Force(Translational1D):
    """1D сила, точка прикладення якої рухається поступально """
    def __init__(self,name,f=None, x=None):
        Translational1D.__init__(self, name, locals())
        self.pins=[dict(x=self.x, f=-self.f)] # один фланець

class System(object):
    """Система компонентів, з'єднаних фланцями"""

```

```

def __init__(self, els, eqs):
    self.els=els # список компонентів
    self.elsd=dict([(e.name,e) for e in els]) # те саме, але словник
    self.eqs=[] # список рівнянь системи
    for e in self.els: # для кожного компонента
        self.eqs+=e.eqs # з'єднати з рівняннями компонента
    self.eqs=self.eqs+eqs # з'єднати з додатковими рівняннями
    self.eqs=Tuple(*self.eqs)

def residualArgs(self,eq):
    "Повертає впорядковані аргументи для нев'язок (функцій і похідних
eq)"
    ss=eq.atoms(Symbol) # множина символів рівняння
    ss.discard(t) # без t
    dss=dict([(i.name,i) for i in ss]) # словник назва:символ
    y=set();yd=set() # функція; похідна
    for a in ss:
        if 'D' in a.name: yd.add(a)
        else: y.add(a)
    y_=[];yd_=[] # пари функція; похідна
    for a in yd:
        b_name=a.name.replace('D','') # знайти пару
        if dss.get(b_name): # якщо пара
            yd_.append(a)
            y_.append(dss[b_name])
    yyd=set(y_+yd_)
    y=y_+list(y-yyd)
    yd=yd_+list(yd-yyd)
    return y,yd

def residual(self,t,y,yd): # нев'язки для Assimulo
    yyd=np.concatenate([[t],y,yd][:self.nv])
    r=self.lambdfun(*yyd)
    return np.array(r)

def solveDAE(self, eq, state, stopTime=10.0):
    """Розв'язує динамічну задачу за доп. Assimulo (ODASSL, IDA)
state - словник з початковим станом"""
    y,yd=self.residualArgs(eq)
    self.y=y
    self.yd=yd
    self.nv=len(y+yd)+1 # кількість аргументів для lambdfun (з t)
    eq0=[e.rhs-e.lhs for e in eq]
    self.lambdfun=lambdify([t]+y+yd,eq0,'numpy')
    #import inspect
    #print(inspect.getsource(self.lambdfun))

    y0=[state[i] for i in y] # початкові умови
    yd0=[state[i] for i in yd]
    # забезпечує однакову довжину y0, yd0 (важливо для ODASSL):
    dn=len(y0)-len(yd0)

```

```

    if dn>0: yd0+=[0.0]*dn
    else: y0+=[0.0]*abs(dn)

    from assimulo.problem import Overdetermined_Problem,
Implicit_Problem
    from assimulo.solvers import ODASSL,IDA

    # model = Overdetermined_Problem(self.residual, y0=y0, yd0=yd0)
    # sim = ODASSL(model)

    model = Implicit_Problem(self.residual, y0=y0, yd0=yd0)
    model.algvar = [1]*len(yd)+[0]*dn #[1,1,1,0,0,0,0,0]
    sim = IDA(model)
    sim.suppress_alg = True
    print(sim.get_options())

    T, Y, Yd = sim.simulate(stopTime)
    #sim.plot()
    return T, Y, Yd

def solve(self,eq,ics): # для статичних задач
    eq=eq.subs(ics)
    return solve(eq) # розв'язувач sympy

def prnt(eq): # друк рівнянь
    print('\nEquations=')
    for i in eq: print(i)

```

Лістинг Є.4 – main1.py

```

1  # -*- coding: utf-8 -*-
2  from pycodyn import *
3
4  # створити компоненти:
5  s1=SpringDamper(name='s1', c=44650.0, d=2120.0)
6  m1=Mass(name='m1',m=3961.0)
7  reqs=s1.pinEqs(1,[m1.pins[0]]) # список додаткових рівнянь
8  s=System(els=[s1,m1], eqs=reqs) # система
9
10 # розв'язує статичну задачу - колона розтягнута на 1 м
11 ics={m1.x:-1.0, m1.v:0.0, m1.a:0.0, s1.x1:0.0, s1.x1p:0.0,
    m1.vp:0.0}
12 d=s.solve(ics)
13
14 def fnBC(d, t):
15     """граничні умови в час t для компонентів fnBC.vrs"""
16     val = 0.0, 0.0
17     return dict(zip(fnBC.vrs, val))
18 fnBC.vrs = s.elsd['s1'].x1, s.elsd['m1'].f2
19
20 # розв'язує динамічну задачу - вільні коливання колони

```

```

21 T,R=s.solveDyn(d, timeEnd=10, fnBC=fnBC)
22 plt.plot(T, [d[m1.x] for d in R])
23 plt.xlabel('t, s'); plt.ylabel('m1.x, m')
24 plt.show()

```

Лістинг Є.5 – trapComponents.py

```

# -*- coding: utf-8 -*-
from pycodyn import *

class Mass(Translational1D):
    def __init__(self,name,m=1.0,x=None,xp=None,v=None,vp=None,
a=None,ap=None,f1=None,f2=None):
        Translational1D.__init__(self, name, locals()) # виклик
конструктора базового класу

        self.eqs=[Eq(self.m*self.a, self.f1+self.f2),
                Eq((self.a+self.ap)/2, (self.v-self.vp)/dt),
                Eq((self.v+self.vp)/2, (self.x-self.xp)/dt)] # система
рівнянь
        self.pins=[dict(x=self.x, xp=self.xp, f=self.f1),
                dict(x=self.x, xp=self.xp, f=self.f2)] # два фланця

class SpringDamper(Translational1D):
    def
__init__(self,name,c=1.0,d=0.1,x1=None,x2=None,x1p=None,x2p=None,vrel=None,
f1=None,f2=None,v1=None,v2=None,v1p=None,v2p=None):
        Translational1D.__init__(self, name, locals())

        self.eqs=[Eq(self.c*(self.x2-self.x1)+self.d*self.vrel, self.f2),
                Eq(-self.f2, self.f1),
                Eq((self.v1+self.v1p)/2, (self.x1-self.x1p)/dt),
                Eq((self.v2+self.v2p)/2, (self.x2-self.x2p)/dt),
                Eq(self.vrel, self.v2-self.v1)] # система рівнянь

        self.pins=[dict(x=self.x1, xp=self.x1p, f=self.f1),
                dict(x=self.x2, xp=self.x2p, f=self.f2)] # два фланця

```

Лістинг Є.6 – main1T.py

```

# -*- coding: utf-8 -*-
from pycodyn import *
from trapComponents import SpringDamper,Mass

# створити компоненти:
s1=SpringDamper(name='s1', c=44650.0, d=2120.0)
m1=Mass(name='m1', m=3961.0)
peqs=s1.pinEqs(1,[m1.pins[0]]) # список додаткових рівнянь
s=System(els=[s1,m1], eqs=peqs) # система
# розв'язує статичну задачу - колона розтягнута на 1 м

```



```

ics={m1.x:-1.0, m1.v:0.0, m1.a:0.0, s1.x1:0.0, s1.x1p:0.0, m1.vp:0.0,
m1.ap:0.0, s1.v1:0.0, s1.v1p:0.0, s1.v2:0.0, s1.v2p:0.0}
d=s.solve(ics)

def fnBC(d, t):
    """граничні умови в час t для компонентів fnBC.vrs"""
    val = 0.0, 0.0
    return dict(zip(fnBC.vrs, val))
fnBC.vrs = s.elsd['s1'].x1, s.elsd['m1'].f2

# розв'язує динамічну задачу - вільні коливання колони
T,R=s.solveDyn(d, timeEnd=10, fnBC=fnBC)
plt.plot(T, [d[m1.x] for d in R])
plt.xlabel('t, s'); plt.ylabel('m1.x, m')
plt.show()

```

Лістинг Є.7 – main1DAE.py

```

1 # -*- coding: utf-8 -*-
2 from pycodynDAE import *
3
4 # створити компоненти:
5 s1=SpringDamper(name='s1', c=44650.0, d=2120.0)
6 m1=Mass(name='m1',m=3961.0)
7 reqs=s1.pinEqs(1,[m1.pins[0]]) # список додаткових рівнянь
8 s=System(els=[s1,m1], eqs=reqs) # система
9 prnt(s.eqs)
10
11 bc={s1.x1:0.0, s1.Dx1:0.0}
12 eq=s.eqs.subs(bc) # постійні граничні умови
13 prnt(eq)
14
15 # статика - колона розтягнута на 1 м
16 ics={m1.x:-1.0, m1.v:0.0, m1.a:0.0, s1.Dx2:0.0}
17 state=s.solve(eq,ics)
18 state.update(ics)
19
20 # динаміка - вільні коливання колони
21 eq=eq.subs({m1.f2:0.0}) # додаткові граничні умови
22 T,Y,Yd=s.solveDAE(eq, state, 10.0)
23
24 import matplotlib.pyplot as plt
25 index=s.y.index(m1.x)
26 plt.plot(T, [v[index] for v in Y])
27 plt.show()

```

Лістинг Є.8 – main1Sym.py

```

1 # -*- coding: utf-8 -*-
2 from pycodynDAE import *

```

```

3
4 # створити компоненти:
5 s1=SpringDamper(name='s1', c=44650.0, d=2120.0)
6 m1=Mass(name='m1', m=3961.0)
7 reqs=s1.pinEqs(1,[m1.pins[0]]) # список додаткових рівнянь
8 s=System(els=[s1,m1], eqs=reqs) # система
9 prnt(s.eqs)
10 eq=s.eqs.subs({m1.f2:0.0, s1.x1:0.0, s1.Dx1:0.0}) # граничні умови
11 prnt(eq)
12
13 # усунути змінні вручну:
14 eq=eq.subs({s1.f2:-s1.f1, m1.f1:s1.f1, s1.x2:m1.x, s1.Dx2:m1.Dx,
    s1.vrel:m1.v, m1.a:m1.Dv})
15 eq=eq.subs(s1.f1, solve(eq[3], s1.f1)[0])
16 eq=Tuple(*set(eq)-{True})
17
18 # або автоматично (спрощений алгоритм):
19 # from eliminate import *
20 # eq=eliminate(eq, keep={m1.Dv,m1.Dx,m1.v,m1.x}, maxEqLen=2)
21 prnt(eq)
22
23 # підстановка функцій і похідних
24 eq=eq.subs({m1.x:Function('m1_x')(t), m1.v:Function('m1_v')(t)})
25 eq=eq.subs({m1.Dx:Derivative('m1_x(t)', t), m1.Dv:Derivative('m1_v(t)',
    t)})
26 prnt(eq)
27 eq=dsolve(eq, ics={Function('m1_x')(0):-1.0, Function('m1_v')(0):0.0}) #
    розв'язати ODE аналітично
28 print(eq)
29 plot(eq[1].rhs, xlim=(0,10), ylim=(-1,1))

```

Лістинг Є.9 – main2s.py

```

# encoding: utf-8
from rycodyn import *

fs=-34687.0 # вага снкції
fr=-18499.0 # вага рідини над плунжером
# компоненти:
s1=SpringDamper(name='s1', c=44650.0, d=2120.7)
m1=Mass(name='m1', m=3961.0)
f1=Force(name='f1')#, f=fs+fr

# додаткові рівняння моделі колони, утворені шляхом з'єднання фланців
компонентів
reqs=s1.pinEqs(1,[m1.pins[0]])
reqs+=m1.pinEqs(1,[f1.pins[0]])
s=System(els=[s1,m1,f1], eqs=reqs) # система

# статична задача - колона під дією максимальних статичних навантажень

```

```

ics={m1.v:0.0, m1.a:0.0, s1.x1:0.0, s1.x1p:0.0, f1.f:fs+fr}
d=s.solve(ics)
print(d[m1.x])

def motion(t):
    """описує гармонічний рух верхньої точки колони і повертає її
    переміщення в час t"""
    A=2.1/2 # амплітуда
    n=6.4/60 # частота
    return A*math.sin(2*math.pi*n*t) # переміщення

def force(v):
    """повертає значення сили на плунжері насоса F, в залежності від його
    швидкості v"""
    F=fs # вага секції
    if v>0: # якщо рух вверх
        F+=fr # збільшити силу на значення ваги рідини
    return F*math.tanh(abs(v)/0.01) # згладжування біля точки v=0

def fnBC(d, t):
    """граничні умови у час t для компонентів fnBC.vrs"""
    val = motion(t), force(d[m1.v])
    return dict(zip(fnBC.vrs, val))
fnBC.vrs = s.elsd['s1'].x1, s.elsd['f1'].f

# розв'язати динамічну задачу - верхня точка має гармонічний рух
T,R=s.solveDyn(d, timeEnd=2*60/6.4, fnBC=fnBC)
R=[r for t,r in zip(T,R) if t>60/6.4] # тільки для останнього періоду
plt.plot([d[s1.x1] for d in R], [d[s1.f1]/1000 for d in R]) # гирлова
динамограма
plt.plot([d[m1.x] for d in R], [(-d[m1.f2]+fs)/1000 for d in R]) #
плунжерна динамограма
plt.xlabel('x, m'); plt.ylabel('f, kN')
plt.show()

```

Лістинг Є.10 – main2.py

```

1 # encoding: utf-8
2 from pycodyn import *
3
4 fs=(-18494.0, -16193.0) # вага секцій
5 fr=-18499.0 # вага рідини над плунжером
6 # компоненти:
7 s1=SpringDamper(name='s1', c=114926.0, d=5458.0)
8 m1=Mass(name='m1', m=2112.0)
9 f1=Force(name='f1', f=fs[0])
10 s2=SpringDamper(name='s2', c=73021.0, d=3468.0)
11 m2=Mass(name='m2', m=1850.0)
12 f2=Force(name='f2') #, f=fs[1]+fr
13 # додаткові рівняння моделі колони, утворені шляхом з'єднання фланців
    компонентів

```

```

14 peqs=s1.pinEqs(1,[m1.pins[0]])
15 peqs+=m1.pinEqs(1,[s2.pins[0],f1.pins[0]])
16 peqs+=s2.pinEqs(1,[m2.pins[0]])
17 peqs+=m2.pinEqs(1,[f2.pins[0]])
18 s=System(els=[s1,m1,s2,m2,f1,f2], eqs=peqs) # система
19
20 # статична задача - колона під дією максимальних статичних навантажень
21 ics={m1.v:0.0, m1.a:0.0, m2.v:0.0, m2.a:0.0, s1.x1:0.0, s1.x1p:0.0,
      f2.f:fs[1]+fr}
22 d=s.solve(ics)
23 print(d[m2.x])
24
25 def motion(t):
26     """описує гармонічний рух верхньої точки колони і повертає її
    переміщення в час t"""
27     A=2.1/2 # амплітуда
28     n=6.4/60 # частота
29     return A*math.sin(2*math.pi*n*t) # переміщення
30
31 def force(v):
32     """повертає значення сили на плунжері насоса F, в залежності від
    його швидкості v"""
33     F=fs[1] # вага другої секції
34     if v>0: # якщо рух вверх
35         F+=fr # збільшити силу на значення ваги рідини
36     return F*math.tanh(abs(v)/0.01) # згладжування біля точки v=0
37
38 def fnBC(d, t):
39     """граничні умови у час t для компонентів fnBC.vrs"""
40     val = motion(t), force(d[m2.v])
41     return dict(zip(fnBC.vrs, val))
42 fnBC.vrs = s.elsd['s1'].x1, s.elsd['f2'].f
43
44 # розв'язати динамічну задачу - верхня точка має гармонічний рух
45 T,R=s.solveDyn(d, timeEnd=2*60/6.4, fnBC=fnBC)
46 R=[r for t,r in zip(T,R) if t>60/6.4] # тільки останній період
47 plt.plot([d[s1.x1] for d in R], [d[s1.f1]/1000 for d in R]) # гирлова
    динамограма
48 plt.plot([d[m2.x] for d in R], [(-d[m2.f2]+fs[1])/1000 for d in R]) #
    плунжерна динамограма
49 plt.xlabel('x, m'); plt.ylabel('f, kN')
50 plt.show()

```

Лістинг Є.11 – main2V.py

```

1 # encoding: utf-8
2 from pycodyn import *
3
4 def event(self, state): # обробник подій
5     # симуляція обриву другої секції, коли сила>56000

```

```

6     if state[s1.f1]>56000:
7         self.fnBC=fnBC2
8         peqs=s1.pinEqs(1,[m1.pins[0]])
9         peqs+=m1.pinEqs(1,[f1.pins[0]])
10        self.__init__(els=[s1,m1,f1], eqs=peqs) # зміна системи
11        self.createCurEqs(fnBC2) # нові поточні рівняння
12 System.event=event
13
14 fs=(-18494.0, -16193.0) # вага секцій
15 fr=-18499.0 # вага рідини
16 # компоненти:
17 s1=SpringDamper(name='s1', c=114926.0, d=5458.0)
18 m1=Mass(name='m1',m=2112.0)
19 f1=Force(name='f1', f=fs[0])
20 s2=SpringDamper(name='s2', c=73021.0, d=3468.0)
21 m2=Mass(name='m2', m=1850.0)
22 f2=Force(name='f2')
23 # додаткові рівняння:
24 peqs=s1.pinEqs(1,[m1.pins[0]])
25 peqs+=m1.pinEqs(1,[s2.pins[0],f1.pins[0]])
26 peqs+=s2.pinEqs(1,[m2.pins[0]])
27 peqs+=m2.pinEqs(1,[f2.pins[0]])
28 s=System(els=[s1,m1,s2,m2,f1,f2], eqs=peqs) # система
29
30 # статична задача - колона під дією максимальних статичних навантажень
31 ics={m1.v:0.0, m1.a:0.0, m2.v:0.0, m2.a:0.0, s1.x1:0.0, s1.x1p:0.0,
      f2.f:fs[1]+fr}
32 d=s.solve(ics)
33 print(d[m2.x])
34
35 def motion(t):
36     """описує гармонічний рух верхньої точки колони і повертає її
      переміщення в час t"""
37     A=2.1/2 # амплітуда
38     n=6.4/60 # частота
39     return A*math.sin(2*math.pi*n*t) # переміщення
40
41 def force(v):
42     """повертає значення сили на плунжері насоса F, в залежності від
      його швидкості v"""
43     F=fs[1] # вага другої секції
44     if v>0: # якщо рух ввверх
45         F+=fr # збільшити силу на значення ваги рідини
46     return F*math.tanh(abs(v)/0.01) # згладжування біля точки v=0
47
48 def fnBC(d, t):
49     """граничні умови у час t для компонентів fnBC.vrs"""
50     val = motion(t), force(d[m2.v])
51     return dict(zip(fnBC.vrs, val))
52 fnBC.vrs = s.elsd['s1'].x1, s.elsd['f2'].f
53

```

```

54 def fnBC2(d, t):
55     """граничні умови 2 у час t для компонентів fnBC2.vrs"""
56     val = (motion(t), )
57     return dict(zip(fnBC.vrs, val))
58 fnBC2.vrs = (s.elsd['s1'].x1, )
59
60 # розв'язати динамічну задачу - верхня точка має гармонічний рух
61 T,R=s.solveDyn(d, timeEnd=2*60/6.4+10, fnBC=fnBC)
62 plt.plot([d[s1.x1] for d in R], [d[s1.f1]/1000 for d in R]) # гирлова
    динамограма
63 plt.xlabel('x, m'); plt.ylabel('f, kN')
64 plt.show()

```

Лістинг С.12 – main2sDAE.py

```

1  # -*- coding: utf-8 -*-
2  from rucodynDAE import *
3
4  fs=-34687.0 # вага секцій
5  fr=-18499.0 # вага рідини
6  # компоненти:
7  s1=SpringDamper(name='s1', c=44650.0, d=2120.7)
8  m1=Mass(name='m1', m=3961.0)
9  reqs=s1.pinEqs(1,[m1.pins[0]]) # список додаткових рівнянь
10 s=System(els=[s1,m1], eqs=reqs) # система
11 prnt(s.eqs)
12
13 # статична задача - колона під дією максимальних статичних навантажень
14 ics={s1.x1:0.0, s1.Dx1:0.0, m1.f2:fs+fr, m1.v:0.0, m1.a:0.0, s1.Dx2:0.0}
15 state=s.solve(s.eqs,ics)
16 state.update(ics)
17
18 # динамічна задача - верхня точка має гармонічний рух
19 A=2.1/2 # амплітуда
20 n=6.4/60 # частота
21 eq=s.eqs.subs({s1.x1: A*sin(2*pi*n*t), s1.Dx1: A*sin(2*pi*n*t).diff(t),
    m1.f2: Piecewise((fs, m1.v<0), (fs+fr*tanh(abs(m1.v)/0.01), m1.v>=0))})
22 # або додати рівняння:
23 #eq=s.eqs+Tuple(Eq(s1.x1, A*sin(2*pi*n*t)), Eq(m1.f2, Piecewise((fs,
    m1.v<0), (fs+fr*tanh(abs(m1.v)/0.01), m1.v>=0)))) )
24 T,Y,Yd=s.solveDAE(eq, state, 20.0)
25
26 import matplotlib.pyplot as plt
27 index=s.y.index(s1.f1)
28 Y=[y for t,y in zip(T,Y) if t>60/6.4] # тільки останній період
29 T=[t for t in T if t>60/6.4]
30 plt.plot(A*np.sin(2*np.pi*n*np.array(T)), [v[index]/1000 for v in Y])
31 plt.show()

```

Лістинг Є.13 – main2DAE.py

```

1 # -*- coding: utf-8 -*-
2 from pycodynDAE import *
3
4 fs=(-18494.0, -16193.0) # вага секцій
5 fr=-18499.0 # вага рідини
6 # компоненти:
7 s1=SpringDamper(name='s1', c=114926.0, d=5458.0)
8 m1=Mass(name='m1', m=2112.0)
9 f1=Force(name='f1', f=fs[0])
10 s2=SpringDamper(name='s2', c=73021.0, d=3468.0)
11 m2=Mass(name='m2', m=1850.0)
12 f2=Force(name='f2') #, f=fs[1]+fr
13 # додаткові рівняння
14 reqs=s1.pinEqs(1,[m1.pins[0]])
15 reqs+=m1.pinEqs(1,[s2.pins[0],f1.pins[0]])
16 reqs+=s2.pinEqs(1,[m2.pins[0]])
17 reqs+=m2.pinEqs(1,[f2.pins[0]])
18 s=System(els=[s1,m1,s2,m2,f1,f2], eqs=reqs) # система
19 prnt(s.eqs)
20
21 # статична задача - колона під дією максимальних статичних навантажень
22 ics={s1.x1:0.0, s1.Dx1:0.0, f2.f:fs[1]+fr, m1.v:0.0, m1.a:0.0,
      s1.Dx2:0.0, s2.Dx1:0.0, s2.Dx2:0.0, m2.v:0.0, m2.a:0.0}
23 state=s.solve(s.eqs,ics)
24 state.update(ics)
25
26 # динамічна задача - верхня точка має гармонічний рух
27 A=2.1/2 # амплітуда
28 n=6.4/60 # частота
29 eq=s.eqs.subs({s1.x1: A*sin(2*pi*n*t), s1.Dx1: A*sin(2*pi*n*t).diff(t),
      m2.f2: Piecewise((fs[1], m2.v<0), (fs[1]+fr*tanh(abs(m2.v)/0.01),
      m2.v>=0))})
30 # або додати рівняння:
31 #eq=s.eqs+Tuple(Eq(s1.x1, A*sin(2*pi*n*t)), Eq(m1.f2, Piecewise((fs,
      m1.v<0), (fs+fr*tanh(abs(m1.v)/0.01), m1.v>=0))) )
32
33 T,Y,Yd=s.solveDAE(eq, state, 2*60/6.4)
34
35 import matplotlib.pyplot as plt
36 index=s.y.index(s1.f1)
37 Y=[y for t,y in zip(T,Y) if t>60/6.4] # тільки останній період
38 T=[t for t in T if t>60/6.4]
39 plt.plot(A*np.sin(2*np.pi*n*np.array(T)), [v[index]/1000 for v in Y])
40 plt.show()

```

Лістинг Є.14 – eliminate.py

```

from sympy import *
def eliminate(eq, keep, maxEqLen=2):

```

```
"""Усуває змінні з рівнянь eq з метою їхнього спрощення
keep - намагались зберегти ці змінні
maxEqLen - максимальна кількість рівнянь після усунення"""
while len(eq)>maxEqLen:
    e=solve(eq, eq.free_symbols-keep, exclude=keep) # для усунення
    e=sorted(e.items(), key=lambda x:len(x[1].atoms())) # спочатку
найкоротші вирази
    eq=eq.subs(e[0][0], e[0][1]) # усунути
    eq=Tuple(*set(eq)-set([True])) # без дублікатів і True
return eq
```


ДОДАТОК Ж

Компонентно-орієнтована модель верстата-качалки для симуляції кінематики

Код програми також доступний в GitHub (vkopey/MMSKDM: Methods of modeling and simulation of kinematics and dynamics of machines. URL: <http://github.com/vkopey/MMSKDM>).

Лістинг Ж.1 – mehanizm_geom7.py

```
#encoding: utf-8
# визначення положення, швидкості і прискорення ланок за кутом повороту а
# кривошипа

from __future__ import division
import matplotlib.pyplot as plt
from scipy.optimize import root # функція для розв'язування системи рівнянь
from math import pi, sin, cos, tan, degrees, atan

class Frame:
    "Компонент описує ланку механізму"
    def __init__(self, x1, y1, x2, y2, L):
        "x1, y1, x2, y2, L - координати точок і довжина ланки"
        self.x1, self.y1, self.x2, self.y2, self.L = x1, y1, x2, y2, L
    def eqs(self): # система рівнянь компонента
        eqs = []
        eqs += [(self.x2 - self.x1)**2 + (self.y2 - self.y1)**2 - self.L**2] #
незмінна відстань між точками
        return eqs
    def plot(self): # рисує компонент
        plt.plot([self.x1, self.x2], [self.y1, self.y2], 'ko-')

class Connector:
    "Компонент описує шарнірне з'єднання двох ланок"
    def __init__(self, e1, e2):
        "e1, e2 - дві ланки, які з'єднуються точками 2 і 1 відповідно"
        self.e1, self.e2 = e1, e2
    def eqs(self): # система рівнянь компонента
        eqs = []
        eqs += [self.e1.x2 - self.e2.x1] # e1.x2 = e2.x1
        eqs += [self.e1.y2 - self.e2.y1] # e1.y2 = e2.y1
        return eqs
    def plot(self): # рисує компонент
        plt.plot([self.e1.x2], [self.e1.y2], 'ro')
```

```

class Connector2:
    "Компонент описує нерухоме з'єднання двох ланок"
    def __init__(self,e1,e2):
        "e1,e2 - дві ланки, які з'єднуються точками 2 і 1 відповідно"
        self.e1,self.e2=e1,e2
    def eqs(self): # система рівнянь компонента
        eqs=[]
        eqs+= [self.e1.x2-self.e2.x1] # e1.x2=e2.x1
        eqs+= [self.e1.y2-self.e2.y1] # e1.y2=e2.y1
        # однаковий кут повороту ланок: tan(a1)=tan(a2)
        eqs+= [(self.e1.y1-self.e1.y2)/(self.e1.x1-self.e1.x2)-(self.e2.y2-
self.e2.y1)/(self.e2.x2-self.e2.x1)]
        return eqs
    def plot(self): # рисує компонент
        plt.plot([self.e1.x2],[self.e1.y2],'yo')

class System:
    "Система компонентів"
    def __init__(self,e):
        "e - список компонентів"
        self.e=e
    def eqs(self): # система усіх рівнянь, які описують поведінку системи
        eqs=[]
        for ei in self.e: # для кожного компонента
            eqs+= ei.eqs() # додати в список рівнянь усі рівняння
компонента
        return eqs
    def plot(self): # рисує усі компоненти системи
        for ei in self.e:
            ei.plot()

def f(X, s): # векторна функція повертає значення лівих частин рівнянь
    exec rootstr+"=X" # невідомі (X - вектор початкових наближень)
    eqs=s.eqs()
    return eqs # якщо усі елементи eqs близькі до 0, то X - шукані корені

t,a=0,0 # початкові значення часу і кута повороту кривошипа
T,X=[],[] # списки значень часу і невідомих
d = type('', (), dict(xa=0, ya=0, xb=-1.345, yb=3.01195, L0=0.81371,
L1=3.0, L2=2.0, L3=2.29))() # відомі постійні параметри механізму
fr0=Frame(x1=0.0,y1=0.0, x2=0.81371,y2=0, L=d.L0) # кривошип
fr1=Frame(x1=0.81371,y1=0, x2=0.65,y2=3, L=3.0) # шатун
fr2=Frame(x1=0.65,y1=3, x2=-1.345,y2=3.01195, L=2.0) # заднє плече
балансира
fr3=Frame(x1=-1.345,y1=3.01195, x2=-3.6,y2=3, L=2.29) # переднє плече
балансира
con0=Connector(fr0,fr1) # шарнір між кривошипом і шатуном
con1=Connector(fr1,fr2) # шарнір між шатуном і балансиром
con2=Connector2(fr2,fr3) # нерухоме з'єднання плечей балансира
s=System([fr0, fr1, fr2, fr3, con0, con1, con2]) # механізм верстата-
гойдалки

```

```

rootstr="s.e[1].x1, s.e[1].y1, s.e[1].x2, s.e[1].y2, s.e[2].x1, s.e[2].y1,
s.e[3].x2, s.e[3].y2"
# рядок rootstr потрібен щоб назви коренів записувались в програмі тільки
один раз

while a<2*pi: # поки кут < 360 градусів
    s.e[0].x2=s.e[0].L*cos(a)+s.e[0].x1 # координата точки кривошипа
dx/L=cos(a)
    s.e[0].y2=s.e[0].L*sin(a)+s.e[0].y1 # координата точки кривошипа
dy/L=sin(a)
    # увага! потрібно запобігати тому щоб початкові наближення коренів =0
    roots=[x+0.001 for x in eval(rootstr)] # наприклад шляхом додавання
0.001
    sol = root(f, roots, args=(s,), method='lm') # розв'язати систему
рівнянь
    exec rootstr+"=sol.x" # корені
    #print eval(rootstr)
    b=atan((s.e[3].y2-s.e[3].y1)/(s.e[3].x2-s.e[3].x1)) # кут повороту
балансира atan(dy/dx)
    x,y=s.e[0].x1+s.e[3].x1-s.e[3].L, -s.e[3].L*b # координати точки
підвісу
    plt.plot([x],[y],'bo')
    plt.text(x+0.1, y, t)
    plt.text(s.e[0].x2, s.e[0].y2, t)
    s.plot() # нарисувати положення ланок механізму
    T.append(t)
    X.append(y)
    t+=1 # збільшити час на крок
    a+=pi/16 #15.5 # збільшити кут на крок
plt.show()

import scipy
dt=T[1]-T[0]
V=scipy.gradient(X, dt) # швидкість (central differences)
A=scipy.gradient(V, dt) # прискорення (central differences)
plt.plot(T, X, 'k-', lw=2)
plt.plot(T, V, 'k--', lw=2)
plt.plot(T, A, 'r', lw=2)
plt.show()

```

ДОДАТОК 3

Python-класи для перебудови параметричної моделі верстата-качалки у SOLIDWORKS

Код програми та інші компоненти моделі доступні в GitHub (vkopey/PumpingUnitModel: Parametric model of pumping unit in SOLIDWORKS. URL: <http://github.com/vkopey/PumpingUnitModel>).

Лістинг 3.1 – PumpingUnit.py

```
# -*- coding: CP1251 -*-
import os
import codecs
import subprocess
from math import *

class SWmodel(object):
    """Клас моделі SolidWorks"""
    d={} # словник параметрів
    fileName=None # ім'я файлу моделі
    fileType=".SLDPRT" # розширення файлу моделі (".SLDPRT", ".SLDASM")

    def create(self):
        """Розраховує параметри моделі"""
        pass

    def rebuildModel(self):
        """Перебудовує файл рівнянь та модель SolidWorks"""
        self.write_dict_to_SW_equations()
        self.rebuildAndSaveModel()

    def rebuildAndSaveModel(self):
        """Перебудовує та зберігає SolidWorks модель шляхом виконання VBS
скрипта"""

        vbs=r"""'Скрипт VBS для перебудови моделі SolidWorks
Dim swApp 'SldWorks.SldWorks
Dim Part 'SldWorks.ModelDoc2
Set swApp = CreateObject("SldWorks.Application")
Set Part = swApp.OpenDoc("{fullFileNameExt}", {docType})
Set Part = swApp.ActivateDoc("{fileNameExt}")
Part.EditRebuild
Part.SaveSilent
Set Part = Nothing
swApp.CloseDoc "{fileName}"""
```

```

Set swApp=Nothing
"""
    fileName=self.fileName
    fileNameExt=self.fileName+self.fileType
    fullFileNameExt=os.path.join(os.getcwd(), fileNameExt)
    if self.fileType==".SLDPRT":
        docType="1" # якщо деталь
    else:
        docType="2" # якщо збірка
    vbs=vbs.format(fullFileNameExt=fullFileNameExt,
fileNameExt=fileNameExt, fileName=fileName, docType=docType)
    scriptFileName=os.path.join(os.getcwd(), "RebuildSWmodelTemp.vbs")
    f=open(scriptFileName, 'w')
    f.write(vbs)
    f.close()
    # виконує процес та чекає його завершення
    subprocess.Popen(r'c:\Windows\system32\wscript.exe
'+scriptFileName).wait()

def read_dict_from_SW_equations(self):
    """Додає елементи в словник self.d
з текстового файлу (utf-8-sig) рівнянь SolidWorks"""
    filename=self.fileName+".txt" # файл рівнянь SolidWorks
    if not os.path.exists(filename): return # якщо файлу не існує,
ВИЙТИ
    f=codecs.open(filename, 'r', 'utf-8-sig') # відкрити файл для читання
    for line in f.readlines(): # для усіх рядків у списку
        if '=' in line: # якщо в рядку є символ "="
            pair=line.split('=') # розділити рядок
            pair=pair[0].strip()[1:-1], pair[1].strip() # видалити
пробіли і лапки

            if pair[1].isdigit(): # якщо всі символи цифри
                val=int(pair[1])
            else: # інакше
                try: # якщо дійсне число
                    val=float(pair[1])
                except ValueError: # якщо рядок
                    val=pair[1]

            self.d[pair[0].encode('CP1251')]=val # записати в словник
    f.close() # закрити файл

def write_dict_to_SW_equations(self):
    """Записує значення елементів словника self.d
в текстовий файл (utf-8-sig) рівнянь SolidWorks"""
    d={} # словник з unicode ключами
    for k in self.d: # перетворюємо ключі в unicode
        d[k.decode('CP1251')]=self.d[k]

    filename=self.fileName+".txt" # файл рівнянь SolidWorks

```

```

if not os.path.exists(filename): return # якщо файлу не існує,
вийти
f=codecs.open(filename,'r','utf-8-sig') # відкрити файл для читання
oldlines=f.readlines() # старий список рядків
newlines=[] # новий список рядків
for line in oldlines: # для усіх рядків у списку
    if '=' in line: # якщо в рядку є символ "="
        pair=line.split('=') # розділити рядок
        pair=pair[0].strip()[1:-1],pair[1].strip() # видалити
пробіли і лапки
        if pair[0] in d.keys(): # якщо ліва частина від "=" є серед
ключів словника
            line='''+pair[0]+'"' = '+str(d[pair[0]])+"\n" #
формувані новий рядок
            newlines.append(line) # додати рядок в новий список рядків
        f.close() # закрити файл

f=codecs.open(filename,'w','utf-8-sig') # відкрити файл для запису
f.writelines(newlines) # записати список нових рядків
f.close() # закрити файл

def assignParam(self, profil, name, nameList):
    """Присвоює значення заданим елементам словника параметрів.
    profil - конструктор відповідного профілю (див. класи профілів),
    name - назва профілю,
    nameList - список назв елементів SolidWorks.
    """
    p=profil()
    p.create(name=name)
    self.d=p.setSWParam(self.d, nameList)

class SWmodelPRT(SWmodel):
    """Клас моделі деталі SolidWorks"""
    fileType=".SLDPRT"

class SWmodelASM(SWmodel):
    """Клас моделі зборки SolidWorks"""
    fileType=".SLDASM"

class PumpingUnit(SWmodelASM):
    """Клас верстата-гойдалки"""
    fileName="Верстат"
    d={
        "Тип" : "СКД8-3-4000",
        "Максимальне допустиме навантаження" : 80.0,
        "Список довжин ходу полірованого штока" : [1200.0, 1600.0, 2000.0,
2500.0, 3000.0],
        "Довжина ходу полірованого штока" : 2000.0,
        "Список кількості гойдань" : [5.0,12.0],
        "Кількість гойдань" : 5.0,
        "Максимальний крутний момент" : 40.0,

```

```

"Довжина переднього плеча балансира" : 2290.0,
"Довжина заднього плеча балансира" : 2000.0,
"Довжина шатуна" : 3000.0,
"Найбільший радіус кривошипа" : 1290.0,
"Радіус кривошипа" : None,
"Горизонтальна відстань між осями опори балансира і тихохідного валу
редуктора" : 1345.0,
"Вертикальна відстань між осями опори балансира і тихохідного валу
редуктора" : None,
"Довжина" : 6900.0,
"Ширина" : 2250.0,
"Висота" : 4910.0,
"Маса" : 11780.0,
"Система урівноважування" : "кривошипна",
"Вага кривошипних противаг" : 750.0,
"Максимальна кількість кривошипних противаг" : 6,
"Номинальна потужність електродвигуна" : 18.5,
"Редуктор" : "Ц2НШ-750 Б"}

def create(self):
    """Розраховує параметри моделі"""
    sList=self.d["Список довжин ходу полірованого штока"]
    s0=self.d["Довжина ходу полірованого штока"]
    # розрахунок вертикальної відстані між осями опори балансира і
тихохідного валу редуктора
    smax=max(sList) # найбільша довжина ходу
    rmax=self.d["Найбільший радіус кривошипа"]
    l=self.d["Довжина шатуна"]
    k=self.d["Довжина заднього плеча балансира"]
    k1=self.d["Довжина переднього плеча балансира"]
    l1=self.d["Горизонтальна відстань між осями опори балансира і
тихохідного валу редуктора"]
    h=smax*k/k1 # вертикальний хід опори траверси
    b=sqrt(k**2-(h/2)**2) # відстань від опори балансира до вертикалі h
    H=sqrt((l-rmax)**2-(b-l1)**2)+h/2
    self.d["Вертикальна відстань між осями опори балансира і
тихохідного валу редуктора"]=H

    # розрахунок радіуса кривошипа
    h=s0*k/k1 # вертикальний хід опори траверси
    b=sqrt(k**2-(h/2)**2) # відстань від опори балансира до вертикалі h
    r=l-sqrt((H-h/2)**2+(b-l1)**2)
    self.d["Радіус кривошипа"]=r

    # створення та розрахунок деталей
    self.GolovBalansir=GolovBalansir()
    self.GolovBalansir.create(paramPU=self.d)
    self.Balansir=Balansir()
    self.Balansir.create(paramPU=self.d,
paramGolovBalansir=self.GolovBalansir.d)

```

```

        self.GolovBalansir.d["Висота між опорами@Профіль
головки"]=self.Balansir.d["Висота@Двотавр профіль"]
        self.Shatun=Shatun()
        self.Shatun.create(paramPU=self.d)
        self.Krivoshyp=Krivoshyp()
        self.Krivoshyp.create(paramPU=self.d)
        self.Traversa=Traversa()
        self.Traversa.create(paramPU=self.d)
        self.Val=Val()
        self.Val.create(paramPU=self.d, paramShatun=self.Shatun.d,
paramTraversa=self.Traversa.d)
        self.Reduktor=Reduktor()
        self.Reduktor.create()
        self.Stiyka=Stiyka()
        self.Stiyka.create(paramPU=self.d, paramReduktor=self.Reduktor.d,
paramVal=self.Val.d, paramKrivoshyp=self.Krivoshyp.d)
        self.Rama=Rama()
        self.Rama.create(paramPU=self.d, paramStiyka=self.Stiyka.d,
paramReduktor=self.Reduktor.d)
        self.Protyvaga=Protyvaga()
        self.Protyvaga.create(paramKrivoshyp=self.Krivoshyp.d)
        # зборки:
        self.BalansirVZbori=BalansirVZbori()
        self.Krivoshypy=Krivoshypy()
        self.TraversaVZbori=TraversaVZbori()
        self.Karkas=Karkas()

def rebuildModel(self):
    self.GolovBalansir.rebuildModel()
    self.Balansir.rebuildModel()
    self.Shatun.rebuildModel()
    self.Krivoshyp.rebuildModel()
    self.Traversa.rebuildModel()
    self.Val.rebuildModel()
    self.Reduktor.rebuildModel()
    self.Stiyka.rebuildModel()
    self.Rama.rebuildModel()
    self.Protyvaga.rebuildModel()
    # зборки:
    self.BalansirVZbori.rebuildModel()
    self.Krivoshypy.rebuildModel()
    self.TraversaVZbori.rebuildModel()
    self.Karkas.rebuildModel()

class Profil(object):
    """Клас профілю"""
    fileName=None
    d={} # словник параметрів

def create(self, name):
    self.getParamFromCSV(name)

```



```

def setSWParam(self,d,nameList):
    """Повертає словник параметрів для SolidWorks.
    d - початковий словник параметрів SolidWorks,
    nameList - список назв елементів SolidWorks.
    Приклад:
    setSWParam({"Висота@Профіль1":0, "Висота@Профіль2":0},
               ["Профіль1","Профіль2"])
    """
    for k in d: # для усіх ключів словника параметрів для SolidWorks
        if k.count('@')==1: # якщо в ключі є тільки 1 символ '@'
            pair=k.split('@') # розділити на дві частини
            if pair[1] in nameList: # якщо назва елемента в списку
                d[k]=self.d[pair[0]] # змінити значення
    return d

def getParamFromCSV(self, name):
    """З файлу CSV записує в словник параметрів параметри профілю з
    назвою name"""
    import csv
    csv_file=open(self.fileName+".csv", "rb")
    reader=csv.DictReader(csv_file, delimiter = ';')
    for row in reader:
        if row["Назва"]==name:
            self.d=row
    csv_file.close()

    # перетворює значення словника в дійсні числа
    for k in self.d:
        if k!="Назва": # окрім назви
            if "," in self.d[k]: # якщо є символ ","
                self.d[k]=float(self.d[k].replace(",","."))
            else:
                self.d[k]=float(self.d[k])

class DvotavrProfil(Profil):
    """Клас двотавра"""
    fileName="Двотаври ГОСТ 8239-89"

class ShvelerProfil(Profil):
    """Клас швелера"""
    fileName="Швелери серія У ДСТУ 3436-96"

class KutnykProfil(Profil):
    """Клас кутника"""
    fileName="Кутник ДСТУ 2251-93"

class Balansir(SWmodelPRT):
    """Клас балансира"""
    fileName="Балансир"
    #словник параметрів

```

```

d={
  "Двотавр номер профілю" : None}

def create(self, paramPU, paramGolovBalansir):
  self.read_dict_from_SW_equations() # читати параметри з файлу
рівнянь

  # профіль двотавра
  self.assignParam(DvotavrProfil, self.d["Двотавр номер профілю"],
["Двотавр профіль"])

  # довжина плеч
  lpp=paramPU["Довжина переднього плеча балансира"]
  lzp=paramPU["Довжина заднього плеча балансира"]
  # довжина головки балансира
  lgb=paramGolovBalansir["Ширина@Профіль головки"]-
paramGolovBalansir["Глибина опори@Профіль головки"]
  self.d["Довжина переднього плеча балансира"]=lpp-lgb
  self.d["Довжина@Двотавр"]=lpp-lgb+self.d["Координата@Профіль опори
балансира"]+self.d["Ширина@Профіль опори балансира"]/2
  self.d["Координата@Профіль опори балансира"]=lzp-
self.d["Ширина@Профіль опори балансира"]/2+self.d["Ширина@Профіль
опори"]/2+self.d["Координата@Профіль опори"]
  # координати ребер жорсткості
  self.d["Координата@Ребро профіль"]=100.0
  self.d["Координата@Масив ребер1"]=(lzp-100)/2
  self.d["Координата@Масив ребер2"]=(lzp+100)
  self.d["Координата@Масив ребер3"]=lzp+lpp/2

class GolovBalansir(SWmodelPRT):
  """Клас головки балансира"""
  fileName="Головка балансира"
  #словник параметрів
  d={
    "Двотавр номер профілю" : None,
    "Радіус@Профіль головки" : 2290.0,
    "Висота між опорами@Профіль головки" : 640.0,
    "Ширина@Профіль головки" : 1000.0,
    "Глибина опори@Профіль головки" : 300.0,
    "Зовнішній радіус@Двотавр профіль" : 6.0,
    "Внутрішній радіус@Двотавр профіль" : 14.0,
    "Товщина середини основи@Двотавр профіль" : 12.3,
    "Товщина стінки@Двотавр профіль" : 7.5,
    "Ширина@Двотавр профіль" : 145.0,
    "Висота@Двотавр профіль" : 360.0,
    "Висота@Перемістити грань1" : 70.0,
    "Висота@Перемістити грань2" : 70.0,
    "Висота осі опори@Профіль головки" : 120.0,
    "Кут@Профіль головки" : 75.0}

  def create(self, paramPU):

```

```

    # профіль двотавра
    self.assignParam(DvotavrProfil, self.d["Двотавр номер профілю"],
["Двотавр профіль"])

    r=paramPU["Довжина переднього плеча балансира"]
    self.d["Радіус@Профіль головки"]=r
    s=max(paramPU["Список довжин ходу полірованого штока"]) # найбільша
    self.d["Кут@Профіль головки"]=degrees(s/r)

class Shatun(SWmodelPRT):
    """Клас шатуна"""
    fileName="Шатун"
    #словник параметрів
    d={"Довжина@Тіло" : 3000.0,
        "Ширина@Опора" : 70.0}
    def create(self, paramPU=None):
        dsh=paramPU["Довжина шатуна"]
        self.d["Довжина@Тіло"]=dsh

class Krivoshyp(SWmodelPRT):
    """Клас кривошипа"""
    fileName="Кривошип"
    #словник параметрів
    d={"Довжина@Ескіз" : 2000.0,
        "Радіус@Ескіз" : 1290.0,
        "Ширина@Профіль" : 150.0,
        "Висота@Ескіз" : 400.0,
        "Координата отвору@Ескіз" : 300.0}

    def create(self, paramPU=None):
        r=paramPU["Радіус кривошипа"]
        rmax=paramPU["Найбільший радіус кривошипа"]
        self.d["Радіус@Ескіз"]=r
        self.d["Довжина@Ескіз"]=rmax+600.0

class Traversa(SWmodelPRT):
    """Клас траверси"""
    fileName="Траверса"
    #словник параметрів
    d={"Зовнішній радіус@Швелер профіль" : 6.0,
        "Внутрішній радіус@Швелер профіль" : 15.0,
        "Товщина середини основи@Швелер профіль" : 13.5,
        "Товщина стінки@Швелер профіль" : 8.0,
        "Кут основи@Швелер профіль" : 5.73,
        "Ширина@Швелер профіль" : 115.0,
        "Висота@Швелер профіль" : 400.0,
        "Півдовжина@Швелер" : 1125.0,
        "Товщина@Профіль ребра" : 10.0,
        "Координата@Профіль ребра" : 100.0,
        "Координата@Масив ребер" : 562.5,
        "Кількість@Масив ребер" : 2,

```

```

"Діаметр@Ескіз отвору під шатун" : 50.0,
"Координата@Ескіз отвору під шатун" : 100.0,
"Висота@Ескіз опори" : 240.0,
"Діаметр отвору@Ескіз опори" : 100.0,
"Висота осі@Ескіз опори" : 120.0,
"Товщина@Опора" : 30.0,
"Координата@Опора" : 170.0,
"Товщина@Профіль основи опори" : 10.0,
"Півдовжина@Основа опори": 240.0}

```

```

def create(self, paramPU=None):
    # профіль швелера
    self.assignParam(ShvelerProfil, self.d["Швелер номер профілю"],
["Швелер профіль"])

```

```

    ln=paramPU["Ширина"]
    self.d["Півдовжина@Швелер"]=ln/2
    self.d["Координата@Масив ребер"]=self.d["Півдовжина@Швелер"]/2

```

```

class Val(SWmodelPRT):
    """Клас валу"""
    fileName="Вал"
    #словник параметрів
    d={"Довжина@Вал" : 2400.0}
    def create(self, paramPU, paramShatun, paramTraversa):
        l1=paramShatun["Ширина@Опора"]
        ln=paramTraversa["Півдовжина@Швелер"]
        l2=paramTraversa["Координата@Ескіз отвору під шатун"]
        self.d["Довжина@Вал"]=2*(ln-l2-l1/2)

```

```

class Reduktor(SWmodelPRT):
    """Клас редуктора"""
    fileName="Редуктор"
    #словник параметрів
    d={"Висота осі@Профіль отвору" : 350.0,
        "Координата осі@Профіль отвору" : 300.0,
        "Довжина@Профіль корпусу" : 1100.0}

```

```

class Stiyka(SWmodelPRT):
    """Клас стійки"""
    fileName="Стійка"
    #словник параметрів
    d={"Ширина@Ескіз основи" : 2000.0,
        "Довжина@Ескіз основи" : 2000.0,
        "Висота@Основа" : 4500.0,
        "Висота@Секція1" : 500.0,
        "Висота@Секція2" : 1500.0,
        "Висота@Секція3" : 2500.0,
        "Висота@Секція4" : 3500.0,
        "Ширина@Ескіз опори балансира" : 400.0,
        "Висота@Ескіз опори балансира" : 400.0,

```

```

"Висота осі@Ескіз опори балансира" : 200.0,
"Товщина@Опора балансира" : 100.0,
"Координата@Опора балансира" : 170.0,
"Товщина@Основа опори" : 20.0,
"Кут" : 8}

def create(self, paramPU, paramReduktor, paramVal, paramKrivoshyp):
    s=paramVal["Довжина@Вал"]-2*paramKrivoshyp["Ширина@Профіль"]-100
    self.d["Ширина@Ескіз основи"]=s
    self.d["Довжина@Ескіз основи"]=s
    h=paramPU["Вертикальна відстань між осями опори балансира і
тихохідного валу редуктора"]
    hr=paramReduktor["Висота осі@Профіль отвору"]
    self.d["Висота@Основа"]=h+hr-self.d["Висота осі@Ескіз опори
балансира"]-self.d["Товщина@Основа опори"]
    hsect1=self.d["Висота@Основа"]/9
    self.d["Висота@Секція1"]=hsect1
    hsect=(self.d["Висота@Основа"]-hsect1)/4
    self.d["Висота@Секція2"]=hsect1+hsect
    self.d["Висота@Секція3"]=hsect1+2*hsect
    self.d["Висота@Секція4"]=hsect1+3*hsect

class Rama(SWmodelPRT):
    """Клас рами"""
    fileName="Рама"
    #словник параметрів
    d={"Довжина@Ескіз" : 5000.0,
      "Ширина@Ескіз" : 2000.0,
      "L1@Ескіз" : 200.0,
      "L2@Ескіз" : 2000.0,
      "L3@Ескіз" : 400.0,
      "L4@Ескіз" : 1000.0,
      "L5@Ескіз" : 300.0}

    def create(self, paramPU, paramStiyka, paramReduktor):
        l0=paramPU["Горизонтальна відстань між осями опори балансира і
тихохідного валу редуктора"]
        s=paramStiyka["Ширина@Ескіз основи"]
        l2=paramStiyka["Довжина@Ескіз основи"]
        lr=paramReduktor["Довжина@Профіль корпусу"]
        ko=paramReduktor["Координата осі@Профіль отвору"]

        self.d["Ширина@Ескіз"]=s
        self.d["L2@Ескіз"]=l2
        self.d["L3@Ескіз"]=l0-l2/2-ko
        self.d["L4@Ескіз"]=lr-100

self.d["Довжина@Ескіз"]=self.d["L1@Ескіз"]+self.d["L2@Ескіз"]+self.d["L3@Ес
кіз"]+self.d["L4@Ескіз"]+self.d["L5@Ескіз"]+700

class Protyvaga(SWmodelPRT):

```

```

"""Клас протываги"""
fileName="Протывага"
#словник параметрів
d={"Радіус@Эскиз1" : 1625.0,
  "Півширина кривошипа@Эскиз1" : 200.0,
  "Ширина@Эскиз1" : 800.0,
  "Товщина@Бобышка-Вытянуть1" : 150.0}

def create(self, paramKrivoshyp):
    self.d["Радіус@Эскиз1"]=paramKrivoshyp["Довжина@Ескиз"]-
paramKrivoshyp["Координата отвору@Ескиз"]
    self.d["Півширина
кривошипа@Эскиз1"]=paramKrivoshyp["Висота@Ескиз"]/2
    self.d["Товщина@Бобышка-
Вытянуть1"]=paramKrivoshyp["Ширина@Профіль"]
    # ширина залежить від маси

class BalansirVZbori(SWmodelASM):
    """Клас балансира в зборі"""
    fileName="Балансир в зборі"

class Krivoshypy(SWmodelASM):
    """Клас кривошипів з валом в зборі"""
    fileName="Кривошипи"

class TraversaVZbori(SWmodelASM):
    """Клас траверси з шатунами в зборі"""
    fileName="Траверса з шатунами"

class Karkas(SWmodelASM):
    """Клас каркасу в зборі"""
    fileName="Каркас"

# pu=PumpingUnit()
# pu.create()
# pu.rebuildModel()

# k=KutnykProfil()
# k.create("150x150x12")
# print k.d["Зовнішній радіус"]

```

ДОДАТОК II

Моделі РЗ для Abaqus/CAE

Доступні також в GitHub (vkopey/ThreadsAbaqus: Abaqus/CAE python scripts for modelling threaded connections of oil and gas equipment. URL: <http://github.com/vkopey/ThreadsAbaqus>).

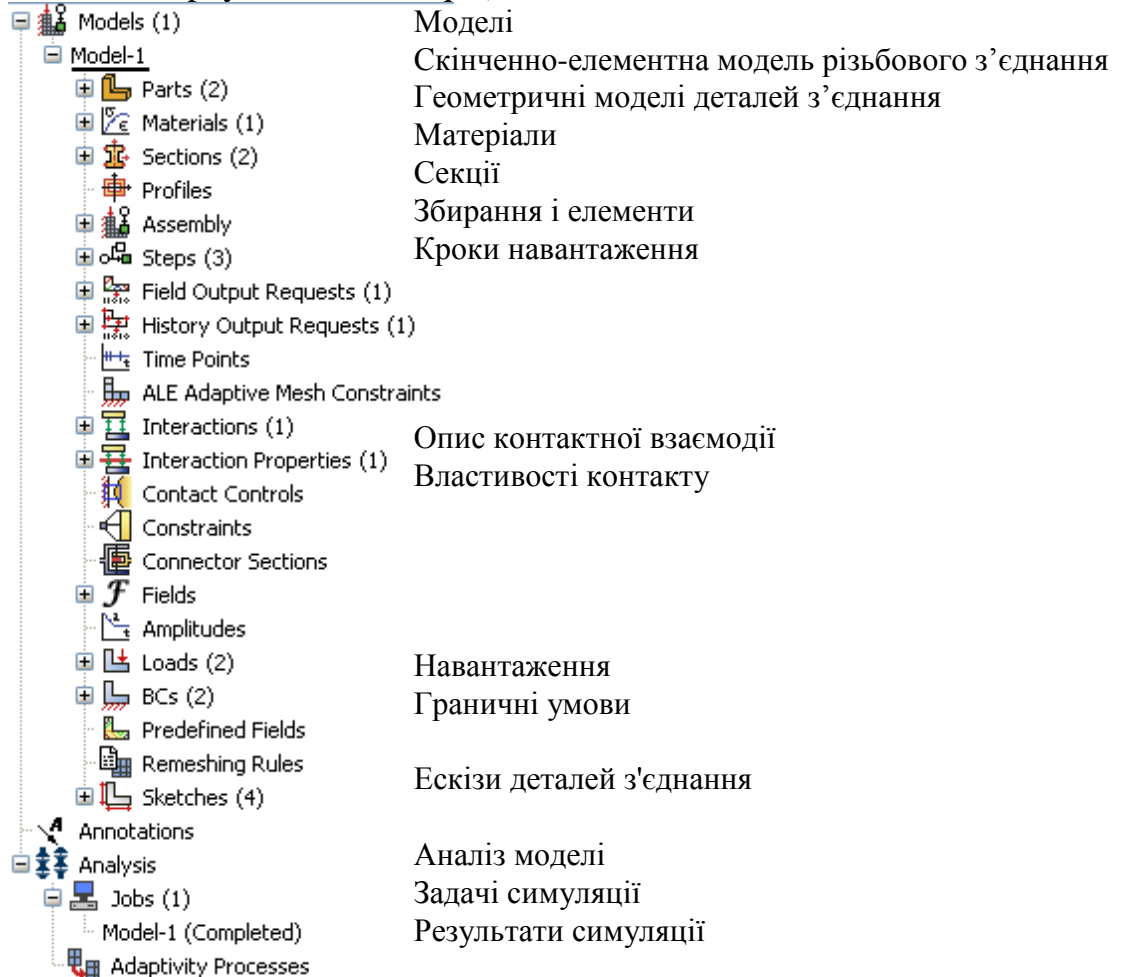


Рисунок II.1 – Дерево побудови моделі муфтового РЗ НКТ в Abaqus 6.8

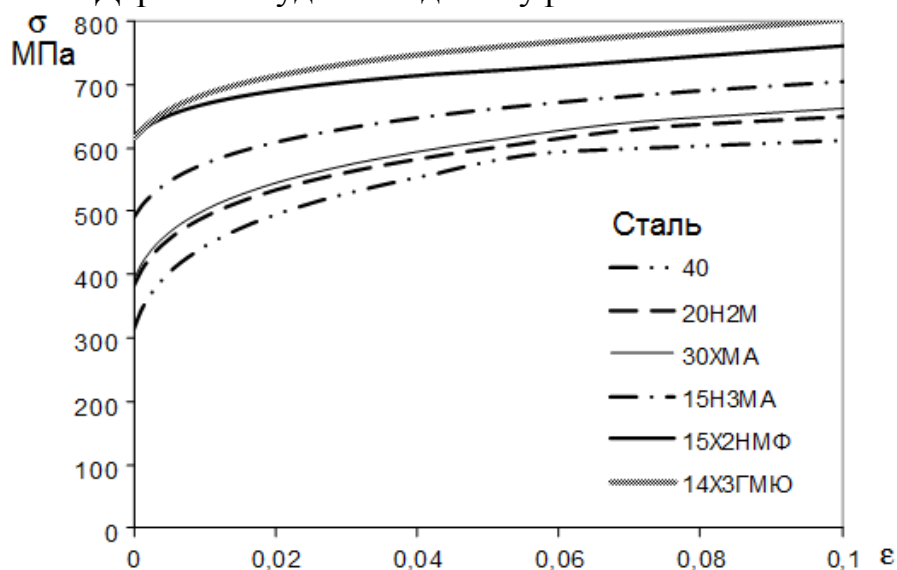
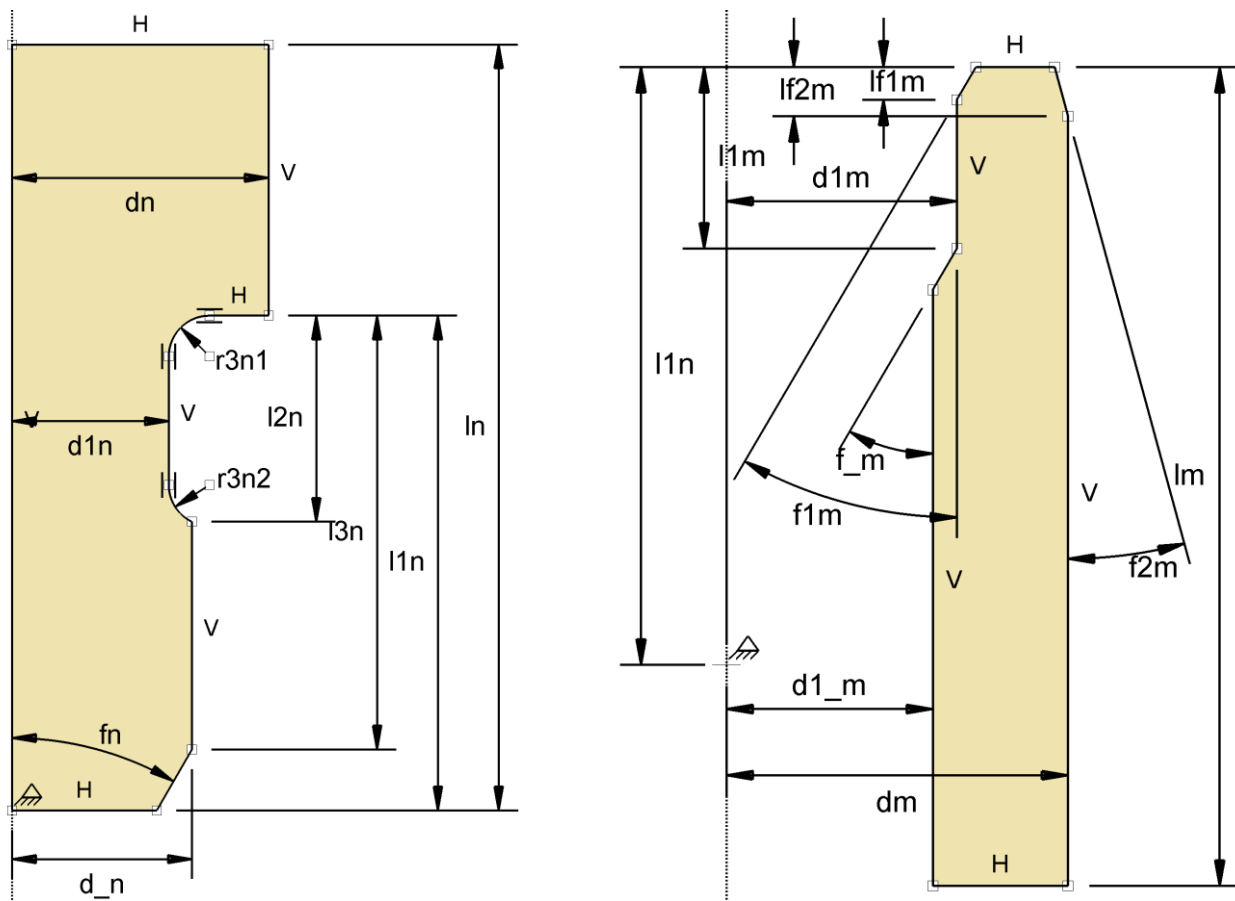
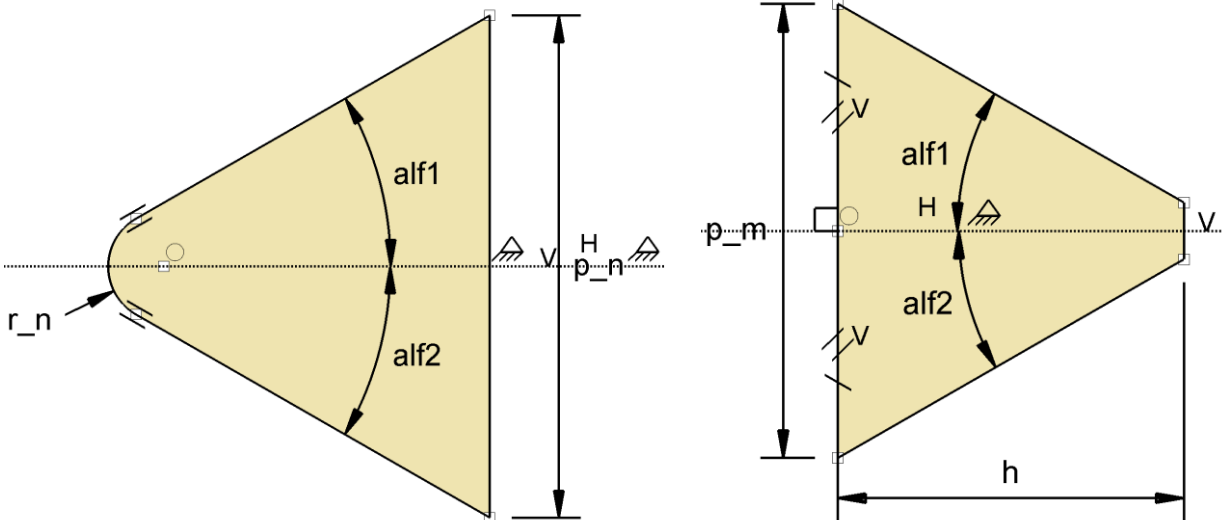


Рисунок II.2 – Моделі істинних діаграм деформування сталей для ШН і муфт



а

б

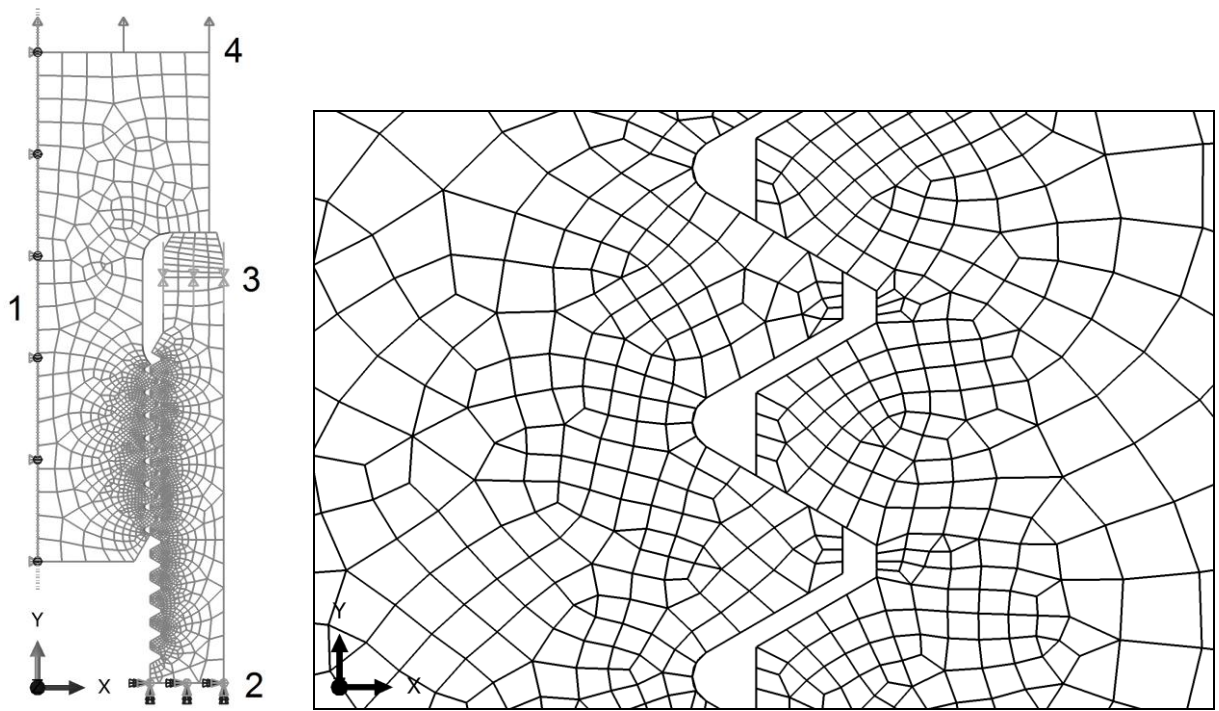


в

г

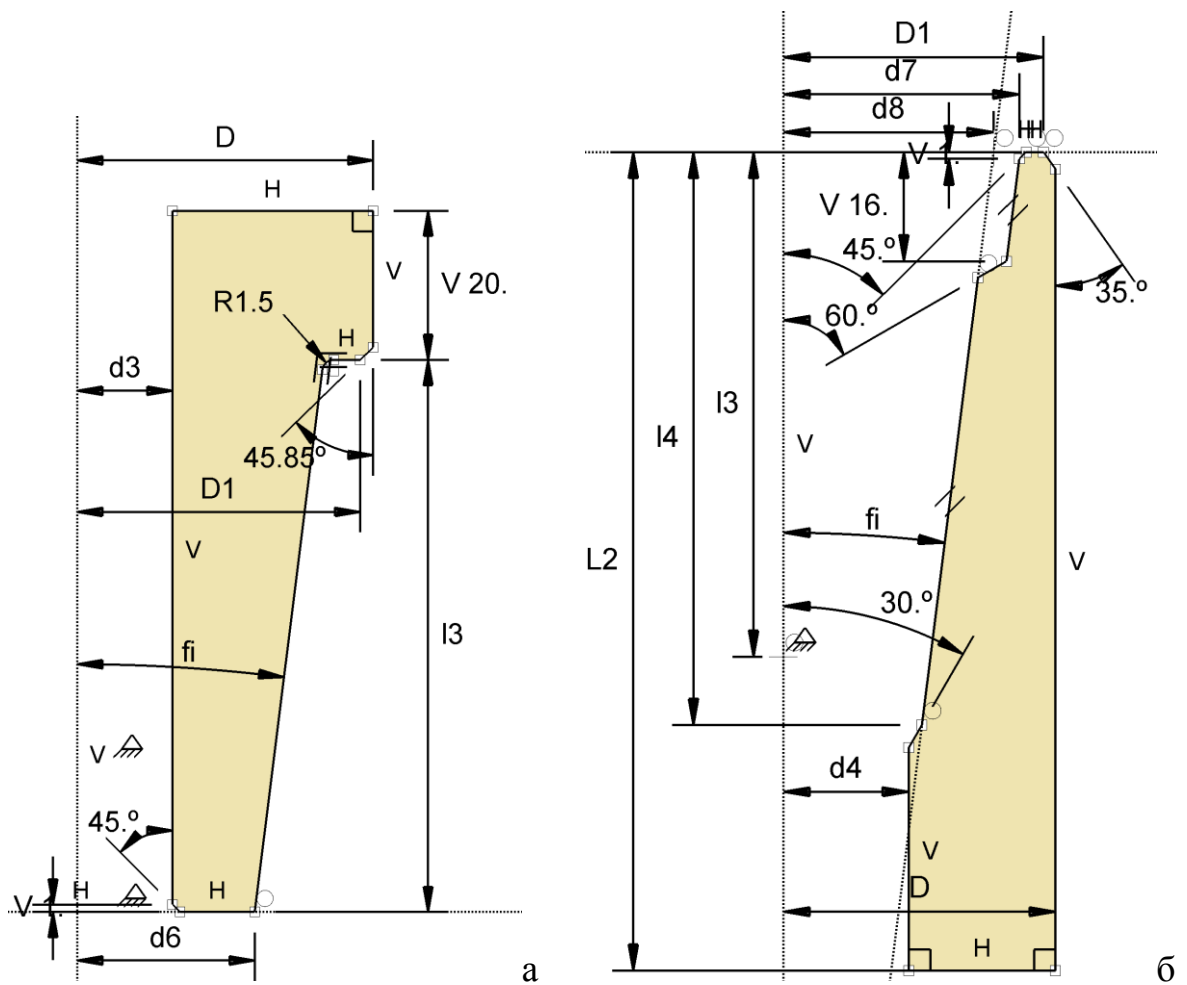
а – заготовки ніпеля, б – заготовки муфти, в – для "вирізання" профілю різьби ніпеля, г – для "вирізання" профілю різьби муфти

Рисунок И.3 – Ескізи для побудови деталей РЗ ШН



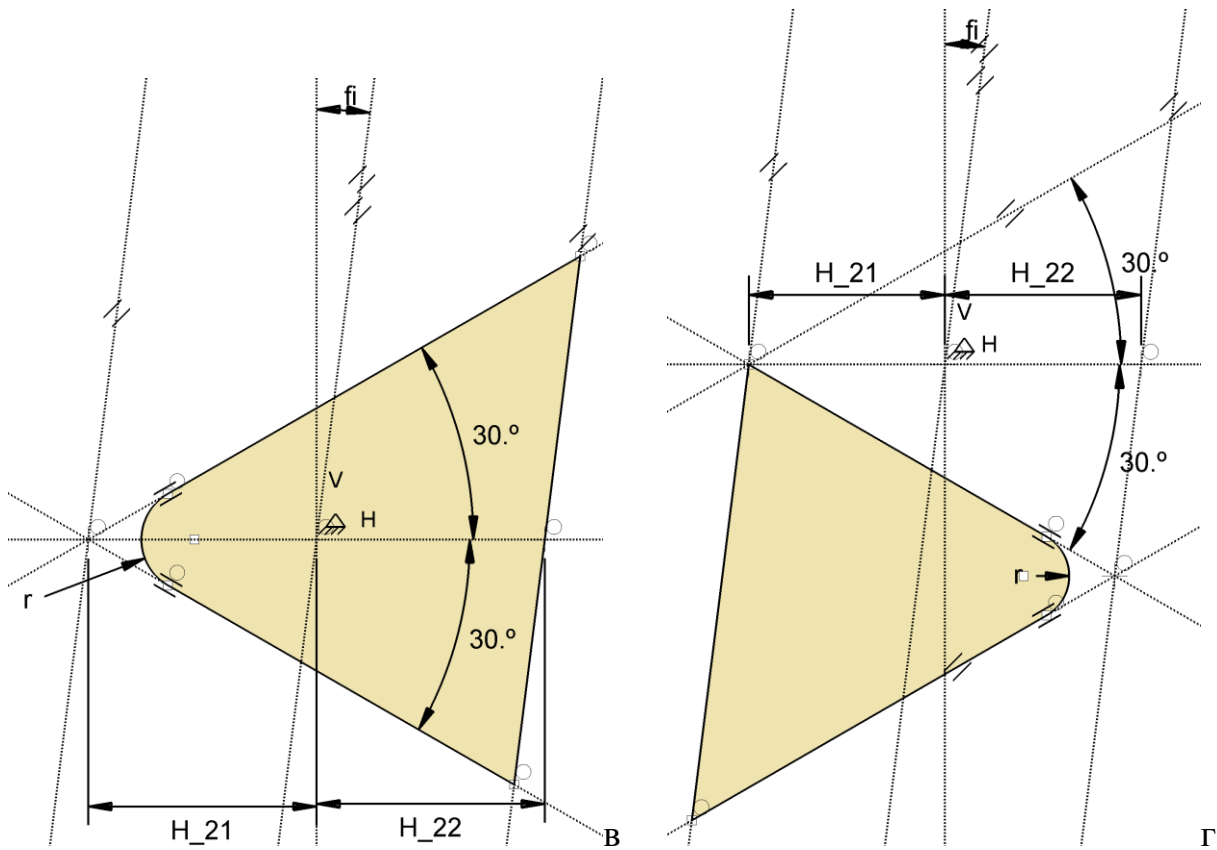
Граничні умови: 1 – (переміщення $U_x=0$); 2 – ($U_x=0, U_y=0$); 3- Δ ; 4 – тиск

Рисунок И.4 – Осесиметрична СЕМ муфтового РЗ ШН діаметром 19 мм



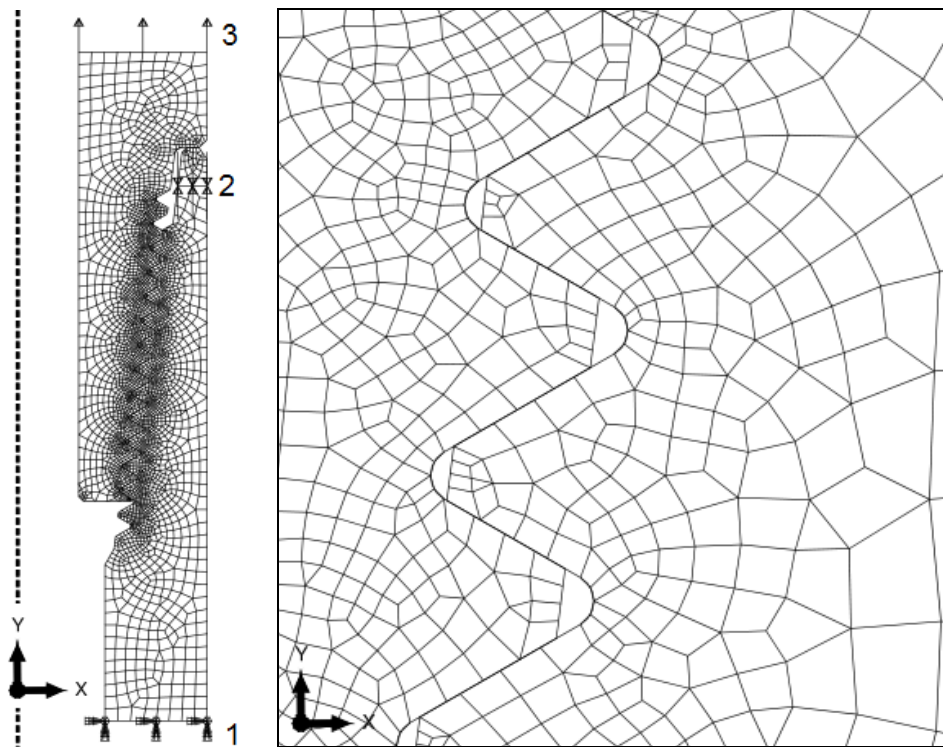
а – заготовки ніпеля, б – заготовки муфти

Рисунок И.5, аркуш 1 – Ескізи для побудови деталей замкового РЗ



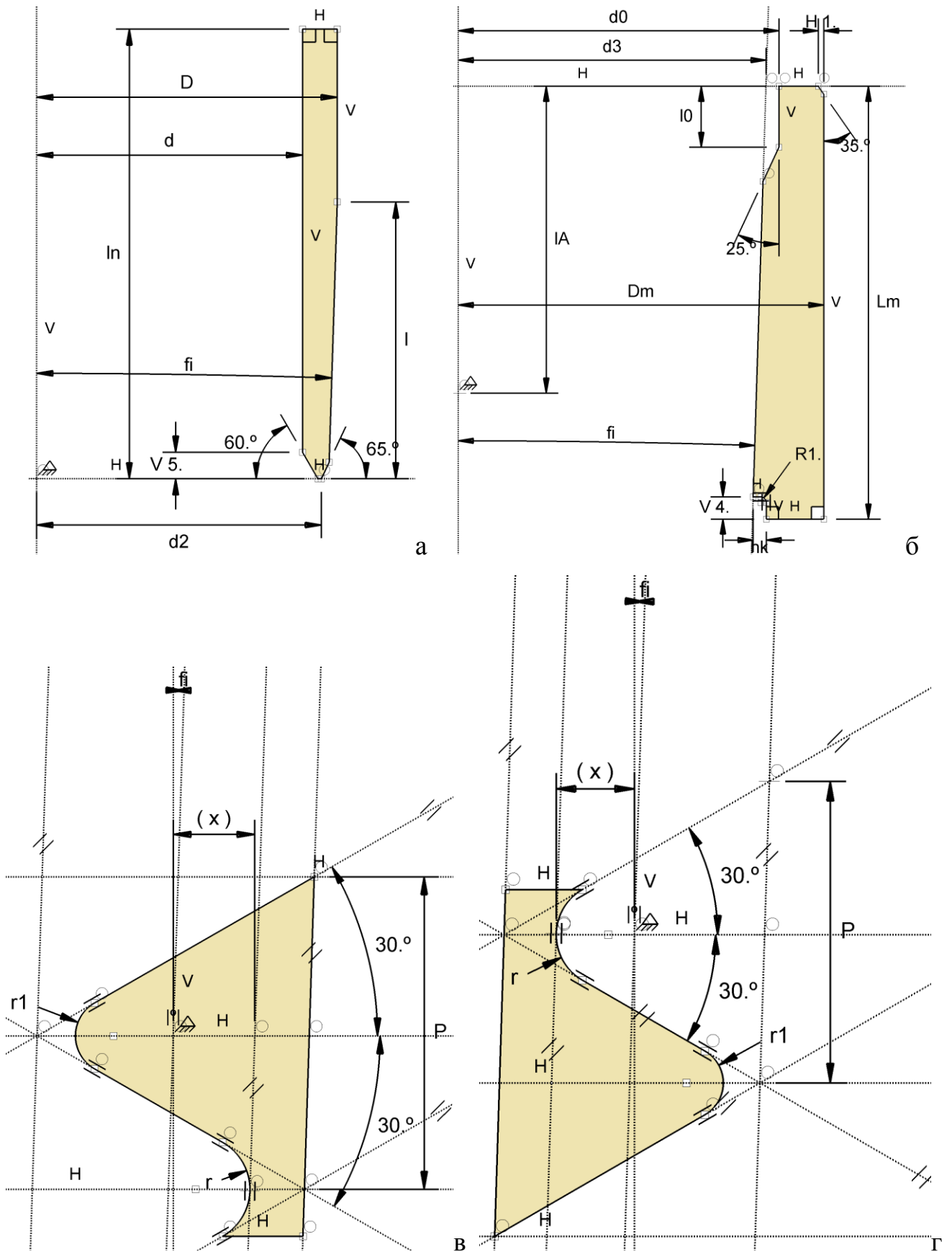
в – для "вирізання" профілю різьби ніпеля, г – для "вирізання" профілю різьби муфти

Рисунок И.5, аркуш 2



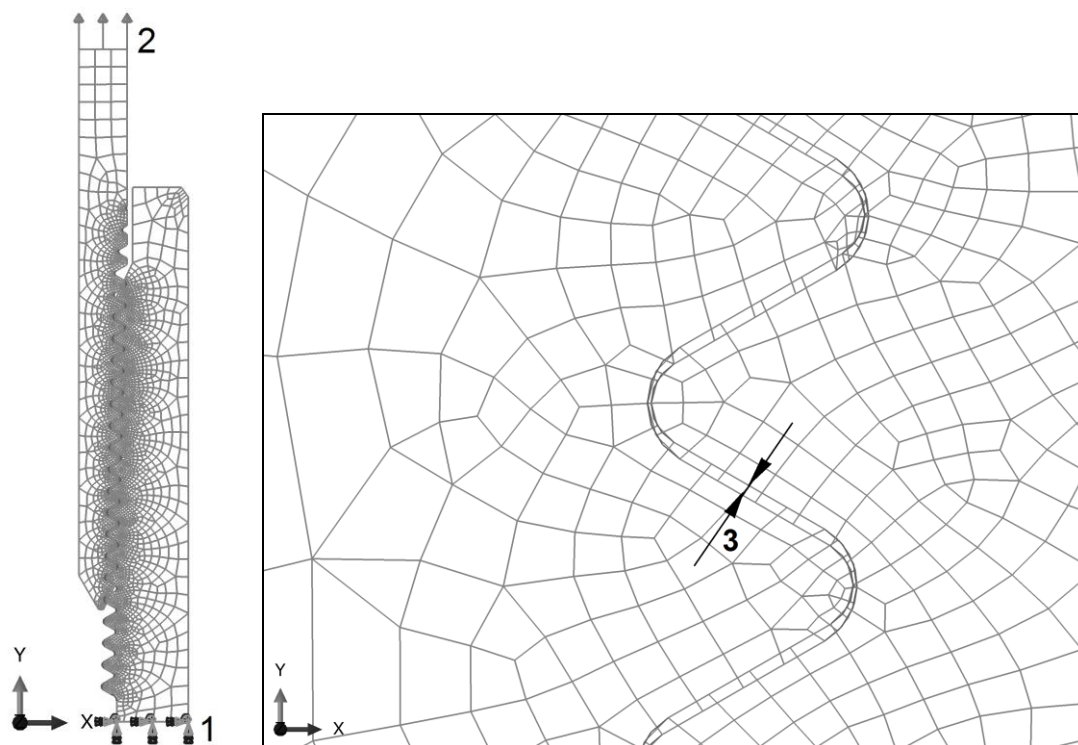
Граничні умови: 1 – (переміщення $U_x=0$, $U_y=0$); 2 – Δ ; 3 – тиск

Рисунок И.6 – Осесиметрична СЕМ замкового РЗ



а – заготовки ніпеля, б – заготовки муфти, в – для "вирізання" профілю різьби
ніпеля, г – для "вирізання" профілю різьби муфти

Рисунок И.7 – Ескізи для побудови деталей РЗ НКТ



Граничні умови: 1 – (переміщення $U_x=0$, $U_y=0$); 2 – тиск, 3 – Interference

Рисунок И.8 – Осесиметрична СЕМ муфтового РЗ НКТ

Макроси для побудови СЕМ РЗ в Abaqus

Вміст пакету:

- tools.py – компоненти для побудови моделі;
- gost13877_96.py – модель муфтового РЗ ШН (ГОСТ 13877-96);
- gost5286_75.py – модель замкового РЗ (ГОСТ 5286-75);
- gost633_80.py – модель муфтового РЗ НКТ (ГОСТ 633-80);
- main.py – розрахунок муфтового РЗ ШН;
- main2.py – розрахунок замкового РЗ;
- main3.py – розрахунок РЗ НКТ.

Лістинг И.1 – tools.py

```
# -*- coding: cp1251 -*-
from math import *
from part import *
from material import *
```

```

from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from job import *
from sketch import *
import regionToolset
from visualization import *
from connectorBehavior import *
import subprocess

if _my_thread_model==2: openMdb(pathName='C:/Temp/zamok.cae')
if _my_thread_model==3: openMdb(pathName='C:/Temp/nkt.cae')
model=mdb.models['Model-1']

class Dim:
    "Клас описує поняття розміру"
    n=0.0 # номінальний розмір
    ei=0.0 # нижнє відхилення
    es=0.0 # верхнє відхилення
    v=0.0 # дійсне значення
    def __init__(self,*x):
        "конструктор, x-кортеж"
        self.n=x[0][0]
        self.ei=x[0][1]
        self.es=x[0][2]
    def min(self):
        "повертає мінімальний розмір"
        return self.n+self.ei
    def max(self):
        "повертає максимальний розмір"
        return self.n+self.es
class Line(object):
    '''Лінія (відрізок або пряма)
    p1,p2 - перша і друга точки
    len - довжина
    angle - кут до 0X в градусах
    '''
    def __init__(self,p1,p2=None,len=None,angle=None):
        self.p1=p1
        x1,y1=self.p1[0],self.p1[1]
        if len==None and p2==None:
            len=1.0
        if angle!=None:
            angle=radians(angle)
        if p2==None:
            self.p2=(x1+len*cos(angle),y1+len*sin(angle))
        else:
            self.p2=p2

```

```

x2,y2=self.p2[0],self.p2[1]
#коефіцієнти рівняння Ax+By+C=0
self.A=y1-y2
self.B=x2-x1
self.C=x1*y2-x2*y1
def points(self):
    '''повертає x1,y1,x2,y2'''
    return self.p1[0],self.p1[1],self.p2[0],self.p2[1]
def angleOX(self):
    '''кут нахилу до ох в радіанах'''
    return atan(-self.A/self.B)
def len(self):
    '''довжина'''
    x1,y1,x2,y2=self.points()
    return sqrt((y2-y1)**2+(x2-x1)**2)
def x(self,y):
    '''координата x точки лінії за координатою y'''
    x1,y1,x2,y2=self.points()
    return (y-y1)*(x2-x1)/(y2-y1)+x1#x
def y(self,x):
    '''координата y точки лінії за координатою x'''
    x1,y1,x2,y2=self.points()
    return (x-x1)*(y2-y1)/(x2-x1)+y1#y
def mpoint(self):
    '''середня точка'''
    x1,y1,x2,y2=self.points()
    return ((x2-x1)/2+x1,(y2-y1)/2+y1)
def dist(self,point):
    '''відстань від лінії до точки point'''
    x,y=point[0],point[1]
    return abs((self.A*x+self.B*y+self.C)/sqrt(self.A**2+self.B**2))
def cros_point(self,line):
    '''точка перетину з лінією line'''
    x=(self.B*line.C-line.B*self.C)/(self.A*line.B-line.A*self.B)
    y=(self.C*line.A-line.C*self.A)/(self.A*line.B-line.A*self.B)
    return (x,y)
def drawAbaqus(self,sketch):
    '''рисувати в Abaqus'''
    g=sketch.Line(point1=self.p1, point2=self.p2)
    return g

```

```
class N_angle(object):
```

```
    '''N-кутник зі скругленнями
```

```
    L - список ліній (по порядку)
```

```
    R - список радіусів скруглень (по порядку).
```

```
    Наприклад, R[0] - скруглення між L[0] і L[1]
```

```
    Приклад:
```

```
    L1=Line(p1=(1.0,1.0),len=2.0,angle=170)
```

```
    cp=Line(p1=L1.p1,angle=90+30).cros_point(Line(p1=L1.p2,angle=90-30))
```

```
    L2=Line(p1=L1.p1,p2=cp)
```

```
    L3=Line(p1=L1.p2,p2=cp)
```

```

T1=N_angle([L1,L2,L3],[0.05,0.05,0.05])
'''
def __init__(self,L,R):
    self.N=len(L) # кількість ліній (кутів)
    L.append(L[0]) # додати в список ліній першу: [L1,L2,L3,L1]
    self.V=range(self.N) # список вершин
    for i in range(self.N):
        self.V[i]=self.Vert(L[i],L[i+1],R[i])
def Vert(self,La,Lb,R):
    '''повертає вершину в форматі:
    (точка, перша лінія, друга лінія, радіус скруглення).
    Якщо перша точка першої лінії співпадає з першою
    або другою точкою другої лінії, то вона є вершиною'''
    if La.p1==Lb.p1 or La.p1==Lb.p2:
        V=(La.p1,La,Lb,R)
    else: V=(Lb.p2,La,Lb,R)
    return V
def drawAbaqus(self,sketch):
    '''рисувати в Abaqus'''
    g=range(self.N)
    for i in range(self.N): # рисувати лінії
        g[i]=self.V[i][1].drawAbaqus(sketch)
    g.append(g[0]) # додати в список ліній першу: [g1,g2,g3,g1]
    for i in range(self.N): # рисувати скруглення
        if self.V[i][3]!=0: # якщо радіус скруглення не 0
            sketch.FilletByRadius(curve1=g[i], curve2=g[i+1],
nearPoint1=self.V[i][1].mpoint(), nearPoint2=self.V[i][2].mpoint(),
radius=self.V[i][3])

class Material:
    '''Клас описує поняття матеріалу
    В Abaqus задається істинна діаграма деформування (див.Stress and strain
measures)
    E - модуль пружності, Па
    mu - коефіцієнт Пуассона
    st - границя текучості, Па
    et - деформація для st
    sb - істинна границя міцності, Па (sv - умовна границя)
    eb - істинна деформація, яка відповідає границі міцності
    delta - відносне видовження
    psi - відносне звуження
    '''
    def __init__(self,E,mu,st,sv,delta,psi):
        '''конструктор'''
        self.E=E # модуль пружності
        self.mu=mu # коефіцієнт Пуассона
        self.st=st # границя текучості
        self.et=st/E # деформація для st
        self.delta=delta/100.0 # відносне видовження після розриву
        self.psi=psi/100.0 # відносне звуження після розриву
        k=0.4 # коефіцієнт(eb=(0.1...0.4,0.2...0.8)delta)

```

```

self.sv=sv # границя міцності
self.sb=sv*(1+k*self.delta) # істинна границя міцності
self.eb=log(1+k*self.delta) # істинна деформація, яка відповідає
границі міцності
# істинне напруження і деформація в момент руйнування
self.sk=0.8*self.sv/(1-self.psi) # 0.8-коефіцієнт руйнуючого
навантаження
#self.sk=self.sv*(1+1.35*self.psi)
self.ek=log(1/(1-self.psi))
def bilinear(self):
    '''Повертає словник еластичних і пластичних властивостей'''
    return {'el':((self.E,self.mu),),
            'pl':((self.st,0.0), # білінійна залежність
                  (self.sb,self.eb))} # або (self.sk,self.ek)
def e(self,s,n):
    '''Степенева залежність e(s)
    n - степінь
    '''
    return self.et*(s/self.st)**n
def power(self,k):
    '''Повертає словник еластичних і пластичних властивостей
    k - кількість ліній для апроксимації пластичної ділянки (2,4,8...)
    '''
    # n визначається з умови проходження через точку (eb+et,sb),
n=6...10
    n=log((self.eb+self.et)/self.et)/log(self.sb/self.st)
    ds=self.sb-self.st
    # степенева залежність
    k=float(k)
    s_e=[(self.st+i/k_*ds,self.e(self.st+i/k_*ds,n)-self.et) for i in
range(0,k+1)]
    #s_e=[(self.st+i*ds,self.e(self.st+i*ds,n)-self.et) for i in
[0.0,0.25,0.5,0.75,1.0]]
    s_e.append((self.sk,self.ek)) # додати точку руйнування
    return {'el':((self.E,self.mu),),
            'pl':tuple(s_e)}

matlib={
'40':Material(E=210000.0e+6,mu=0.28,st=314.0e+6,sv=559.0e+6,delta=16.0,psi=
45.0),
'40fesafe':Material(E=200000.0e+6,mu=0.33,st=314.0e+6,sv=559.0e+6,delta=16.
0,psi=45.0),
'20H2M':Material(E=210000.0e+6,mu=0.28,st=382.0e+6,sv=588.0e+6,delta=21.0,p
si=56.0),
'30ХМА':Material(E=210000.0e+6,mu=0.28,st=392.0e+6,sv=598.0e+6,delta=20.0,p
si=62.0),
'15H3МА':Material(E=210000.0e+6,mu=0.28,st=490.0e+6,sv=637.0e+6,delta=22.0,
psi=60.0),
'15Х2НМФ':Material(E=210000.0e+6,mu=0.28,st=617.0e+6,sv=686.0e+6,delta=16.0
,psi=63.0),

```



```
'15X2ГМФ':Material(E=210000.0e+6,mu=0.28,st=617.0e+6,sv=686.0e+6,delta=16.0
,psi=63.0),
'14X3ГМЮ':Material(E=210000.0e+6,mu=0.28,st=617.0e+6,sv=725.0e+6,delta=16.0
,psi=63.0),
}
steel20={'el':((210000000000.0, 0.28), ),
         'pl':((620000000.0, 0.0),
              (640000000.0, 0.02),
              (800000000.0, 0.04),
              (860000000.0, 0.08),
              (864000000.0, 0.11))}
steel45={'el':((210000000000.0, 0.28), ),
         'pl':((620000000.0, 0.0),
              (640000000.0, 0.02),
              (800000000.0, 0.04),
              (860000000.0, 0.08),
              (864000000.0, 0.11))}
```

```
def delCutExtrude():
    "Знищує усі елементи, назва яких починається з 'Cut extrude' і
    'Partition face-1'"
    if model.parts['Part-2'].features.has_key('Partition face-1'):
        del model.parts['Part-2'].features['Partition face-1'] # знищує
    Partition face
    for f in model.parts['Part-1'].features.values():
        if f.name[:11]=='Cut extrude': model.parts['Part-
1'].deleteFeatures((f.name,))
    for f in model.parts['Part-2'].features.values():
        if f.name[:11]=='Cut extrude': model.parts['Part-
2'].deleteFeatures((f.name,))
def set_values(sketch,p):
    '''
    Присвоює значення параметрам ескізу.
    Приклад:
    par={'aint':0,'aext':0,'rint':0,'Rext':dn.v,'len':l1n.v+20}
    set_values(sketch='nipple',p=par)
    '''
    s=model.sketches[sketch]
    for k,v in p.iteritems():
        s.parameters[k].setValues(expression=str(v))
def set_values2(sketch,base,p):
    '''
    Присвоює значення параметрам ескізу.
    Приклад:
    par={'aint':0,'aext':0,'rint':0,'Rext':dn.v,'len':l1n.v+20}
    set_values2(sketch='nipple',base='Quad_h',p=par)
    '''
    s=model.ConstrainedSketch(name=sketch, objectToCopy=mdb.models['Model-
1'].sketches[base])
    for k,v in p.iteritems():
        s.parameters[k].setValues(expression=str(v))
```

```

def part_builder(part,sketch='Sketch-1',vector=(0.0,0.0),oper='shell'):
    '''Додає до деталі виріз або поверхню задану ескізом
    Приклад:
    part_builder(p, 'groove', (0,0), 'cut')
    '''
    s=model.ConstrainedSketch(name='__profile__',sheetSize=200.)
    part.projectReferencesOntoSketch(filter=COPLANAR_EDGES, sketch=s)
    s.ConstructionLine(point1=(0.0,0.0), point2=(0.0, 10.0))
    s.retrieveSketch(sketch=model.sketches[sketch])
    s.move(objectList=s.geometry.values(),vector=vector)
    if oper=='shell': part.Shell(sketch=s)
    if oper=='cut': part.Cut(sketch=s)
    del s
def createPart(n,s):
    '''Створює деталь
    n - ім'я
    s - ескіз'''
    model.Part(dimensionality=AXISYMMETRIC, name=n, type=DEFORMABLE_BODY)
    model.parts[n].BaseShell(sketch=model.sketches[s])
def createPart3D(frompart,sketch,part):
    '''Створює 3D деталь на основі осесиметричної деталі
    Створюється ескіз sketch як проекція осесиметричної деталі'''
    s=model.ConstrainedSketch(name=sketch,sheetSize=200.)
    tmp=model.ConstrainedSketch(name='__profile__',sheetSize=200.)
    #s.sketchOptions.setValues(viewStyle=AXISYM)

model.parts[frompart].projectReferencesOntoSketch(filter=COPLANAR_EDGES,
sketch=tmp)
ln=tmp.ConstructionLine(point1=(0.0,0.0), point2=(0.0, 10.0))
tmp.FixedConstraint(entity=ln)
tmp.assignCenterline(line=ln)
s.retrieveSketch(sketch=tmp)
#s.sketchOptions.setValues(constructionGeometry=ON)
p=model.Part(dimensionality=THREE_D, name=part, type=DEFORMABLE_BODY)
p.BaseSolidRevolve(angle=360.0, flipRevolveDirection=OFF, sketch=s)
def createCut(Part,Sketch,Begin,P,Fi,Len,X,Y,dx,dy):
    '''Створює частину профілю різьби
    Part - деталь (рядок)
    Sketch - ескіз (рядок)
    Begin - початок різьби (ціле)
    P - крок різьби (дійсне)
    Fi - кут конуса конічної різьби (градуси)
    Len - довжина різьби (дійсне)
    X,Y - початкові координати центра профілю
    dx - радіальний напрямок подачі (+1 - вправо, -1 - вліво)
    dy - осьовий напрямок подачі (+1 - вверх, -1 - вниз)
    '''
    # можна це зробити також за допомогою LinearInstancePattern
    i=Begin # номер витка (0-перший)
    while i*P<=Len: # довжина різьби
        s=model.ConstrainedSketch(name='__profile__',sheetSize=200.)

```

```

model.parts[Part].projectReferencesOntoSketch(filter=COPLANAR_EDGES,
sketch=s)
    s.ConstructionLine(point1=(0.0,0.0), point2=(0.0, 10.0))
    s.retrieveSketch(sketch=model.sketches[Sketch])
    s.delete(objectList=(s.vertices.findAt((0.0,0.0),), ))

s.move(objectList=s.geometry.values(),vector=(X+dx*P*tan(Fi*pi/180)*i,Y+dy*
P*i))
    model.parts[Part].Cut(sketch=s)
    del s
    i=i+1
    return i-1
def createPartition(part,offset):
    '''Ділить поверхню деталі лінією (0.0,offset,0.0),
(2000.0,offset,0.0)'''
    model.parts[part].PartitionFaceByShortestPath(faces=
        model.parts[part].faces[0], point1=(0.0,offset,0.0),
point2=(2000.0,offset,0.0))
def createPartition3D(part,offset):
    '''Ділить об'єм деталі площиною зміщеною від XZPLANE на відстань
offset'''
    p = model.parts[part]
    p.DatumPlaneByPrincipalPlane(principalPlane=XZPLANE, offset=offset)
    dp = p.datums.values()[-1]
    p.PartitionCellByDatumPlane(datumPlane=dp, cells=p.cells[0])
def createMaterial(n,et,pt,kinematic=False):
    '''Створює матеріал
n - ім'я
et - пружні характеристики
pt - пластичні характеристики
kinematic - модель зміцнення (True - кінематичне, False - ізотропне)
'''
    m=model.Material(name=n)
    m.Elastic(table=et)
    if kinematic:
        m.Plastic(hardening=KINEMATIC, table=pt)# pt - тільки дві точки !
    else:
        m.Plastic(table=pt)

def createSectionAssign(n,m,p):
    '''Створює і присвоює секції деталі
n - ім'я
m - матеріал
p - деталь
'''
    model.HomogeneousSolidSection(material=m, name=n, thickness=None)
    model.parts[p].SectionAssignment(region=Region(
        faces=model.parts[p].faces), sectionName=n)
def createSectionAssign3D(n,m,p):
    '''Створює і присвоює секції деталі
'''

```

```

n - ім'я
m - матеріал
p - деталь
...

if model.parts[p].sectionAssignments:
    del model.parts[p].sectionAssignments[0]
model.HomogeneousSolidSection(material=m, name=n, thickness=None)

model.parts[p].SectionAssignment(region=Region(cells=model.parts[p].cells),
sectionName=n)

def createAssemblyInstance(n,p):
    '''Створює елемент збірки
n - ім'я
p - деталь
...

    model.rootAssembly.Instance(dependent=OFF, name=n, part=model.parts[p])
def createAssemblyInstance3D(n,p):
    '''Створює елемент збірки
n - ім'я
p - деталь
...

    #model.rootAssembly.DatumCsysByDefault(CARTESIAN)
    model.rootAssembly.Instance(dependent=OFF, name=n, part=model.parts[p])
def createStep(n,pr):
    '''Створює крок
n - ім'я
pr - попередній крок
...

    model.StaticStep(name=n, previous=pr)
def createContactSet(n,i,ep):
    '''Створює набір для контакту
n - ім'я
i - елемент зборки
ep - кортеж точок кромки не для контакту
...

    model.rootAssembly.regenerate()
    ae=model.rootAssembly.instances[i].edges
    e=ae.findAt(*ep) # *ep - розпакування кортежу
    p=[x.pointOn for x in ae if x not in e]
    model.rootAssembly.Set(name=n,edges=ae.findAt(*p))
def createContactSet3D(n,i,ep):
    '''Створює набір для контакту
n - ім'я
i - елемент збірки
ep - кортеж точок кромки не для контакту
...

    model.rootAssembly.regenerate()
    ae=model.rootAssembly.instances[i].faces
    e=ae.findAt(*ep) # *ep - розпакування кортежу
    p=[x.pointOn for x in ae if x not in e]

```

```

    model.rootAssembly.Set(name=n, faces=ae.findAt(*p))
def createContactProperty():
    '''Створює властивості контакту'''
    model.ContactProperty('IntProp-1')
    model.interactionProperties['IntProp-1'].TangentialBehavior(
        dependencies=0, directionality=ISOTROPIC,
elasticSlipStiffness=None,
        formulation=PENALTY, fraction=0.005, maximumElasticSlip=FRACTION,
        pressureDependency=OFF, shearStressLimit=None,
slipRateDependency=OFF,
        table=((0.05, ), ), temperatureDependency=OFF)
    model.interactionProperties['IntProp-1'].NormalBehavior(
        allowSeparation=ON,
constraintEnforcementMethod=DEFAULT, pressureOverclosure=HARD)
def createContact():
    '''Створює контакт'''
    sm=model.rootAssembly.sets['Master']
    ss=model.rootAssembly.sets['Slave']
    model.SurfaceToSurfaceContactStd(adjustMethod=None,
        clearanceRegion=None, createStepName='Step-1', datumAxis=None,
enforcement=
        SURFACE_TO_SURFACE, initialClearance=OMIT,
interactionProperty='IntProp-1',
        interferenceType=SHRINK_FIT, master=Region(side1Edges=sm.edges),
name='Int-1',
        slave=Region(side1Edges=ss.edges), sliding=SMALL,
surfaceSmoothing=None,
        thickness=ON)
def createContact3D():
    '''Створює контакт'''
    sm=model.rootAssembly.sets['Master']
    ss=model.rootAssembly.sets['Slave']
    model.SurfaceToSurfaceContactStd(adjustMethod=None,
        clearanceRegion=None, createStepName='Step-1', datumAxis=None,
        initialClearance=OMIT, interactionProperty='IntProp-1',
master=Region(
        side1Faces=sm.faces), name='Int-1', slave=Region(
        side1Faces=ss.faces), sliding=FINITE, smooth=0.2)
def createBCSet(n,i,ep):
    '''Створює набір для граничної умови
n - ім'я
i - елемент зборки
ep - кортеж точок кромки для граничної умови
'''
s=model.rootAssembly.Set(edges=model.rootAssembly.instances[i].edges.findAt
(ep), name=n)
def createBCSet3D(n,i,ep):
    '''Створює набір для граничної умови
n - ім'я
i - елемент зборки
'''

```

ep - кортеж точок кромок для граничної умови
 ...

```
s=model.rootAssembly.Set(faces=model.rootAssembly.instances[i].faces.findAt
(ep), name=n)
```

```
def createBC_Pressure(step):
    '''Створює тиск. Приклад:
    createBC_Pressure([('Step-1',-1.0),('Step-2',-
276.0e+6*d0.v**2/dn.v**2)])'''
    s=model.rootAssembly.sets['Pressure']
    model.Pressure(amplitude=UNSET, createStepName=step[0][0],
        distributionType=UNIFORM, field='', magnitude=step[0][1],
name='Pressure',
        region=Region(side1Edges=s.edges))
    for x in step:
        model.loads['Pressure'].setValuesInStep(magnitude=x[1],
stepName=x[0])
def createBC_Axis():
    '''Створює граничні умови на осі (для осесиметричних моделей)'''
    s=model.rootAssembly.sets['Axis']
    model.DisplacementBC(amplitude=UNSET, createStepName='Step-1',
distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None,
name=
'Axis', region=Region(edges=s.edges), u1=0.0, u2=UNSET, ur3=0.0)
def createBC_Encastre():
    '''Створює закріплення'''
    s=model.rootAssembly.sets['Encastre']
    model.EncastreBC(createStepName='Step-1', name='Encastre',
region=Region(edges=s.edges))
def createBC_BoltLoad(part,point,value):
    '''Створює BoltLoad. Приклад:
    createBC_BoltLoad('Part-2-1',em3,-0.1)'''
    model.BoltLoad(boltMethod=ADJUST_LENGTH, createStepName='Step-1',
datumAxis=
        model.rootAssembly.instances[part].datums[1],
        magnitude=value, name='BoltLoad', region=Region(
            side1Edges=model.rootAssembly.instances[part].edges.findAt((point,
))))))
def createBC_Pressure3D(step):
    '''Створює тиск. Приклад:
    createBC_Pressure([('Step-1',-1.0),('Step-2',-
276.0e+6*d0.v**2/dn.v**2)])'''
    s=model.rootAssembly.sets['Pressure']
    model.Pressure(amplitude=UNSET, createStepName=step[0][0],
        distributionType=UNIFORM, field='', magnitude=step[0][1],
name='Pressure',
        region=Region(side1Faces=s.faces))
    for x in step:
        model.loads['Pressure'].setValuesInStep(magnitude=x[1],
stepName=x[0])
```

```

def createBC_Encastre3D():
    '''Створює закріплення'''
    s=model.rootAssembly.sets['Encastre']
    model.EncastreBC(createStepName='Step-1', name='Encastre',
region=Region(faces=s.faces))
def createBC_BoltLoad3D(part,point,value):
    '''Створює BoltLoad. Приклад:
createBC_BoltLoad('Part-2-1',em3,-0.1)'''
    model.BoltLoad(boltMethod=ADJUST_LENGTH, createStepName='Step-1',
datumAxis=
        model.rootAssembly.instances[part].datums[1],
        magnitude=value, name='BoltLoad', region=Region(
            side1Faces=model.rootAssembly.instances[part].faces.findAt((point,
))))))
def createMesh():
    '''Створює сітку'''
    model.rootAssembly.seedPartInstance(deviationFactor=0.1,
        regions=(model.rootAssembly.instances['Part-1-1'],
            model.rootAssembly.instances['Part-2-1']), size=2.6)
    sm=model.rootAssembly.sets['Master']
    ss=model.rootAssembly.sets['Slave']
    model.rootAssembly.seedEdgeByNumber(edges=sm.edges, number=4)
    model.rootAssembly.seedEdgeByNumber(edges=ss.edges, number=4)
    model.rootAssembly.generateMesh(regions=(
        model.rootAssembly.instances['Part-1-1'],
        model.rootAssembly.instances['Part-2-1']))
def createMesh3D():
    '''Створює сітку'''
    model.rootAssembly.setMeshControls(elemShape=TET, regions=
        model.rootAssembly.instances['Part-3-1'].cells+\
        model.rootAssembly.instances['Part-4-1'].cells, technique=FREE)
    model.rootAssembly.seedPartInstance(deviationFactor=0.1,
        regions=(model.rootAssembly.instances['Part-3-1'],
            model.rootAssembly.instances['Part-4-1']), size=2.8)
    model.rootAssembly.generateMesh(regions=(
        model.rootAssembly.instances['Part-3-1'],
        model.rootAssembly.instances['Part-4-1']))
def createEdgesSet(n, i, p, exclude=False):
    '''Створює Set з ребер (ребро визначається довільною точкою на ньому)
exclude=True - з усіх ребер крім заданих (тільки для OdbSet!)
createEdgesSet(n='Set-6',i='Part-1-1',p=((enr1, ),(en1, )))'''
    if exclude==True:
        edges=model.rootAssembly.instances[i].edges[:]
        xEdges=model.rootAssembly.instances[i].edges.findAt(*p)
        model.rootAssembly.Set(edges=edges, xEdges=xEdges, name=n)
    else:
        edges=model.rootAssembly.instances[i].edges.findAt(*p)
        model.rootAssembly.Set(edges=edges, name=n)
def createVerticesSet(n, i, p):
    '''Створює Set з вершин
createVerticesSet(n='Set-7',i='Part-1-1',p=(((0,0,0), ),))'''

```

```

model.rootAssembly.Set(vertices=model.rootAssembly.instances[i].vertices.findAt(*p), name=n)
def createSet(n, r):
    '''Створює Set з Region
    r = regionToolset.Region(edges=e,vertices=v,xEdges=xe,xVertices=xv)
    createSet(n='Set-8', r=r)'''
    model.rootAssembly.Set(region=r, name=n)
def delItems():
    '''Знищує елементи'''
    if model.steps.has_key('Step-2'): del model.steps['Step-2']
    if model.steps.has_key('Step-1'): del model.steps['Step-1']
def createJobSubmit():
    '''Створює задачу і виконує її'''
    myJob = mdb.Job(name=model.name, model=model.name)
    myJob.submit()
    # Чекає поки задача не буде розв'язана
    myJob.waitForCompletion()
def createObdNodeSet(coords,name='MYSET',prt='Part-1-1',item_type='vertex'):
    '''Створює ObdNodeSet за заданим координатами ребром або вершиною'''
    if item_type=='edge':
        _item=model.rootAssembly.instances[prt].edges.findAt(coordinates=coords)
        #або getClosest()
        if item_type=='vertex':
            _item=model.rootAssembly.instances[prt].vertices.findAt(coordinates=coords)
            _nodes=_item.getNodes() #model.rootAssembly.sets['Set-6'].nodes
            _nodeLabels=[]
            for x in _nodes:
                _nodeLabels.append(x.label)
        _nset=myOdb.rootAssembly.NodeSetFromNodeLabels(name=name,nodeLabels=((prt.upper(), _nodeLabels),))
def readODB_path(path,step,var,intersections=False):
    '''Читає результати з останнього фрейму кроку на заданому шляху з точок
    path - шлях (список точок, >=1)
    step - крок
    var - змінна:
    (('S', INTEGRATION_POINT, ((INVARIANT, 'Mises'), )), )
    (('S', INTEGRATION_POINT, ((INVARIANT, 'Pressure'), )), )
    (('S', INTEGRATION_POINT, ((COMPONENT, 'S11'), )), )
    (('U', NODAL, ((INVARIANT, 'Magnitude'), )), )
    (('U', NODAL, ((COMPONENT, 'U1'), )), )
    (('CPRESS', ELEMENT_NODAL), )
    'D' #коефіцієнт запасу втомної міцності
    intersections=True - додає проміжні точки
    Приклад: readODB_path(path=((1.97212e+001, 3.65000e+001, 0.0),
),step='Step-1',var=var)
    '''

```



```

    pth=session.Path(name='Path-tmp', type=POINT_LIST, expression=path)
#шлях
    if var=='D':
        frame1 = session.scratchOdb['Model-1.odb'].steps['Session
Step'].frames[0]
        session.viewports['Viewport: 1'].odbDisplay.setFrame(frame=frame1)
        dat=session.XYDataFromPath(name='XYData-1', path=pth,
includeIntersections=intersections, shape=UNDEFORMED, labelType=SEQ_ID)
    else:
        st=myOdb.steps[step].number-1 # індекс кроку
        fr=len(myOdb.steps[step].frames)-1 # індекс останнього фрейму
        #session.viewports['Viewport: 1'].odbDisplay.setFrame(step=st,
frame=fr)
        #session.viewports['Viewport:
1'].odbDisplay.setPrimaryVariable(variableLabel='S',
outputPosition=INTEGRATION_POINT, refinement=(INVARIANT, 'Tresca'))
        dat=session.XYDataFromPath(name='XYData-1', path=pth,
includeIntersections=intersections, shape=UNDEFORMED,
labelType=SEQ_ID, step=st, frame=fr, variable=var) # дані
        res=[] # список результатів
        for x in dat.data:
            res.append(x[1]) # додати до списку результатів
# видалити тимчасові дані
        del session.paths['Path-tmp']
        for k in session.xyDataObjects.keys():
            del session.xyDataObjects[k]
        return res # повертає список значень

def readODB_set(set, step, var, pos=NODAL):
    '''Читає результати з останнього фрейму кроку на заданій множині
set - множина
step - крок
var - змінна:
(('S', INTEGRATION_POINT, ((INVARIANT, 'Mises'), )), )
(('CPRESS', ELEMENT_NODAL), )
pos - позиція: NODAL - для вузлів, INTEGRATION_POINT - для елементів
Приклад: readODB_set(set='Cont', step='Step-1', var=var)
'''
    if pos==NODAL:
        dat=session.xyDataListFromField(odb=myOdb, outputPosition=NODAL,
variable=var, nodeSets=(set.upper(),)) # дані
        if pos==INTEGRATION_POINT:
            dat=session.xyDataListFromField(odb=myOdb,
outputPosition=INTEGRATION_POINT, variable=var, elementSets=(set.upper(),))
#дані
        res=[] # список результатів
        for x in dat: # для всіх вузлів
            n=0
            for k in myOdb.steps.keys(): # для всіх кроків
                n=n+len(myOdb.steps[k].frames) # сумарна кількість фреймів до k
кроку включно

```

```

        if k==step: res.append(x.data[n-1][1]) # додати до списку
результатів
        # data - це ((час,значення),(час,значення)...)
# видалити тимчасові дані
for k in session.xyDataObjects.keys():
    del session.xyDataObjects[k]
return res # повертає список значень

def readODB_set_(set,var):
'''Повертає список результатів в вузлах заданої множини
(для змінних fe-safe)
set - множина
var - змінна:
('LOGLife-Repeats', ELEMENT_NODAL), )
('FOS@Life=Infinite', ELEMENT_NODAL), )
('%Failure@Life=5E6-Repeats', ELEMENT_NODAL), )
Приклад: readODB_set_(set='Set-1',var=var)
'''

# отримати дані
dat=session.xyDataListFromField(odb=myOdb, outputPosition=NODAL,
variable=var, nodeSets=(set.upper(),)) # дані

res=[] # список результатів
for x in dat: # для всіх вузлів
    # x.data - це ((час,значення),(час,значення)...)
    res.append(x.data) # дані

# видалити тимчасові дані
for k in session.xyDataObjects.keys():
    del session.xyDataObjects[k]
return res # повертає список значень

def readODB_set2(set,step,var,pos=NODAL):
'''Читає результати з останнього фрейму кроку на заданій множині
(менш універсальна альтернатива readODB_set())
set - множина
step - крок
var - змінна:
('S','Mises')
('S','Pressure')
('U','Magnitude')
('U','U1')
('CPRESS','')
('D','') # коефіцієнт запасу втомної міцності
pos - позиція: NODAL - для вузлів,INTEGRATION_POINT - для елементів
Приклад: readODB_set2(set='Cont',step='Step-1',var=('S','Mises'))
'''

if pos==NODAL:
    s=myOdb.rootAssembly.nodeSets[set.upper()] # множина вузлів
if pos==INTEGRATION_POINT:
    s=myOdb.rootAssembly.elementSets[set.upper()] # множина елементів

```

```

    if var[0]=='D':
        fo=session.scratchOdb['Model-1.odb'].steps['Session
Step'].frames[-1].fieldOutputs['D'].getSubset(region=s,position=pos) # дані
    else:
        fo=myOdb.steps[step].frames[-
1].fieldOutputs[var[0]].getSubset(region=s,position=pos) # дані
        #openOdb(r'C:/Temp/Model-1.odb').steps['Step-
1'].frames[4].fieldOutputs['CPRESS'].getSubset(position=NODAL,
region=openOdb(r'C:/Temp/Model-
1.odb').rootAssembly.nodeSets['CONT']).values[0].data
        res=[] # список результатів
        for v in fo.values: # для кожного вузла/елемента
            if var[1]=='Mises': res.append(v.mises) # додати до списку
результатів
            if var[1]=='Pressure': res.append(v.press)
            if var[0]=='U' and var[1]=='Magnitude': res.append(v.magnitude)
            if var[1]=='U1': res.append(v.data.tolist()[0])
            if var[1]=='U2': res.append(v.data.tolist()[1])
            if var[0]=='CPRESS': res.append(v.data)
            if var[0]=='D': res.append(v.data)
        return res # повертає список значень

def runFeSafe(input_odb,input_stlx,output_odb):
    '''Виконує аналіз втомної міцності у fe-safe
input_odb - назва вхідного файлу результатів Abaqus (без розширення
.odb)
input_stlx - назва файлу моделі fe-safe (без розширення .stlx)
output_odb - назва вихідного файлу результатів Abaqus (без розширення
.odb)
Якщо з FEA моделі імпортуються напруження і деформації, то перед
створенням файлу sltx в налаштуваннях Analysis Options необхідно вибрати
Read strains from FE Models.
'''
    s=r'd:\Program Files\Safe_Technology\fe-safe\version.6.2\exe\fe-
safe_cl.exe -s j=c:\1\{iodb}.odb b=c:\1\{istlx}.stlx o=c:\1\{oodb}.odb'
    s=s.format(iodb=input_odb, istlx=input_stlx, oodb=output_odb)
    # виконує обчислення в fe-safe та чекає завершення
    subprocess.Popen(s).communicate()

def writeLDFfile(filename,lst):
    '''Змінює вміст файлу визначення навантаження LDF fe-safe
filename - імя файлу
lst - список рядків-даних
'''
    f=open(filename, "w")
    s=""
# .ldf file created by fe-safe compliant product 6.2-01[mswin]

INIT
transitions=Yes
END

```

```

# Block number 1
BLOCK n=1, scale=1
lh=0 1 , ds=1, scale=1
lh=0 {x} , ds=2, scale=1
END

""" # шаблон файлу
    s=s.format(x=lst[0]) # вміст файлу з шаблону
    f.write(s)
    f.close()

def SF_field(s1='Step-1',s2='Step-2',Sn=207000000,m=1):
    '''Розраховує поле коефіцієнта запасу втомної міцності за критерієм
    Сайнса'''

    S3_2 = myOdb.steps[s2].frames[-
1].fieldOutputs['S'].getScalarField(invariant=MAX_PRINCIPAL)
    S3_1 = myOdb.steps[s1].frames[-
1].fieldOutputs['S'].getScalarField(invariant=MAX_PRINCIPAL)
    Sm3=(S3_2+S3_1)/2
    Sa3=(S3_2-S3_1)/2

    S2_2 = myOdb.steps[s2].frames[-
1].fieldOutputs['S'].getScalarField(invariant=MID_PRINCIPAL)
    S2_1 = myOdb.steps[s1].frames[-
1].fieldOutputs['S'].getScalarField(invariant=MID_PRINCIPAL)
    Sm2=(S2_2+S2_1)/2
    Sa2=(S2_2-S2_1)/2

    S1_2 = myOdb.steps[s2].frames[-
1].fieldOutputs['S'].getScalarField(invariant=MIN_PRINCIPAL)
    S1_1 = myOdb.steps[s1].frames[-
1].fieldOutputs['S'].getScalarField(invariant=MIN_PRINCIPAL)
    Sm1=(S1_2+S1_1)/2
    Sa1=(S1_2-S1_1)/2

    D =(Sn-m*(Sm1+Sm2+Sm3)/3)/power(((power((Sa1-Sa2), 2)+power((Sa2-Sa3),
2)+power((Sa3-Sa1), 2))/2), 0.5)

    scratchOdb = session.ScratchOdb(oddb=myOdb)
    sessionStep = scratchOdb.Step(name='Session Step',description='Step for
Viewer non-persistent fields',domain=TIME,timePeriod=1.0)
    sessionFrame = sessionStep.Frame(frameId=0, frameValue=0.0,
description='Session Frame')
    sessionField = sessionFrame.FieldOutput(name='D', description='D',
field=D)
def saveDB(name):
    import shutil,os,shelve
    if not os.path.exists('MyDB'):
        os.mkdir('MyDB')

```

```

shutil.copyfile(model.name + '.odb', 'MyDB/'+name+'.odb') # копіювати
файл

my_db = shelve.open("MyDB/mydb") # відкрити файл полиці
my_db[name]=[mat1,mat2,d,d_n] # записати у полицю під ключем name
my_db.close() # закрити файл полиці
def readDB(name):
import shelve
my_db = shelve.open("MyDB/mydb") # відкрити файл полиці
print my_db.keys() # вивести список усіх ключів
if my_db.has_key(name): # якщо є ключ name
    print my_db[name]
my_db.close() # закрити файл полиці

```

Лістинг И.2 – gost13877_96.py

```

# -*- coding: cp1251 -*-
if _my_thread_model!=1: from tools import *

rod19={'d_n':(27, -0.48, -0.376),#
'd2_n':(25.35, -0.204, -0.047),#
'd1_n':(24.25, 0, -0.415),
'r_n':(0.28, 0, 0.08),
'dn':(38.1, -0.25, 0.13),
'd1n':(23.24, -0.13, 0.13),
'l1n':(36.5, 0, 1.6),
'l2n':(15, 0.2, 1),
'l3n':(32, 0, 1.5),
'l4n':(48, -1, 1.5),
'r3n':(3, 0, 0.8),
'd_m':(27, 0, 0.27),
'd2_m':(25.35, 0, 0.202),
'd1_m':(24.25, 0, 0.54),
'dm':(41.3, -0.25, 0.13),
'd1m':(27.43, 0, 0.25),
'l_m':(102, -1, 1),
'd0':(19.1,-0.41,0.2),
'p_n':(2.54,0,0),
'p_m':(2.54,0,0)}
rod22={'d_n':(27, -0.48, -0.376),#
'd2_n':(25.35, -0.204, -0.047),#
'd1_n':(24.25, 0, -0.415),
'r_n':(0.28, 0, 0.08),
'dn':(38.1, -0.25, 0.13),
'd1n':(23.24, -0.13, 0.13),
'l1n':(36.5, 0, 1.6),
'l2n':(15, 0.2, 1),
'l3n':(32, 0, 1.5),
'l4n':(48, -1, 1.5),
'r3n':(3, 0, 0.8),

```

```
'd_m':(27, 0, 0.27),
'd2_m':(25.35, 0, 0.202),
'd1_m':(24.25, 0, 0.54),
'dm':(41.3, -0.25, 0.13),
'd1m':(27.43, 0, 0.25),
'lm':(102, -1, 1),
'd0':(19.1,-0.41,0.2),
'p_n':(2.54,0,0),
'p_m':(2.54,0,0)}
```

```
rod={19:rod19,22:rod22} # словник типорозмірів штанг
diameter=19 # діаметр штанги
d={} # словник усіх розмірів моделі
for x in rod[diameter].iterkeys():
    d[x]=Dim(rod[diameter][x]) # копіюємо ключі, а значення перетворюємо в
    розміри Dim
```

```
l_0=0 # скорочення муфти при згвинчуванні (0, якщо задано Bolt Load)
#=====параметри ніпеля штанги=====
d['d_n'] = d['d_n'].min() / 2 # зовнішній діаметр різьби/2
d['d2_n'] = d['d2_n'].min() / 2 # середній діаметр різьби/2
d['d1_n'] = d['d1_n'].min() / 2 # внутрішній діаметр різьби/2!ei*
d['r_n'] = d['r_n'].min() # радіус западин різьби
d['p_n'] = d['p_n'].min() # крок різьби
d['dn'] = d['dn'].min() / 2 # діаметр бурта/2
d['d1n'] = d['d1n'].min() / 2 # діаметр зарізьбової канавки/2
d['l1n'] = d['l1n'].min()+my_iter2 # довжина ніпеля
d['l2n'] = d['l2n'].min()+my_iter2 # довжина зарізьбової канавки
d['l3n'] = d['l3n'].min()+my_iter2 # довжина ніпеля без фаски на різьбі
d['l4n'] = d['l4n'].min()+my_iter2 # довжина ніпеля з буртом
d['r3n'] = d['r3n'].min() # радіус скруглень зарізьбової канавки
d['d0']=d['d0'].min()/2 # діаметр тіла/2
#=====параметри муфти=====
d['d_m'] = d['d_m'].max() / 2 # зовнішній діаметр різьби/2!es*
d['d2_m'] = d['d2_m'].max() / 2 # середній діаметр різьби/2
d['d1_m'] = d['d1_m'].max() / 2 # внутрішній діаметр різьби/2
d['p_m'] = d['p_m'].min() # крок різьби
d['dm'] = d['dm'].min() / 2 # зовнішній діаметр/2
d['d1m'] = d['d1m'].max() / 2 # внутрішній діаметр опорної поверхні/2
d['lm'] = d['lm'].min() / 2 +my_iter2 # довжина муфти/2
#=====допоміжні параметри=====
d['dn_']=d['d2_n']+0.25*d['p_n']/tan(30*pi/180) # зовнішній діаметр вершин
трикутника профілю ніпеля
d['ln_']=d['l1n']-d['l2n'] # z-координата першої западини ніпеля (довжина
різьби ніпеля)
d['dm_']=d['d2_m']-0.25*d['p_m']/tan(30*pi/180) # внутрішній діаметр вершин
трикутника профілю муфти
d['lm_']=d['lm']-11.1 # довжина різьби муфти
d['l2m_']=d['ln_']+ceil((d['l2n']-11.1)/d['p_m'])*d['p_m']-3*d['p_m']/2-
(d['d2_m']-d['d2_n'])*tan(30*pi/180) # z-координата першої западини муфти
#ceil((d['l2n']-11.1)/d['p_m'])*d['p_m'] - перші неробочі витки муфти
```

```

#-3*d['p_m']/2-(d['d2_m']-d['d2_n'])*tan(30*pi/180) - зміщення профілю
муфти
#=====точки характерних кромки моделі=====
en1=(d['dn']/2, d['l1n']+20, 0.0) # верхній торець штанги
en2=(0.0, (d['l1n']+20)/2, 0.0) # вісь ніпеля
en3=(d['d1_n']/2,0.0,0.0) # нижній торець штанги
en4=(d['dn'],d['l1n']+10,0.0) # зовнішній циліндр бурта
enr1=(d['d2_n']-0.25*d['p_n']/tan(30*pi/180)+d['r_n']/sin(30*pi/180)-
d['r_n'],d['l1n'],0.0) # центр першої западини ніпеля
em1=((d['dm']+d['d_m'])/2, d['l1n']-d['lm']+1, 0.0) # нижній торець муфти
(зміщення +1)
em2=(d['dm'],d['l1n']/2,0.0) # зовнішній циліндр муфти
em3=((d['dm']+d['d1m'])/2,d['l1n']-5,0.) # центр Partition face-1 (для Bolt
Load)
em4=((d['dm']+d['d1m'])/2,d['l1n'],0.) # верхній торець муфти
nn=8 # кількість западин ніпеля для дослідження
#mat1=matlib['40'].power(8) # матеріал 1
#mat2=matlib['40'].power(8) # матеріал 2
mat1=matlib['40fesafe'].bilinear() # матеріал 1
mat2=matlib['40fesafe'].bilinear() # матеріал 2
bolt_load=-my_iter1 #-0.1
load1=-1*d['d0']**2/d['dn']**2
load2=-170.0e+6*d['d0']**2/d['dn']**2

def createSketch1():
    "Створює ескіз профілю різьби ніпеля"
    s=model.ConstrainedSketch(name='Sketch-1', sheetSize=200.0)
    # Геометрія
    g1=s.ConstructionLine(angle=0.0, point1=(0.0, 0.0))
    s.HorizontalConstraint(entity=g1)
    s.FixedConstraint(entity=g1)
    g2=s.Line(point1=(0.0, -15.0), point2=(0.0, 15.0))
    s.VerticalConstraint(entity=g2)
    g3=s.Line(point1=(0.0, 15.0), point2=(-20.0, 5.0))
    g4=s.Line(point1=(0.0, -15.0), point2=(-20.0, -5.0))
    g5=s.ArcByStartEndTangent(entity=g3, point1=(-20.0, 5.0), point2=(-
20.0, -5.0))
    s.TangentConstraint(entity1=g4, entity2=g5)
    s.CoincidentConstraint(entity1=g5.getVertices()[2], entity2=g1)
    # Розміри і параметри
    d1=s.VerticalDimension(vertex1=g2.getVertices()[0],
vertex2=g2.getVertices()[1],textPoint=(0.0, 0.0))
    s.Parameter(name='p_n', path='dimensions[0]')
    d2=s.AngularDimension(line1=g1, line2=g3, textPoint=(10.0, 10.0))
    s.Parameter(name='alf1', path='dimensions[1]')
    d3=s.AngularDimension(line1=g1, line2=g4, textPoint=(10.0, -10.0))
    s.Parameter(name='alf2', path='dimensions[2]')
    d4=s.RadialDimension(curve=g5, textPoint=(0.0, 0.0))
    s.Parameter(name='r_n', path='dimensions[3]')
def createSketch2():
    "Створює ескіз профілю різьби муфти"

```

```

s=model.ConstrainedSketch(name='Sketch-2', sheetSize=200.0)
# Геометрія
g1=s.ConstructionLine(angle=0.0,point1=(0.0, 0.0))
s.HorizontalConstraint(entity=g1)
s.FixedConstraint(entity=g1)
g2=s.Line(point1=(0.0, 0.0), point2=(0.0, 20.0))
s.VerticalConstraint(entity=g2)
s.PerpendicularConstraint(entity1=g1, entity2=g2)
s.CoincidentConstraint(entity1=g2.getVertices()[0], entity2=g1)
g3=s.Line(point1=(0.0, 0.0), point2=(0.0, -20.0))
s.VerticalConstraint(entity=g3)
s.ParallelConstraint(entity1=g2, entity2=g3)
s.EqualLengthConstraint(entity1=g2, entity2=g3)
g4=s.Line(point1=(0.0, -20.0), point2=(25.0, -5.0))
g5=s.Line(point1=(25.0, -5.0), point2=(25.0, 5.0))
s.VerticalConstraint(entity=g5)
g6=s.Line(point1=(25.0, 5.0), point2=(0.0, 20.0))
# Розміри і параметри
d1=s.VerticalDimension(vertex1=g2.getVertices()[1],
vertex2=g3.getVertices()[1],textPoint=(0.0, 0.0))
s.Parameter(name='p_m', path='dimensions[0]')
d2=s.AngularDimension(line1=g1, line2=g6, textPoint=(-10.0, 10.0))
s.Parameter(name='alf1', path='dimensions[1]')
d3=s.AngularDimension(line1=g1, line2=g4, textPoint=(-10.0, -10.0))
s.Parameter(name='alf2', path='dimensions[2]')
d4=s.DistanceDimension(entity1=g2, entity2=g5, textPoint=(0.0, 0.0))
s.Parameter(name='h', path='dimensions[3]')
def createSketch3():
    "Створює ескіз заготовки ніпеля"
    s=model.ConstrainedSketch(name='Sketch-3', sheetSize=200.0)
    # Геометрія
    g1=s.Line(point1=(0.0, 0.0), point2=(0.0, 50.0))
    s.VerticalConstraint(entity=g1)
    s.FixedConstraint(entity=g1.getVertices()[0])
    g2=s.Line(point1=(0.0, 50.0), point2=(20.0, 50.0))
    s.HorizontalConstraint(entity=g2)
    g3=s.Line(point1=(20.0, 50.0), point2=(20.0, 40.0))
    s.VerticalConstraint(entity=g3)
    g4=s.Line(point1=(20.0, 40.0), point2=(10.0, 40.0))
    s.HorizontalConstraint(entity=g4)
    g5=s.ArcByStartEndTangent(entity=g4, point1=(10.0, 40.0), point2=(5.0,
35.0))
    g6=s.Line(point1=(5.0, 35.0), point2=(5.0, 30.0))
    s.VerticalConstraint(entity=g6)
    s.TangentConstraint(entity1=g5, entity2=g6)
    g7=s.ArcByStartEndTangent(entity=g6, point1=(5.0, 30.0), point2=(10.0,
25.0))
    g8=s.Line(point1=(10.0, 25.0), point2=(10.0, 5.0))
    s.VerticalConstraint(entity=g8)
    g9=s.Line(point1=(10.0, 5.0), point2=(5.0, 0.0))
    g10=s.Line(point1=(5.0, 0.0), point2=(0.0, 0.0))

```



```

s.HorizontalConstraint(entity=g10)
# Розміри і параметри
d1=s.DistanceDimension(entity1=g2, entity2=g10, textPoint=(0.0, 0.0))
s.Parameter(name='l1n', path='dimensions[0]')
d2=s.DistanceDimension(entity1=g1, entity2=g3, textPoint=(0.0, 0.0))
s.Parameter(name='d1n', path='dimensions[1]')
d3=s.DistanceDimension(entity1=g1, entity2=g8, textPoint=(0.0, 0.0))
s.Parameter(name='d_n', path='dimensions[2]')
d4=s.DistanceDimension(entity1=g1, entity2=g6, textPoint=(0.0, 0.0))
s.Parameter(name='d1n', path='dimensions[3]')
d5=s.DistanceDimension(entity1=g4, entity2=g10, textPoint=(0.0, 0.0))
s.Parameter(name='l1n', path='dimensions[4]')
d6=s.DistanceDimension(entity1=g4, entity2=g8.getVertices()[0],
textPoint=(0.0, 0.0))
s.Parameter(name='l2n', path='dimensions[5]')
d7=s.DistanceDimension(entity1=g4, entity2=g8.getVertices()[1],
textPoint=(0.0, 0.0))
s.Parameter(name='l3n', path='dimensions[6]')
d8=s.AngularDimension(line1=g9, line2=g1, textPoint=(10.0, 10.0))
s.Parameter(name='fn', path='dimensions[7]')
d9=s.RadialDimension(curve=g5, textPoint=(0.0, 0.0))
s.Parameter(name='r3n1', path='dimensions[8]')
d10=s.RadialDimension(curve=g7, textPoint=(0.0, 0.0))
s.Parameter(name='r3n2', path='dimensions[9]')
s.ConstructionLine(point1=(0.0, -100.0), point2=(0.0, 100.0))
def createSketch4():
    "Створює ескіз заготовки муфти"
    s=model.ConstrainedSketch(name='Sketch-4', sheetSize=200.0)
    # Геометрія
    v1=s.Spot(point=(0.0, 0.0))
    s.FixedConstraint(entity=v1)
    g1=s.Line(point1=(10.0, -10.0), point2=(10.0, 20.0))
    s.VerticalConstraint(entity=g1)
    g2=s.Line(point1=(10.0, 20.0), point2=(15.0, 25.0))
    g3=s.Line(point1=(15.0, 25.0), point2=(15.0, 35.0))
    s.VerticalConstraint(entity=g3)
    g4=s.Line(point1=(15.0, 35.0), point2=(20.0, 40.0))
    g5=s.Line(point1=(20.0, 40.0), point2=(25.0, 40.0))
    s.HorizontalConstraint(entity=g5)
    g6=s.Line(point1=(25.0, 40.0), point2=(30.0, 35.0))
    g7=s.Line(point1=(30.0, 35.0), point2=(30.0, -10.0))
    s.VerticalConstraint(entity=g7)
    g8=s.Line(point1=(30.0, -10.0), point2=(10.0, -10.0))
    s.HorizontalConstraint(entity=g8)
    # Розміри і параметри
    d1=s.DistanceDimension(entity1=g5, entity2=v1, textPoint=(0.0, 0.0))
    s.Parameter(name='l1n', path='dimensions[0]')
    d2=s.DistanceDimension(entity1=g5, entity2=g2.getVertices()[1],
textPoint=(0.0, 0.0))
    s.Parameter(name='l1m', path='dimensions[1]')

```

```

d3=s.DistanceDimension(entity1=g5, entity2=g3.getVertices()[1],
textPoint=(0.0, 0.0))
s.Parameter(name='lf1m', path='dimensions[2]')
d4=s.DistanceDimension(entity1=g5, entity2=g6.getVertices()[1],
textPoint=(0.0, 0.0))
s.Parameter(name='lf2m', path='dimensions[3]')
d5=s.DistanceDimension(entity1=g5, entity2=g8, textPoint=(0.0, 0.0))
s.Parameter(name='lm', path='dimensions[4]')
d6=s.DistanceDimension(entity1=g1, entity2=v1, textPoint=(0.0, 0.0))
s.Parameter(name='d1_m', path='dimensions[5]')
d7=s.DistanceDimension(entity1=g3, entity2=v1, textPoint=(0.0, 0.0))
s.Parameter(name='d1m', path='dimensions[6]')
d8=s.DistanceDimension(entity1=g7, entity2=v1, textPoint=(0.0, 0.0))
s.Parameter(name='dm', path='dimensions[7]')
d9=s.AngularDimension(line1=g1, line2=g2, textPoint=(10.0, 10.0))
s.Parameter(name='f_m', path='dimensions[8]')
d10=s.AngularDimension(line1=g3, line2=g4, textPoint=(10.0, 10.0))
s.Parameter(name='f1m', path='dimensions[9]')
d11=s.AngularDimension(line1=g6, line2=g7, textPoint=(50.0, -50.0))
s.Parameter(name='f2m', path='dimensions[10]')
s.ConstructionLine(point1=(0.0, -100.0), point2=(0.0, 100.0))
def create_nipple2():
par={'aint':0,'aext':0,'Rint':0,'Rext':d['dn'],'len':d['l1n']+20}
set_values2(sketch='nipple',base='Quad_h',p=par)
p=model.Part(dimensionality=AXISYMMETRIC, name='Part-1',
type=DEFORMABLE_BODY)
p.BaseShell(sketch=model.sketches['nipple'])
del model.sketches['nipple']

par={'aint':0,'aext':0,'Rint':d['d1n'],'Rext':d['dn'],'len':d['l2n'],'Ra':d
['r3n'],'Rb':d['r3n']}
set_values2(sketch='groove',base='Quad_h_fillet_int',p=par)
part_builder(p, 'groove', (0,d['l1n']-d['l2n']), 'cut')
del model.sketches['groove']

par={'aint':0,'aext':0,'Rint':d['d_n'],'Rext':d['dn'],'len':d['l1n']-
d['l2n']/2}
set_values2(sketch='cut_nipple',base='Quad_h',p=par)
part_builder(p, 'cut_nipple', (0,0), 'cut')
del model.sketches['cut_nipple']

par={'aa':30,'ab':90,'h':d['l1n']-d['l3n']}
set_values2(sketch='chamfer',base='Triangle_v_int',p=par)
part_builder(p, 'chamfer', (d['d_n'],0), 'cut')
del model.sketches['chamfer']

par={'aa':60,'ab':60,'len':d['p_n'],'R':d['r_n']}
set_values2(sketch='thread',base='Triangle_v_fillet_int',p=par)
i=0 # номер витка (0-перший)
while i*d['p_n']<=d['ln_']: # довжина різьби

```

```

        part_builder(p, 'thread', (d['dn_'],d['ln_']-d['p_n']*i), 'cut')
        i+=1
del model.sketches['thread']

def create_coupling2():
    par={'aint':0,'aext':0,'Rint':d['d1_m'],'Rext':d['dm'],'len':d['l1n']-
11.1}
    set_values2(sketch='coupling',base='Quad_h',p=par)
    p=model.Part(dimensionality=AXISYMMETRIC, name='Part-2',
type=DEFORMABLE_BODY)
    p.BaseShell(sketch=model.sketches['coupling'])
    del model.sketches['coupling']

    par={'aint':0,'aext':0,'Rint':d['d1_m'],'Rext':d['dm'],'len':d['lm']-
d['l1n']}
    set_values2(sketch='coupling_bot',base='Quad_h',p=par)
    part_builder(p, 'coupling_bot', (0,-(d['lm']-d['l1n'])), 'shell')
    del model.sketches['coupling_bot']

    par={'aint':0,'aext':0,'Rint':d['d1m'],'Rext':d['dm'],'len':11.1}
    set_values2(sketch='coupling_top',base='Quad_h',p=par)
    part_builder(p, 'coupling_top', (0,d['l1n']-11.1), 'shell')
    del model.sketches['coupling_top']

    #par={'aa':90,'ab':30,'t':d['d1m']-d['d1_m']}
    s=model.ConstrainedSketch(name='Sketch-1', sheetSize=200.0)
    s.Line(point1=(0.0, d['l1n']-11.1), point2=(d['d1m'], d['l1n']-11.1))
    s.Line(point1=(0.0, d['l1n']-11.1), point2=(0.0, (d['l1n']-11.1)-
d['d1m']/tan(radians(30))))
    s.Line(point1=(d['d1m'], d['l1n']-11.1), point2=(0.0, (d['l1n']-11.1)-
d['d1m']/tan(radians(30))))
    # створити клас
    #set_values2(sketch='chamfer',base='Sketch-1',p=par)
    part_builder(p, 'Sketch-1', (0,0), 'cut')
    del model.sketches['Sketch-1']

    par={'aa':90,'ab':30,'h':2}
    set_values2(sketch='chamfer',base='Triangle_v_ext',p=par)
    part_builder(p, 'chamfer', (d['d1m'],d['l1n']-2), 'cut')
    del model.sketches['chamfer']

    par={'aa':90,'ab':15,'h':3}
    set_values2(sketch='chamfer',base='Triangle_v_int',p=par)
    part_builder(p, 'chamfer', (d['dm'],d['l1n']-3), 'cut')
    del model.sketches['chamfer']

    par={'aa':120,'ab':120,'h':d['p_m'],'t':2.2-0.275}
    set_values2(sketch='thread',base='Quad_v',p=par)
    i=0
    while i*d['p_m']<=d['lm']-3*d['p_m']/2: # довжина різьби-3*d['p_m']/2
        part_builder(p, 'thread', (d['dm_'],d['l2m_']-d['p_m']*i), 'cut')

```

```

    i+=1
del model.sketches['thread']

def create_nipple3():
    s=model.ConstrainedSketch(name='nipple', sheetSize=200.0)
    L1=Line(p1=(0,0),len=d['l1n']+20,angle=90)
    L2=Line(p1=L1.p1,len=d['dn'],angle=0)
    L3=Line(p1=L2.p2,len=d['l1n']+20,angle=90)
    L4=Line(p1=L3.p2,p2=L1.p2)
    N_angle([L1,L2,L3,L4], [0,0,0,0]).drawAbaqus(s)
    del L1,L2,L3,L4
    s.ConstructionLine(point1=(0.0, 0.0), angle=90.0)
    p=model.Part(dimensionality=AXISYMMETRIC, name='Part-1',
type=DEFORMABLE_BODY)
    p.BaseShell(sketch=model.sketches['nipple'])
    del s

    s=model.ConstrainedSketch(name='groove', sheetSize=200.0)
    L1=Line(p1=(d['dn'],d['l1n']-d['l2n']),len=d['l2n'],angle=90)
    L2=Line(p1=L1.p2,len=d['dn']-d['d1n'],angle=180)
    L3=Line(p1=L2.p2,len=d['l2n'],angle=360-90)
    L4=Line(p1=L3.p2,p2=L1.p1)
    N_angle([L1,L2,L3,L4], [0,d['r3n'],d['r3n'],0]).drawAbaqus(s)
    del L1,L2,L3,L4
    part_builder(p, 'groove', (0,0), 'cut')
    del s

    s=model.ConstrainedSketch(name='cut_nipple', sheetSize=200.0)
    L1=Line(p1=(d['d_n'],0),len=d['l1n']-d['l2n']/2,angle=90)
    L2=Line(p1=L1.p2,len=d['dn']-d['d_n'],angle=0)
    L3=Line(p1=L2.p2,len=d['l1n']-d['l2n']/2,angle=360-90)
    L4=Line(p1=L3.p2,p2=L1.p1)
    N_angle([L1,L2,L3,L4], [0,0,0,0]).drawAbaqus(s)
    del L1,L2,L3,L4
    part_builder(p, 'cut_nipple', (0,0), 'cut')
    del s

    s=model.ConstrainedSketch(name='chamfer', sheetSize=200.0)
    L1=Line(p1=(d['d_n'],0),len=d['l1n']-d['l3n'],angle=90)
    cp=Line(p1=L1.p1,angle=180).cros_point(Line(p1=L1.p2,angle=180+60))
    L2=Line(p1=L1.p1,p2=cp)
    L3=Line(p1=L1.p2,p2=cp)
    N_angle([L1,L2,L3], [0,0,0]).drawAbaqus(s)
    del L1,L2,L3,cp
    part_builder(p, 'chamfer', (0,0), 'cut')
    del s

    s=model.ConstrainedSketch(name='thread', sheetSize=200.0)
    L1=Line(p1=(d['dn_'],0),len=d['p_n'],angle=90)
    cp=Line(p1=L1.p1,angle=90+60).cros_point(Line(p1=L1.p2,angle=270-60))
    L2=Line(p1=L1.p1,p2=cp)

```

```

L3=Line(p1=L1.p2,p2=cp)
N_angle([L1,L2,L3], [0,d['r_n'],0]).drawAbaqus(s)
del L1,L2,L3,cp
i=0 # номер витка (0-перший)
while i*d['p_n']<=d['ln_']: # довжина різьби
    part_builder(p, 'thread', (0,d['ln_']-d['p_n']*i), 'cut')
    i+=1
del s

def create_coupling3():
s=model.ConstrainedSketch(name='coupling', sheetSize=200.0)
L1=Line(p1=(d['dm'],-(d['lm']-d['l1n'])),len=d['lm']-11.1,angle=90)
L2=Line(p1=L1.p2,len=d['dm']-d['d1_m'],angle=180)
L3=Line(p1=L2.p2,len=d['lm']-11.1,angle=270)
L4=Line(p1=L3.p2,p2=L1.p1)
N_angle([L1,L2,L3,L4], [0,0,0,0]).drawAbaqus(s)
del L1,L2,L3,L4
s.ConstructionLine(point1=(0.0, 0.0), angle=90.0)
p=model.Part(dimensionality=AXISYMMETRIC, name='Part-2',
type=DEFORMABLE_BODY)
p.BaseShell(sketch=model.sketches['coupling'])
del s

s=model.ConstrainedSketch(name='top', sheetSize=200.0)
L1=Line(p1=(d['dm'],d['l1n']-11.1),len=11.1,angle=90)
L2=Line(p1=L1.p2,len=d['dm']-d['d1m'],angle=180)
L3=Line(p1=L2.p2,len=11.1,angle=270)
L4=Line(p1=L3.p2,p2=L1.p1)
N_angle([L1,L2,L3,L4], [0,0,0,0]).drawAbaqus(s)
del L1,L2,L3,L4
part_builder(p, 'top', (0,0), 'shell')
del s

s=model.ConstrainedSketch(name='chamfer', sheetSize=200.0)
L1=Line(p1=(d['d1m'],d['l1n']-11.1),len=d['d1m']-d['d1_m'],angle=180)
cp=Line(p1=L1.p1,angle=180+60).cros_point(Line(p1=L1.p2,angle=270))
L2=Line(p1=L1.p1,p2=cp)
L3=Line(p1=L1.p2,p2=cp)
N_angle([L1,L2,L3], [0,0,0]).drawAbaqus(s)
del L1,L2,L3,cp
part_builder(p, 'chamfer', (0,0), 'cut')
del s

s=model.ConstrainedSketch(name='chamfer', sheetSize=200.0)
L1=Line(p1=(d['d1m'],d['l1n']),len=2.0,angle=270)
cp=Line(p1=L1.p1,angle=0).cros_point(Line(p1=L1.p2,angle=90-30))
L2=Line(p1=L1.p1,p2=cp)
L3=Line(p1=L1.p2,p2=cp)
N_angle([L1,L2,L3], [0,0,0]).drawAbaqus(s)
del L1,L2,L3,cp
part_builder(p, 'chamfer', (0,0), 'cut')

```

```

del s

s=model.ConstrainedSketch(name='chamfer', sheetSize=200.0)
L1=Line(p1=(d['dm'],d['ln']),len=3.0,angle=270)
cp=Line(p1=L1.p1,angle=180).cros_point(Line(p1=L1.p2,angle=90+15))
L2=Line(p1=L1.p1,p2=cp)
L3=Line(p1=L1.p2,p2=cp)
N_angle([L1,L2,L3], [0,0,0]).drawAbaqus(s)
del L1,L2,L3,cp
part_builder(p, 'chamfer', (0,0), 'cut')
del s

s=model.ConstrainedSketch(name='thread', sheetSize=200.0)
L1=Line(p1=(d['dm_'],0),len=d['p_m'],angle=90)
cp1=Line(p1=L1.p1,angle=90-
60).cros_point(Line(p1=(d['dm_']+1.93,0),angle=90))
L2=Line(p1=L1.p1,p2=cp1)
cp2=Line(p1=L1.p2,angle=270+60).cros_point(Line(p1=cp1,angle=90))
L3=Line(p1=cp1,p2=cp2)
L4=Line(p1=L1.p2,p2=cp2)
N_angle([L1,L2,L3,L4], [0,0,0,0]).drawAbaqus(s)
del L1,L2,L3,L4,cp1,cp2
i=0
while i*d['p_m']<=d['lm_']-3*d['p_m']/2:#довжина різьби-3*d['p_m']/2
    part_builder(p, 'thread', (0,d['l2m_']-d['p_m']*i), 'cut')
    i+=1
del s
def createProfile():
    "Створює профіль різьби ніпеля і муфти"
    delCutExtrude()
    # створення профілю різьби ніпеля
    # можна це зробити також за допомогою LinearInstancePattern
    i=0 # номер витка (0-перший)
    while i*d['p_n']<=d['ln_']:#довжина різьби
        s=model.ConstrainedSketch(name='__profile__',sheetSize=200.)
        #s.sketchOptions.setValues(viewStyle=AXISYM)
        model.parts['Part-
1'].projectReferencesOntoSketch(filter=COPLANAR_EDGES, sketch=s)
        s.ConstructionLine(point1=(0.0,0.0), point2=(0.0, 10.0))
        s.retrieveSketch(sketch=model.sketches['Sketch-1'])
        s.move(objectList=s.geometry.values(),vector=(d['dn_'],d['ln_']-
d['p_n']*i))
        model.parts['Part-1'].Cut(sketch=s)
        del s
        i=i+1
    # створення профілю різьби муфти
    i=0
    while i*d['p_m']<=d['lm_']-3*d['p_m']/2: # довжина різьби-3*d['p_m']/2
        s=model.ConstrainedSketch(name='__profile__',sheetSize=200.)
        #s.sketchOptions.setValues(viewStyle=AXISYM)

```

```

    model.parts['Part-
2'].projectReferencesOntoSketch(filter=COPLANAR_EDGES, sketch=s)
    s.ConstructionLine(point1=(0.0,0.0), point2=(0.0, 10.0))
    s.retrieveSketch(sketch=model.sketches['Sketch-2'])
    s.move(objectList=s.geometry.values(),vector=(d['dm_'],d['l2m_']-
d['p_m']*i))
    model.parts['Part-2'].Cut(sketch=s)
    del s
    i=i+1

def create_nipple_coupling():
    '''Спосіб побудови геометрії за ескізами заготовок деталей'''
    createSketch1()
    createSketch2()
    createSketch3()
    createSketch4()
    # параметри профілю різьби ніпеля
    par={'p_n':d['p_n'],'alf1':30,'alf2':30,'r_n':d['r_n']}
    set_values(sketch='Sketch-1',p=par)
    # параметри профілю різьби муфти
    par={'p_m':d['p_m'],'alf1':30,'alf2':30,'h':2.2-0.275}
    set_values(sketch='Sketch-2',p=par)
    # параметри заготовки ніпеля

par={'l1n':d['l1n']+20,'d_n':d['d_n'],'dn':d['dn'],'d1n':d['d1n'],'l1n':d['l
1n'],

'l2n':d['l2n'],'l3n':d['l3n'],'r3n1':d['r3n'],'r3n2':d['r3n'],'fn':30}
set_values(sketch='Sketch-3',p=par)
# параметри заготовки муфти

par={'dm':d['dm'],'d1m':d['d1m'],'d1_m':d['d1_m'],'lm':d['lm'],'l1n':d['l1n
']+1_,
    'lf1m':2,'lf2m':3,'f1m':30,'f2m':15,'f_m':30,'l1m':11.1}
set_values(sketch='Sketch-4',p=par)
createPart(n='Part-1',s='Sketch-3')
createPart(n='Part-2',s='Sketch-4')
createProfile()
def create_nipple_coupling2():
    '''Спосіб побудови геометрії за готовими простими ескізами'''
    create_nipple2()
    create_coupling2()
def create_nipple_coupling3():
    '''Спосіб побудови геометрії за простими ескізами'''
    create_nipple3()
    create_coupling3()

create_nipple_coupling()
#create_nipple_coupling2()
#create_nipple_coupling3()
#createPart3D('Part-1','nipple','Part-3')

```

```

#createPart3D('Part-2','coupling','Part-4')
createPartition(part='Part-2',offset=em3[1])
#createPartition3D(part='Part-4',offset=em3[1])
# ізотропне зміцнення:
#createMaterial('Material-1',et=mat1['el'],pt=mat1['pl'])
#createMaterial('Material-2',et=mat2['el'],pt=mat2['pl'])
# кінематичне зміцнення:
createMaterial('Material-1',et=mat1['el'],pt=mat1['pl'],kinematic=True)
createMaterial('Material-2',et=mat2['el'],pt=mat2['pl'],kinematic=True)
createSectionAssign(n='Section-1',m='Material-1',p='Part-1')
createSectionAssign(n='Section-2',m='Material-2',p='Part-2')
#createSectionAssign3D(n='Section-1',m='Material-1',p='Part-3')
#createSectionAssign3D(n='Section-2',m='Material-2',p='Part-4')
createAssemblyInstance(n='Part-1-1',p='Part-1')
createAssemblyInstance(n='Part-2-1',p='Part-2')
#createAssemblyInstance3D(n='Part-3-1',p='Part-3')
#createAssemblyInstance3D(n='Part-4-1',p='Part-4')
createStep(n='Step-1',pr='Initial')
createStep(n='Step-2',pr='Step-1')
createContactSet(n='Slave',i='Part-1-1',ep=((en1, ), (en2, ), (en3, ),
(en4, ),)) # створюємо набір кромок контакту для ніпеля
createContactSet(n='Master',i='Part-2-1',ep=((em1, ), (em2, ),(em3,))) #
створюємо набір кромок контакту для муфти
#createContactSet3D(n='Slave',i='Part-3-1',ep=((en1, ), (en3, ), (en4, ),))
# створюємо набір кромок контакту для ніпеля
#createContactSet3D(n='Master',i='Part-4-1',ep=((em1, ), (em2, ),(em3,)))
# створюємо набір кромок контакту для муфти
createContactProperty()
createContact()
#createContact3D()
createBCSet(n='Pressure',i='Part-1-1',ep=(en1, )) # тиск
createBCSet(n='Axis',i='Part-1-1',ep=(en2, )) # закріплення
createBCSet(n='Encastre',i='Part-2-1',ep=(em1, )) # закріплення
#createBCSet3D(n='Pressure',i='Part-3-1',ep=(en1, )) # тиск
#createBCSet3D(n='Encastre',i='Part-4-1',ep=(em1, )) # закріплення
createBC_Pressure(['Step-1',load1],['Step-2',load2])
createBC_Axis()
createBC_Encastre()
createBC_BoltLoad('Part-2-1',em3,bolt_load)
#createBC_Pressure3D(['Step-1',-1.0],['Step-2',-
276.0e+6*d['d0']**2/d['dn']**2])
#createBC_Encastre3D()
#createBC_BoltLoad3D('Part-2-1',em3,-0.1)
createMesh()
#createMesh3D()
createEdgesSet(n='Cont',i='Part-2-1',p=((em4, ),) )
createEdgesSet(n='First',i='Part-1-1',p=((enr1, ),) )
createVerticesSet(n='Zero point',i='Part-1-1',p=((0,0,0),) )
createJobSubmit()

```



```

def createResults():
    global myOdb
    myOdb = openOdb(path=model.name + '.odb')
    session.viewports['Viewport: 1'].setValues(displayedObject=myOdb)
    SF_field()
    cont_pres=readODB_set(set='Cont',step='Step-2',var= (('CPRESS',
ELEMENT_NODAL), ),pos=NODAL)
    result1=sum(cont_pres)/len(cont_pres)

result2=min(readODB_set2(set='First',step='',var=('D',''),pos=INTEGRATION_P
OINT))
    result3=max(readODB_set(set='First',step='Step-2',var= (('S',
INTEGRATION_POINT, ((INVARIANT, 'Mises'), )), ),pos=INTEGRATION_POINT))
    result4=readODB_path(path=((d['d1n'], d['l1n']-d['l2n']]/2,
0.0),),step='Step-1',var= (('S', INTEGRATION_POINT, ((INVARIANT, 'Mises'),
)), ),intersections=False)
    myOdb.close()

# результати з fe-safe
oodb='results' # назва бази даних результатів
runFeSafe('Model-1','my',oodb) # виконати fe-safe
myOdb = openOdb(path=oodb + '.odb') # відкрити базу даних результатів
session.viewports['Viewport: 1'].setValues(displayedObject=myOdb)

# отримати логарифм довговічності у множині вузлів Set-1
var= (('LOGLife-Repeats', ELEMENT_NODAL), )
x1=readODB_set_(set='SLAVE',var=var)
x1min = min([x[0][1] for x in x1]) # знайти мінімальне значення

# отримати відсоток відмов у множині вузлів Set-1
var= (('%%Failure@Life=5E6-Repeats', ELEMENT_NODAL), )
x3=readODB_set_(set='SLAVE',var=var)
x3max = max([x[0][1] for x in x3]) # знайти максимальне значення

myOdb.close()

writer.writerow([my_iter1,my_iter2,result1,result2,result3,result4,x1min,x3
max])

createResults()

#saveDB('B')
#readDB('B')

```

Лістинг И.3 – gost5286_75.py

```

# -*- coding: cp1251 -*-
if _my_thread_model!=2: from tools import *

zn80={'D':(80,-0.5,0.5), # зовнішній діаметр труби ніпеля
'D1':(76.5,-0.5,0.5), # зовнішній діаметр упорного торця

```

```

'd3':(25.0,-0.6,0.6), # внутрішній діаметр ніпеля
'd4':(36.0,-0.6,0.6), # внутрішній діаметр муфти
'L2':(240.0,0.0,0.0), # довжина муфти*
'dsr':(60.080,0.0,0.0), # середній діаметр різьби в основній площині
'd5':(66.674,0.0,0.0), # діаметр більшої основи конуса ніпеля*
'd6':(47.674,0.0,0.0), # діаметр меншої основи конуса ніпеля*
'l3':(76.0,-2.0,0), # довжина конуса ніпеля
'd7':(68.3,-0.6,0.6), # діаметр конічної виточки в площині торця муфти
'd8':(61.422,0.0,0.0), # внутрішній діаметр різьби в площині торця
муфти*
'l4':(82.0,0.0,0.0), # відстань від торця до кінця різьби з повним
профілем муфти (не менше)
'P':(5.080,0.0,0.0), # крок різьби паралельно осі різьби
'fi':(atan(0.25/2)*180/pi,0.0,0.0), # кут нахилу
'H':(4.376,0.0,0.0), # висота гострокутного профілю
'h1':(2.993,0.0,0.0), # висота профілю різьби
'h':(2.626,0.0,0.0), # робоча висота профілю
'l':(0.875,0.0,0.0), # висота зрізу вершин
'f':(0.508,0.0,0.0), # відтин впадини
'a':(1.016,0.0,0.0), # площадка*
'r':(0.508,0.0,0.0), # радіус заокруглень впадин*
'r_':(0.38,0.0,0.0)} # радіус спряжень (не більше)
zn95={'D':(80, -0.5, 0.5),
'D1':(76.5,-0.5,0.5),
'd3':(25.0,-0.6,0.6),
'd4':(36.0,-0.6,0.6),
'L2':(240.0,0.0,0.0),
'dsr':(60.080,0.0,0.0),
'd5':(66.674,0.0,0.0),
'd6':(47.674,0.0,0.0),
'l3':(76.0,-2.0,0),
'd7':(68.3,-0.6,0.6),
'd8':(61.422,0.0,0.0),
'l4':(82.0,0.0,0.0),
'P':(5.080,0.0,0.0),
'fi':(atan(0.25/2)*180/pi,0.0,0.0),
'H':(4.376,0.0,0.0),
'h1':(2.993,0.0,0.0),
'h':(2.626,0.0,0.0),
'l':(0.875,0.0,0.0),
'f':(0.508,0.0,0.0),
'a':(1.016,0.0,0.0),
'r':(0.508,0.0,0.0),
'r_':(0.38,0.0,0.0)}

zamok={80:zn80,95:zn95} # словник типорозмірів
diameter=80 # типорозмір
d={} # словник усіх розмірів моделі
for x in zamok[diameter].iterkeys():
    d[x]=Dim(zamok[diameter][x]) # копіюємо ключі, а значення перетворюємо
в розміри Dim

```

```

d['D'].v=d['D'].min()/2
d['D1'].v=d['D1'].min()/2
d['d3'].v=d['d3'].max()/2
d['d4'].v=d['d4'].max()/2
d['L2'].v=d['L2'].n/2
d['dsr'].v=d['dsr'].n/2
d['d5'].v=d['d5'].n/2
d['d6'].v=d['d6'].n/2
d['l3'].v=d['l3'].min()
d['d7'].v=d['d7'].max()/2
d['d8'].v=d['d8'].max()/2
d['l4'].v=d['l4'].n
d['P'].v=d['P'].n
d['fi'].v=d['fi'].n
d['H'].v=d['H'].n
d['h1'].v=d['h1'].n
d['h'].v=d['h'].n
d['l'].v=d['l'].n
d['f'].v=d['f'].n
d['a'].v=d['a'].n
d['r'].v=d['r'].n
d['r_'].v=d['r_'].n
#=====точки характерних кромок моделі=====
en1=((d['D'].v+d['d3'].v)/2,d['l3'].v+20.0,0.0) # верхній торець ніпеля
en2=(d['d3'].v,d['l3'].v/2,0.0) # внутрішній циліндр ніпеля
en3=(d['D'].v,d['l3'].v+10.0,0.0) # зовнішній циліндр ніпеля
em1=((d['D'].v+d['d4'].v)/2,d['l3'].v-d['L2'].v,0.0) # нижній торець муфти
em2=(d['D'].v,0.0,0.0) # зовнішній циліндр муфти
em3=(d['d4'].v,d['l3'].v-d['L2'].v+5.0,0.0) # внутрішній циліндр муфти
em4=((d['D'].v+d['d7'].v)/2,d['l3'].v-8.0,0.0) # близько центру Partition
face-1 (для Bolt Load)
mat1=matlib['40'].power(8) # матеріал 1
mat2=Material(E=210000.0e+6,mu=0.28,st=my_iter2*1e+6,sv=750.0e+6,delta=18.0
,psi=60.0).power(8) # матеріал 2
bolt_load=-my_iter1 #-0.1
load1=-1
load2=-155.1e+6 #-1.0e+6/(pi*(d['D'].v/1000)**2-pi*(d['d3'].v/1000)**2) #
або в ньютонках

def createProfile():
    "Створює профіль різьби ніпеля і муфти"
    # створення профілю різьби ніпеля
    createCut(Part='Part-1',Sketch='Sketch-
3',Begin=0,P=d['P'].v,Fi=d['fi'].v,Len=d['l3'].v-
15.875+d['P'].v,X=d['dsr'].v,Y=d['l3'].v-15.875,dx=-1,dy=-1)
    # збіг різьби ніпеля
    createCut(Part='Part-1',Sketch='Sketch-
3',Begin=1,P=d['P'].v,Fi=d['fi'].v,Len=6.35,X=d['dsr'].v,Y=d['l3'].v-
15.875,dx=1,dy=1)
    # створення профілю різьби муфти

```

```

createCut(Part='Part-2',Sketch='Sketch-
4',Begin=0,P=d['P'].v,Fi=d['fi'].v,Len=d['l4'].v-
16,X=d['dsr'].v,Y=d['l3'].v-15.875,dx=-1,dy=-1)

# параметри заготовки ніпеля
par={'l3':d['l3'].v,'d6':d['d6'].v,'fi':d['fi'].v,'D':d['D'].v,'D1':d['D1']
.v,
     'd3':d['d3'].v}
set_values(sketch='Sketch-1',p=par)
# параметри заготовки муфти
par={'l3':d['l3'].v,'l4':d['l4'].v+2,'fi':d['fi'].v,'D':d['D'].v,'d4':d['d4']
.v,
     'l2':d['l2'].v,'D1':d['D1'].v,'d7':d['d7'].v,'d8':d['d8'].v}
set_values(sketch='Sketch-2',p=par)
# параметри профілю різьби ніпеля
par={'fi':d['fi'].v,'H_21':d['H'].v/2,'H_22':d['H'].v/2,'r':d['r'].v}
set_values(sketch='Sketch-3',p=par)
# параметри профілю різьби муфти
par={'fi':d['fi'].v,'H_21':d['H'].v/2,'H_22':d['H'].v/2,'r':d['r'].v}
set_values(sketch='Sketch-4',p=par)

createPart(n='Part-1',s='Sketch-1')
createPart(n='Part-2',s='Sketch-2')
createProfile()
createPartition(part='Part-2',offset=d['l3'].v-8.0)
createMaterial('Material-1',et=mat1['el'],pt=mat1['pl'])
createMaterial('Material-2',et=mat2['el'],pt=mat2['pl'])
createSectionAssign(n='Section-1',m='Material-1',p='Part-1')
createSectionAssign(n='Section-2',m='Material-2',p='Part-2')
createAssemblyInstance(n='Part-1-1',p='Part-1')
createAssemblyInstance(n='Part-2-1',p='Part-2')
createStep(n='Step-1',pr='Initial')
createStep(n='Step-2',pr='Step-1')
createContactSet(n='Slave',i='Part-1-1',ep=((en1, ), (en2, ), (en3, ),)) #
створюємо набір кромки контакту для ніпеля
createContactSet(n='Master',i='Part-2-1',ep=((em1, ), (em2, ),(em3, ),(em4,
),)) # створюємо набір кромки контакту для муфти
createContactProperty()
createContact()
createBCSet(n='Pressure',i='Part-1-1',ep=(en1, )) # тиск
createBCSet(n='Encastre',i='Part-2-1',ep=(em1, )) # закріплення
createBC_Pressure(['Step-1',load1],['Step-2',load2])
createBC_Encastre()
createBC_BoltLoad('Part-2-1',em4,bolt_load)
createMesh()
createEdgesSet(n='Cont',i='Part-2-1',p((((36.725, 74.0, 0)), ),) )
createEdgesSet(n='First',i='Part-1-1',p((((27.725, 53.045, 0)), ),) )
createJobSubmit()

myOdb = openOdb(path=model.name + '.odb')
def createResults():

```

```

    session.viewports['Viewport: 1'].setValues(displayedObject=myOdb)
    SF_field()
    cont_pres=readODB_set(set='Cont',step='Step-2',var= (('CPRESS',
ELEMENT_NODAL), ),pos=NODAL)
    result1=sum(cont_pres)/len(cont_pres)

result2=min(readODB_set2(set='First',step='',var=('D',''),pos=INTEGRATION_P
OINT))
    result3=max(readODB_set(set='First',step='Step-2',var= (('S',
INTEGRATION_POINT, ((INVARIANT, 'Mises'), )), ),pos=INTEGRATION_POINT))
    writer.writerow([my_iter1,my_iter2,result1,result2,result3])

createResults()
myOdb.close()

```

Лістинг И.4 – gost633_80.py

```

# -*- coding: cp1251 -*-
if _my_thread_model!=3: from tools import *

nkt114={'D':(114.3,0.0,0.0), # зовнішній діаметр труби
'd':(100.3,0.0,0.0), # внутрішній діаметр труби
'Dm':(132.1,0.0,0.0), # зовнішній діаметр муфти
'Lm':(156.0,0.0,0.0), # довжина муфти*
'P':(3.175,0.0,0.0), # крок різьби паралельно осі різьби
'dsr':(112.566,0.0,0.0), # середній діаметр різьби в основній площині
'd1':(111.031,0.0,0.0), # зовнішній діаметр різьби в площині торця
труби
'd2':(107.411,0.0,0.0), # внутрішній діаметр різьби в площині торця
труби
'L':(65.0,-3.2,3.2), # загальна довжина різьби труби
'l':(52.3,0.0,0.0), # довжина різьби труби до основної площини (з
повним профілем)
'l1':(10.0,0.0,0.0), # максимальна довжина збігу різьби труби
'd3':(111.219,0.0,0.0), # внутрішній діаметр різьби в площині торця
муфти
'd0':(115.9,0.0,0.8), # діаметр циліндричної виточки муфти
'l0':(9.5,-0.5,1.5), # глибина виточки муфти
'A':(6.5,0.0,0.0), # натяг при згвинчуванні вручну
'fi':(atan(1.0/32)*180/pi,0.0,0.0), # кут нахилу
'H':(2.75,0.0,0.0), # висота вихідного профілю
'h1':(1.81,-0.1,0.05), # висота профілю різьби
'h':(1.734,0.0,0.0), # робоча висота профілю
'alfa_2':(30.0,-1.0,1.0), # кут нахилу сторони профілю alfa/2
'r':(0.508,0.0,0.045), # радіус заокруглення вершини профілю
'r1':(0.432,-0.045,0.0)} # радіус заокруглення впадини профілю
nkt102={'D':(114.3,-0.9,0.9),
'd':(100.3,0.0,0.0),
'Dm':(132.1,0.0,0.0),
'Lm':(156.0,0.0,0.0),
'P':(3.175,0.0,0.0),

```

```

'dsr':(112.566,0.0,0.0),
'd1':(111.031,0.0,0.0),
'd2':(107.411,0.0,0.0),
'L':(65.0,-3.2,3.2),
'l':(52.3,0.0,0.0),
'l1':(10.0,0.0,0.0),
'd3':(111.219,0.0,0.0),
'd0':(115.9,0.0,0.8),
'l0':(9.5,-0.5,1.5),
'A':(6.5,0.0,0.0),
'fi':(atan(0.0625/2)*180/pi,0.0,0.0),
'H':(2.75,0.0,0.0),
'h1':(1.81,-0.1,0.05),
'h':(1.734,0.0,0.0),
'alfa_2':(30.0,-1.0,1.0),
'r':(0.508,0.0,0.045),
'r1':(0.432,-0.045,0.0)}

```

```

nkt={114:nkt114,102:nkt102} # словник типорозмірів
diameter=114 # типорозмір
d={} # словник усіх розмірів моделі
for x in nkt[diameter].iterkeys():
    d[x]=Dim(nkt[diameter][x]) # копіюємо ключі, а значення перетворюємо в
розміри Dim

```

```

d['D'].v=d['D'].n/2
d['d'].v=d['d'].max()/2
d['Dm'].v=d['Dm'].min()/2
d['Lm'].v=d['Lm'].n/2
d['P'].v=d['P'].n
d['dsr'].v=d['dsr'].n/2
d['d1'].v=d['d1'].n/2
d['d2'].v=d['d2'].n/2
d['L'].v=d['L'].min()
d['l'].v=d['l'].n
d['l1'].v=d['l1'].n
d['d3'].v=d['d3'].n/2
d['d0'].v=d['d0'].min()/2
d['l0'].v=d['l0'].max()
d['A'].v=d['A'].n
d['fi'].v=d['fi'].n
d['H'].v=d['H'].n
d['h1'].v=d['h1'].max()
d['h'].v=d['h'].n
d['alfa_2'].v=d['alfa_2'].n
d['r'].v=d['r'].max()
d['r1'].v=d['r1'].min()
#=====точки характерних кромок моделі=====
en1=((d['D'].v+d['d'].v)/2,d['L'].v+20,0.0) # верхній торець ніпеля
en2=(d['d'].v,d['L'].v/2,0.0) # внутрішній циліндр ніпеля
en3=(d['D'].v,d['L'].v+20-5,0.0) # зовнішній циліндр ніпеля

```

```

em1=(d['Dm'].v-5,d['L'].v-d['A'].v-d['Lm'].v,0.0) # нижній торець муфти
em2=(d['Dm'].v,0.0,0.0) # зовнішній циліндр муфти
mat1=matlib['40'].power(8) # матеріал 1
mat2=matlib['40'].power(8) # матеріал 2
bolt_load=my_iter1
load1=-1
load2=-my_iter2*1e+6

def createProfile():
    '''Створює профіль різьби ніпеля і муфти'''
    #X,Y=const+-n*p
    # створення профілю різьби ніпеля
    x=model.sketches['Sketch-3'].parameters['x'].value # допоміжний
    параметр
    dsr=d['D'].v-x # середній діаметр в основній площині
    createCut(Part='Part-1',Sketch='Sketch-
    3',Begin=0,P=d['P'].v,Fi=d['fi'].v,Len=d['L'].v-12.7,X=dsr,Y=d['L'].v-
    12.7,dx=-1,dy=-1)
    # витки з зрізаними вершинами
    n=createCut(Part='Part-1',Sketch='Sketch-
    3',Begin=1,P=d['P'].v,Fi=d['fi'].v,Len=12.7-d['l1'].v,X=d['D'].v-x,
    Y=d['L'].v-12.7,dx=1,dy=1)
    # збіг різьби
    createCut(Part='Part-1',Sketch='Sketch-
    3',Begin=1,P=d['P'].v,Fi=10.0,Len=d['l1'].v,X=d['D'].v-x,Y=d['L'].v-
    12.7+n*d['P'].v,dx=1,dy=1)
    #12.7 - кратне усім стандартним крокам
    #d['D'].v-x - середній діаметр в основній площині
    # створення профіля різьби муфти
    x=model.sketches['Sketch-4'].parameters['x'].value # допоміжний
    параметр
    createCut(Part='Part-2',Sketch='Sketch-
    4',Begin=0,P=d['P'].v,Fi=d['fi'].v,Len=d['Lm'].v-4,X=dsr,Y=d['L'].v-
    12.7,dx=-1,dy=-1)
    #X=d['d3'].v+x,Y=d['L'].v-d['A'].v

# параметри заготовки ніпеля
par={'ln':d['L'].v+20,'D':d['D'].v,'d':d['d'].v,'d2':d['d2'].v,'l':d['L'].v
-12.7,'fi':d['fi'].v}
set_values(sketch='Sketch-1',p=par)
# параметри заготовки муфти
par={'Dm':d['Dm'].v,'Lm':d['Lm'].v,'d3':d['d3'].v,'d0':d['d0'].v,
'l0':d['l0'].v,'fi':d['fi'].v,'lA':d['L'].v-
d['A'].v,'hk':d['h1'].v+0.5}
set_values(sketch='Sketch-2',p=par)
# параметри профілю різьби ніпеля
par={'fi':d['fi'].v,'P':d['P'].v,'r1':d['r1'].v,'r':d['r'].v}
set_values(sketch='Sketch-3',p=par)
# параметри профілю різьби муфти
par={'fi':d['fi'].v,'P':d['P'].v,'r1':d['r1'].v,'r':d['r'].v}
set_values(sketch='Sketch-4',p=par)

```

```

createPart(n='Part-1',s='Sketch-1')
createPart(n='Part-2',s='Sketch-2')
createProfile()
createMaterial('Material-1',et=mat1['el'],pt=mat1['pl'])
createMaterial('Material-2',et=mat2['el'],pt=mat2['pl'])
createSectionAssign(n='Section-1',m='Material-1',p='Part-1')
createSectionAssign(n='Section-2',m='Material-2',p='Part-2')
createAssemblyInstance(n='Part-1-1',p='Part-1')
createAssemblyInstance(n='Part-2-1',p='Part-2')
createStep(n='Step-1',pr='Initial')
createStep(n='Step-2',pr='Step-1')
createContactSet(n='Slave',i='Part-1-1',ep=((en1, ), (en2, ), (en3, ),)) #
створюємо набір кромок контакту для ніпеля
createContactSet(n='Master',i='Part-2-1',ep=((em1, ), (em2, ),)) #
створюємо набір кромок контакту для муфти
createContactProperty()
createContact()
createBCSet(n='Pressure',i='Part-1-1',ep=(en1, )) # тиск
createBCSet(n='Encastre',i='Part-2-1',ep=(em1, )) # закріплення
createBC_Pressure(['Step-1',load1],['Step-2',load2])
createBC_Encastre()
createMesh()
model.rootAssembly.translate(instanceList=('Part-2-1', ),
    vector=(0.0, bolt_load*d['P'].v, 0.0)) # моделювання згвинчування
(0(вручну),1,2(станок))

work_list=range(13) # розглядаємо 13 витків ніпеля
n=0
#(56.2047863424959,45.1455709929578,0) - координати робочої сторони першого
витка
for x in work_list: # створюємо Set для кожної робочої сторони витка
    x=(56.2047863424959-n*d['P'].v*tan(radians(d['fi'].v)),
45.1455709929578-n*d['P'].v, 0)
    createEdgesSet(n='work'+str(n),i='Part-1-1',p=((x, ),) )
    n+=1
n=0
nwork_list=range(13) # розглядаємо 13 витків ніпеля
for x in nwork_list: # створюємо Set для кожної неробочої сторони витка
    x=(56.1551769672515-n*d['P'].v*tan(radians(d['fi'].v)),
43.5580709924834-n*d['P'].v, 0)
    createEdgesSet(n='nwork'+str(n),i='Part-1-1',p=((x, ),) )
    n+=1

createJobSubmit()

myOdb = openOdb(path=model.name + '.odb')
def createResults():
    session.viewports['Viewport: 1'].setValues(displayedObject=myOdb)
    for x in range(13):

```



```

        cont_pres=readODB_set2(set='work'+str(x),step='Step-
2',var=('CPRESS',''),pos=NODAL)
        result1=sum(cont_pres)/len(cont_pres)
        cont_pres=readODB_set2(set='nwork'+str(x),step='Step-
2',var=('CPRESS',''),pos=NODAL)
        result2=sum(cont_pres)/len(cont_pres)
        writer.writerow([my_iter1,my_iter2,x,result1,result2])

createResults()
myOdb.close()

```

ЛІСТИНГ И.5 – main.py

```

# -*- coding: cp1251 -*-
_my_thread_model=1
import csv
csv_file=open("results.csv", "wb")
writer = csv.writer(csv_file,delimiter = ';')
writer.writerow(['boltload','groove','cont','D','first','centr'])
my_iter2=0
my_iter1=0.1
execfile('tools.py')
execfile('gost13877_96.py')
#for my_iter2 in [10*x for x in range(0,3+1,1)]:
#    for my_iter1 in [x/100.0 for x in range(0,30+5,5)]:
#        execfile('tools.py')
#        execfile('gost13877_96.py')
#        print 'my_iters=',my_iter1,my_iter2
#        Mdb()
#        session.viewports['Viewport: 1'].setValues(displayedObject=None)
csv_file.close()

```

ЛІСТИНГ И.6 – main2.py

```

# -*- coding: cp1251 -*-
_my_thread_model=2
import csv
csv_file=open("results.csv", "wb")
writer = csv.writer(csv_file,delimiter = ';')
writer.writerow(['boltload','sv','cont','Dfirst','Sfirst'])
#my_iter2=0
#my_iter1=0.1
#execfile('tools.py')
#execfile('gost5286_75.py')
for my_iter2 in [100*x for x in range(3,5+1,1)]: # границя плинності
матеріалу муфти, МПа
    for my_iter1 in [x/100.0 for x in range(0,30+5,5)]: # величина
згвинчування, мм
        execfile('tools.py')
        execfile('gost5286_75.py')

```

```
    print 'my_itors=',my_iter1,my_iter2
    Mdb()
    session.viewports['Viewport: 1'].setValues(displayedObject=None)
csv_file.close()
```

ЛІСТИНГ И.7 – main3.py

```
# -*- coding: cp1251 -*-
_my_thread_model=3
import csv
csv_file=open("results.csv", "wb")
writer = csv.writer(csv_file,delimiter = ';')
writer.writerow(['boltload','press','n','cont_work','cont_nwork'])
#my_iter2=155.1
#my_iter1=0
#execfile('tools.py')
#execfile('gost633_80.py')
for my_iter2 in [0.000001,100,200,300]:
    for my_iter1 in [0,1,2]:
        execfile('tools.py')
        execfile('gost633_80.py')
        print 'my_itors=',my_iter1,my_iter2
        Mdb()
        session.viewports['Viewport: 1'].setValues(displayedObject=None)
csv_file.close()
```

ДОДАТОК І

Пакет ThreadsOCC – прикладна САПР різьбових з'єднань

Код програм доступний також в GitHub (vkorey/ThreadsOCC. URL: <http://github.com/vkorey/ThreadsOCC>). Вміст пакету:

- gost13877_96params.py – параметри з'єднання ШН за ГОСТ 13877-96;
- myBaseGeom.py – базові класи PythonOCC;
- cсx_inp.py – робота з файлами calculix .inp;
- cсx_out.py – читає дані з файлу результатів CalculiX .frd;
- main.py – головний модуль.

Лістинг I.1 – gost13877_96params.py

```
# -*- coding: utf-8 -*-
from math import *
steel45={'el':((21000000000.0, 0.28), ),
         'pl':((620000000.0, 0.0),
              (640000000.0, 0.02),
              (800000000.0, 0.04),
              (860000000.0, 0.08),
              (864000000.0, 0.11))}

##
class Dim:
    "Клас описує поняття розміру"
    n=0.0 # номінальний розмір
    ei=0.0 # нижнє відхилення
    es=0.0 # верхнє відхилення
    v=0.0 # дійсне значення
    def __init__(self,n=None,ei=None,es=None,doc=""):
        self.n=n
        self.ei=ei
        self.es=es
        self.__doc__=doc.decode('utf-8')

    def min(self):
        "повертає мінімальний розмір"
        return self.n+self.ei

    def max(self):
        "повертає максимальний розмір"
        return self.n+self.es

##
class Rod(object):
```

```

d_n=Dim(doc="зовнішній діаметр різьби")
d2_n=Dim(doc="середній діаметр різьби")
d1_n=Dim(doc="внутрішній діаметр різьби")
r_n=Dim(doc="радіус западин різьби")
dn=Dim(doc="діаметр бурта")
d1n=Dim(doc="діаметр зарізьбової канавки")
l1n=Dim(doc="довжина ніпеля")
l2n=Dim(doc="довжина зарізьбової канавки")
l3n=Dim(doc="довжина ніпеля без фаски на різьбі")
l4n=Dim(doc="довжина ніпеля з буртом")
r3n=Dim(doc="радіус скруглень зарізьбової канавки")
d_m=Dim(doc="зовнішній діаметр різьби")
d2_m=Dim(doc="середній діаметр різьби")
d1_m=Dim(doc="внутрішній діаметр різьби")
dm=Dim(doc="зовнішній діаметр")
d1m=Dim(doc="внутрішній діаметр опорної поверхні")
lm=Dim(doc="довжина муфти")
d0=Dim(doc="діаметр тіла")
p_n=Dim(doc="крок різьби")
p_m=Dim(doc="крок різьби")

def setModelParams(self,**args):
    self.l_0=0 # скорочення муфти при згвинчуванні (0, якщо задано Bolt
Load)
    #=====параметри ніпеля штанги=====
    self.d_n = self.d_n.min() / 2 # зовнішній діаметр різьби/2
    self.d2_n = self.d2_n.min() / 2 # середній діаметр різьби/2
    self.d1_n = self.d1_n.min() / 2 # внутрішній діаметр різьби/2!ei*
    self.r_n = self.r_n.min() # радіус западин різьби
    self.p_n = self.p_n.min() # крок різьби
    self.dn = self.dn.min() / 2 # діаметр бурта/2
    self.d1n = self.d1n.min() / 2 # діаметр зарізьбової канавки/2
    self.l1n = self.l1n.min() # довжина ніпеля
    self.l2n = self.l2n.min() # довжина зарізьбової канавки
    self.l3n = self.l3n.min() # довжина ніпеля без фаски на різьбі
    self.l4n = self.l4n.min() # довжина ніпеля з буртом
    self.r3n = self.r3n.min() # радіус скруглень зарізьбової канавки
    self.d0 = self.d0.min() / 2 # діаметр тіла/2
    #=====параметри муфти=====
    self.d_m = self.d_m.max() / 2 # зовнішній діаметр різьби/2!es*
    self.d2_m = self.d2_m.max() / 2 # середній діаметр різьби/2
    self.d1_m = self.d1_m.max() / 2 # внутрішній діаметр різьби/2
    self.p_m = self.p_m.min() # крок різьби
    self.dm = self.dm.min() / 2 # зовнішній діаметр/2
    self.d1m = self.d1m.max() / 2 # внутрішній діаметр опорної
поверхні/2
    self.lm = self.lm.min() / 2 # довжина муфти/2

    for k in args:
        self.__dict__[k]=args[k]

```

```

#=====допоміжні параметри=====
self.dn_=self.d2_n+0.25*self.p_n/tan(30*pi/180) # зовнішній діаметр
вершин трикутника профілю ніпеля
self.ln_=self.l1n-self.l2n # z-координата першої западини ніпеля
(довжина різьби ніпеля)
self.dm_=self.d2_m-0.25*self.p_m/tan(30*pi/180) # внутрішній
діаметр вершин трикутника профілю муфти
self.lm_=self.lm-11.1 # довжина різьби муфти
self.l2m_=self.ln_+ceil((self.l2n-11.1)/self.p_m)*self.p_m-
3*self.p_m/2-(self.d2_m-self.d2_n)*tan(30*pi/180) # z-координата першої
западини муфти
#ceil((self.l2n-11.1)/self.p_m)*self.p_m - перші неробочі витки
муфти
#-3*self.p_m/2-(self.d2_m-self.d2_n)*tan(30*pi/180) - зміщення
профілю муфти
#=====точки характерних кромek моделі=====
self.en1=(self.dn/2, self.l4n, 0.0) # верхній торець штанги (було
l1n+20)
self.en2=(0.0, self.l4n/2, 0.0) # вісь ніпеля
self.en3=(self.d1_n/2,0.0,0.0) # нижній торець штанги
self.en4=(self.dn,self.l4n-5,0.0) # зовнішній циліндр бурта
self.enr1=(self.d2_n-
0.25*self.p_n/tan(30*pi/180)+self.r_n/sin(30*pi/180)-self.r_n,self.ln_,0.0)
# центр першої западини ніпеля
self.em1=((self.dm+self.d_m)/2, self.l1n-self.lm+self.l_, 0.0) #
нижній торець муфти (зміщення +self.l_)
self.em2=(self.dm,self.l1n/2,0.0) # зовнішній циліндр муфти
self.em3=((self.dm+self.d1m)/2,self.l1n-5,0.) # центр Partition
face-1 (для Bolt Load)
self.em4=((self.dm+self.d1m)/2,self.l1n,0.) # верхній торець муфти
self.nn=8 # кількість западин ніпеля для дослідження
self.mat1=steel45 # матеріал 1
self.mat2=steel45 # матеріал 2
self.bolt_load=-0.1
self.load1=-1*self.d0**2/self.dn**2
self.load2=-155.1*self.d0**2/self.dn**2

##
class Rod19(Rod):
    d_n=Dim(27, -0.48, -0.376)
    d2_n=Dim(25.35, -0.204, -0.047)
    d1_n=Dim(24.25, 0, -0.415)
    r_n=Dim(0.28, 0, 0.08)
    dn=Dim(38.1, -0.25, 0.13)
    d1n=Dim(23.24, -0.13, 0.13)
    l1n=Dim(36.5, 0, 1.6)
    l2n=Dim(15, 0.2, 1)
    l3n=Dim(32, 0, 1.5)
    l4n=Dim(48, -1, 1.5)
    r3n=Dim(3, 0, 0.8)
    d_m=Dim(27, 0, 0.27)

```

```

d2_m=Dim(25.35, 0, 0.202)
d1_m=Dim(24.25, 0, 0.54)
dm=Dim(41.3, -0.25, 0.13)
d1m=Dim(27.43, 0, 0.25)
lm=Dim(102, -1, 1)
d0=Dim(19.1,-0.41,0.2)
p_n=Dim(2.54,0,0)
p_m=Dim(2.54,0,0)
def __init__(self,**args):
    for k in Rod19.__dict__:
        if Rod19.__dict__[k].__class__==Dim:
            Rod19.__dict__[k].__doc__=Rod.__dict__[k].__doc__ # атрибут
документації
    self.setModelParams(**args)

if __name__=="__main__":
    r=Rod19()
    #r.setModelParams()
    r.dn_

```

Лістинг I.2 – myBaseGeom.py

```

# encoding: utf-8
from OCC import VERSION # версія PythonOCC
print "OCC version",VERSION # версія '0.16.2-0.18.1'

# Геометричний процесор gp - незбережувані базові геометричні об'єкти
# З цими об'єктами працюють за значенням
from OCC.gp import *

# Збережувані базові 3D геометричні об'єкти
# Ці об'єкти STEP-оброблювані і з ними працюють за посиланням
from OCC.Geom import *
# Збережувані базові 2D геометричні об'єкти
# Ці об'єкти STEP-оброблювані і з ними працюють за посиланням
from OCC.Geom2d import *

# Алгоритми для побудови елементарних геометричних об'єктів OCC.Geom
from OCC.GC import *
# Алгоритми для побудови елементарних геометричних об'єктів OCC.Geom2d
from OCC.GCE2d import *

```

Лістинг I.3 – cscx_inp.py

```

# encoding: utf-8
import subprocess

filenameGmsh="gmsh.inp"
filename="model.inp"
nodes={} # вузли

```

```

lines={} # вузли ліній
elements={} # елементи поверхонь

# шаблон нижньої частини файлу .inp
_tmp=""
*MATERIAL, NAME=mat1
*ELASTIC
210000.0, 0.3
*MATERIAL, NAME=mat2
*ELASTIC
210000.0, 0.3
*SOLID SECTION, ELSET=Surface1, MATERIAL=mat1
1.
*SOLID SECTION, ELSET=Surface2, MATERIAL=mat2
1.
*SOLID SECTION, ELSET=Surface3, MATERIAL=mat2
1.
{surfSlaveDefinitions}
{surfMasterDefinitions}
*SURFACE, NAME = surfBoltLoad, TYPE = ELEMENT
{surfBoltLoadDefinitions}
*PRE-TENSION SECTION, NODE={preTNode}, SURFACE=surfBoltLoad
0.0,1.0,0.0
*SURFACE INTERACTION, NAME=Int1
*SURFACE BEHAVIOR, PRESSURE-OVERCLOSURE=LINEAR
**1.e7, 3.
*CONTACT PAIR, INTERACTION=Int1, ADJUST=0.0, TYPE=SURFACE TO SURFACE
Slave, Master
*BOUNDARY
{boundLine1},1,2,0.0
*BOUNDARY
{boundLine2},1,1,0.0
*TIME POINTS,NAME=T1
1.0,2.0

*STEP
*STATIC
**1.e-4,1.
*BOUNDARY
{preTNode}, 1, 1, {boltLoad}
*NODE FILE, TIME POINTS=T1
U,
*EL FILE, TIME POINTS=T1
S,
*CONTACT FILE, TIME POINTS=T1
CDIS, CSTR
*END STEP

*STEP
*STATIC
**1.e-4,1.

```

```

**CLOAD
**{loadLine1}, 2, 400.0
*DLOAD
{surfLoadDefinitions}
**CLOAD
**{preTNode}, 1, -4000.0
*BOUNDARY
{preTNode}, 1, 1, {boltLoad}
*NODE FILE, TIME POINTS=T1
U,
*EL FILE, TIME POINTS=T1
S,
*CONTACT FILE, TIME POINTS=T1
CDIS, CSTR
*END STEP
""

```

```

def mesh():
    """"Створює сітку у Gmsh""""
    gmshPath=r"e:\Portable\gmsh-4.4.0-Windows64\gmsh.exe"
    subprocess.Popen("%s model.brep -2 -o gmsh.inp -format inp -order 2 -
algo del2d -clscale 0.5 -clcurv"%gmshPath, shell=True).wait() # Gmsh-
>Abaqus

```

```

def runCCX():
    """"Виконує розрахунок у ccx""""
    ccxPath=r"e:\CL33-win64\bin\ccx\ccx215.exe"
    s=subprocess.check_output([ccxPath, "-i", filename[:-4]], shell=True)
    L=[ln.strip() for ln in s.splitlines()[-10:]] # останні рядки виведення
    if "Job finished" in L:
        return 1 # якщо розрахунок успішний
    return 0

```

додати заміну дуже малих чисел на 0?

```

def appendNode():
    """"Додає у файл inp визначення нового вузла і повертає його номер""""
    f=readINP(filename) # прочитати рядки
    lns=[] # список нових рядків файлу
    t='***** E L E M E N T S *****\n'
    prev="" # попередній рядок
    for ln in f: # для кожного рядка
        if ln==t: # якщо закінчилась секція вузлів
            lastNode=int(prev.split(',')[0].strip()) # номер останнього
вузла
            # додати визначення ще одного вузла
            lns.append(str(lastNode+1)+", 0, 0, 0\n")
            prev=ln
            lns.append(ln)
    writeINP(lns) # зберегти рядки
    return lastNode+1

```



```

def elset2nset():
    """Конвертує файл gmsH.inp в CalculiX.inp
    аналогічно утиліті Prool's GMSH.inp to CCX.inp
    Увага! Застосовувати відразу після GmsH"""
    f=readINP(filenameGmsH) # прочитати рядки
    lns=[] # список нових рядків файлу
    t="" # поточна секція
    t1="*ELEMENT, type=T3D3, ELSET="
    t2="*NSET, NSET="
    nn=[] # вузли лінії
    for ln in f: # для кожного рядка
        # якщо заголовок секції *ELEMENT, type=T3D3, ELSET=
        if ln.startswith(t1):
            if nn: # якщо не перша секція
                # список вузлів і заголовків
                lnn="\n".join(nn)+"\n"+ln.replace(t1, t2)
            else: # якщо перша секція
                # тільки заголовок
                lnn=ln.replace(t1, t2)
            lns.append(lnn)
            t=t1 # позначити, що ми в середині секції
            nn=[] # очистити список вузлів
        # якщо в середині секції
        elif t==t1 and not ln.startswith("*ELEMENT"):
            ns=[x.strip() for x in ln.split(",")] # вузли
            if not nn: # якщо ми на початку секції
                nn+=ns[1:] # додати три вузла
            else: # якщо ми не на початку секції
                nn+=ns[2:] # додати два вузла
        # якщо секція закінчилась
        elif t==t1 and ln.startswith("*ELEMENT"):
            # список вузлів і заголовків
            lnn="\n".join(nn)+"\n"+ln
            lns.append(lnn)
            t="" # позначити, що ми поза секцією
            nn=[] # очистити список вузлів
        # якщо інші рядки
        else:
            lns.append(ln) # залишаємо без змін
    writeINP(lns) # зберегти рядки

def reverseOrient():
    """Змінює орієнтацію елементів у файлі .inp на протилежну.
    Нумерація вузлів елемента CAХ6 змінюється так:
    1,2,3,4,5,6 -> 1,3,2,6,5,4
    Увага! Застосовувати після elset2nset(), якщо
    орієнтація елементів неправильна"""
    f=readINP() # прочитати рядки
    lns=[] # список нових рядків файлу
    t="" # поточна секція

```

```

t1="*ELEMENT, type=CPS6, ELSET="
for ln in f: # для кожного рядка
    # якщо початок секції
    if ln.startswith(t1):
        t=t1
        lns.append(ln) # залишити без змін
    # якщо в секції
    elif t==t1 and ln.strip()[0] in "0123456789":
        el=[x.strip() for x in ln.split(",")]
        # змінити порядок вузлів
        eln=[el[0],el[1],el[3],el[2],el[6],el[5],el[4]]
        lnn=", ".join(eln)+"\n"
        lns.append(lnn)
    # якщо інші рядки
    else:
        t=""
        lns.append(ln) # залишити без змін
writeINP(lns) # зберегти рядки

```

```

def parse():
    """Парсер файлів calculix .inp
    Увага! Застосовувати після elset2nset() і reverseOrient()"""
    f=readINP() # прочитати рядки
    t="" # поточна секція файлу inp
    t1="*NODE"
    t3="*NSET"
    t4="*ELEMENT"
    nset="" # назва лінії (множини вузлів)
    elset="" # назва поверхні (множини елементів)

    for ln in f: # для кожного рядка
        # парсимо в залежності від рядка ln і секції t
        # заголовок секції *NODE
        if ln.startswith(t1):
            t=t1
        # вузол
        elif t==t1 and ln.strip()[0] in "0123456789":
            ns=[e.strip() for e in ln.split(",")]
            nodes[int(ns[0])] = float(ns[1]), float(ns[2]), float(ns[3])
        # заголовок секції *NSET
        elif ln.startswith(t3):
            t=t3
            nset=ln.split("=")[1].strip()
            lines[nset]=[]
        # вузол лінії
        elif t==t3 and ln.strip()[0] in "0123456789":
            lines[nset].append(int(ln.strip()))
        # секція *Element
        elif ln.startswith(t4):
            t=t4
            elset=ln.split(",")[2].split("=")[1].strip()

```

```

        elements[elset]={}
        # вузли елемента
        elif t==t4 and ln.strip()[0] in "0123456789":
            es=[e.strip() for e in ln.split(",")]
            elements[elset][int(es[0])]=int(es[1]), int(es[2]), int(es[3]),
int(es[4]), int(es[5]), int(es[6])

def findLine(x1,y1,x2,y2):
    """Шукає криву за двома точками"""
    r=11 # точність. Можна дати заокруглення до r знаків, якщо не знаходить
    p=[round(x,r) for x in (x1,y1,x2,y2)] # заокруглити
    p1=p[0],p[1]
    p2=p[2],p[3]
    for ln in lines: # для кожної лінії
        ns=[nodes[x] for x in lines[ln]] # список вузлів на лінії
        nsr=[(round(n[0],r),round(n[1],r)) for n in ns] # заокруглити
        #if p1 in nsr and p2 in nsr: # якщо точки в списку <Тут не зовсім
вірно. Див. нижче>
        # якщо крайні точки збігаються
        if p1==nsr[0] and p2==nsr[-1]:
            return ln
        if p2==nsr[0] and p1==nsr[-1]:
            return ln
    return None

def readINP(filename=filename):
    """Повертає рядки файлу .inp"""
    f=open(filename, 'r')
    ls=f.readlines()
    f.close()
    return ls

def writeINP(ls):
    """Записує рядки ls у файл .inp"""
    f=open(filename, 'w')
    f.writelines(ls)
    f.close()

def writeFinalINP(d):
    """Записує кінцевий файл .inp"""
    f=open(filename, 'r')
    s1=f.read()
    f.close()
    s1=replaceElType(s1)
    s2=_tmp.format(**d)
    f=open(filename, 'w')
    f.write(s1+s2)
    f.close()

def replaceElType(s):
    """Замінює тип елементів у тексті s"""

```

```

old="*ELEMENT, type=CPS6, ELSET="
new="*ELEMENT, type=CAX6, ELSET="
return s.replace(old, new)

def partByElement(e):
    """Визначає деталь за елементом"""
    for s in elements: # для кожної деталі
        if e in elements[s]: # якщо елемент серед елементів деталі
            return s # повертає назву деталі
    return None

def partByLine(ln):
    """Визначає деталь за лінією"""
    b=set(lines[ln]) # множина вузлів лінії
    for s in elements: # для кожної деталі
        a=set() # множина вузлів деталі
        for e in elements[s]: # для кожного елемента деталі
            ns=set(elements[s][e]) # множина вузлів елемента
            a.update(ns) # додати до множини вузлів деталі
        if b.issubset(a): # якщо вузли лінії серед вузлів деталі
            return s # повернути назву деталі
    return None

def findElementSurface(e):
    """Повертає назву лінії і назву сторони елемента на лінії.
    Тільки для елементів типу CAX6 !
    Наприклад, елемент CAX6 має вузли 1,2,3,4,5,6
    тоді сторони елемента нумеруються так:
    s1: 1-2, s2: 2-3, s3: 3-1"""
    s=partByElement(e) # назва деталі
    ns=elements[s][e] # вузли елемента
    for ln in lines: # для кожної лінії
        ns1=lines[ln] # вузли лінії
        nc=set(ns)&set(ns1) # множина спільних вузлів елемента і лінії
        if len(nc)==3: # якщо спільні три вузла (всього 6)
            s1=set(ns[:2])&nc
            s2=set(ns[1:3])&nc
            s3=set((ns[2],ns[0]))&nc
            # якщо обидва кутові вузли на лінії, то
            # повертає назву лінії і назву сторони елемента на лінії
            if len(s1)==2: return (ln,"S1")
            if len(s2)==2: return (ln,"S2")
            if len(s3)==2: return (ln,"S3")
    return (None,None)

def elementsByLine(ln,s=None):
    """Повертає список елементів лінії (код з surfByLineE1)"""
    res=[]
    if s==None: s=partByLine(ln) # назва деталі
    for e in elements[s]: # для кожного елемента деталі
        ls=findElementSurface(e)

```

```

        if ls[0]==ln: # якщо елемент на лінії
            res.append(e)
    return res

def surfByLineEl(ln,s=None):
    """Код визначення *SURFACE за елементами лінії"""
    surdef=""
    if s==None: s=partByLine(ln) # назва деталі

    for e in elements[s]: # для кожного елемента деталі
        ls=findElementSurface(e)
        if ls[0]==ln: # якщо елемент на лінії
            surdef+=str(e)+", "+str(ls[1])+"\n"
    return surdef

def dloadDefs(ln,s,value):
    _=surfByLineEl(ln,s)
    return _.replace('S','P').replace('\n',' ', %f\n'%value)

def surfDefs(lines,surfname):
    """Код визначення *SURFACE, які відповідають lines"""
    s=""
    s=""*SURFACE, NAME = {name}, TYPE = ELEMENT\n"".format(name=surfname)
    for ln in lines:
        if ln: #!!!
            s+=surfByLineEl(ln)
    return s

def surfNodeDefs(lines,surfname):
    """Код визначення *SURFACE TYPE=NODE, які відповідають lines"""
    s=""
    s=""*SURFACE, NAME = {name}, TYPE = NODE\n"".format(name=surfname)
    ls=[ln for ln in lines if ln]
    return s+',\n'.join(ls)

def findLines(pts):
    lns=[]
    for p1,p2 in pts:
        ln=findLine(x1=p1[0],y1=p1[1],x2=p2[0],y2=p2[1])
        lns.append(ln)
    return lns

def nodesByLines(lns):
    lnNodes=set()
    for ln in lns:
        if ln: lnNodes.update(lines[ln])
    return lnNodes

def nearestNode(x,y):
    mn=1e30
    for n in nodes:
        xn,yn,zn=nodes[n]
        d=((x-xn)**2+(y-yn)**2)**0.5

```

```

        if d<mn:
            mn=d
            nnode=n
    return nnode,mn

if __name__=='__main__':
    print "ccx_inp"
    elset2nset()
    #reverseOrient()
    parse()
    print nearestNode(11.13, 23.87)

    # for n in nodes:
    #     print n, nodes[n]

    # for n in lines:
    #     print n, lines[n]

    # for n in lines:
    #     print n, [nodes[x] for x in lines[n]]

    # for s in elements:
    #     print s
    #     for e in elements[s]:
    #         #print e,elements[s][e]
    #         print findElementSurface(e)

    # print findLine(0,0,0.8864911064067351,0)
    #
    # print surfByLineEl("Line1")
    #
    # print surfsDefs(["Line1","Line2","Line3"])

    # for e in elements['Surface2']:
    #     print partByElement(e)

    # print partByLine("Line1")

```

Лістинг I.4 – ccx_out.py

```

# -*- coding: utf-8 -*-
"""!Увага! Читає тільки 1 інкремент з напруженнями,
тому застосуйте *TIME POINTS
"""
file=r'model.frd'

def parseLines(startLine):
    "Читає дані з блоку що починається з startLine"
    res={} # напруження для кожного вузла
    inside=False # чи всередині блоку
    with open(file,'r') as f:

```

```

    for ln in f: # для економії пам'яті
        if ln.startswith(startLine): # початок блоку
            inside=True
            if inside: # всередині блоку
                r=parseLine(ln)
                if r!=None: res[r[0]]=r[1]
            if inside and ln==' -3\n': break # кінець блоку
    return res

def parseLine(ln): # парсить рядок з напруженнями вузла
    if not ln.startswith(" -1"): return
    nodeRes=[]
    s=ln.strip()
    for i in range(7): # розбити рядок на 7 частин по 12 символів
        nodeRes.append(s[12*i:12*i+12])
    node=int(nodeRes[0][2:]) # вузол
    r=[float(i) for i in nodeRes[1:]] # значення
    return node,r

def stressMises(Sx,Sy,Sz,Txy,Tyz,Txz):
    "Екв. напруження за критерієм Мізеса"
    # $(1/2**0.5)*((Sx-Sy)**2+(Sy-Sz)**2+(Sz-Sx)**2+6*Txy**2+6*Tyz**2+6*Txz**2)**0.5$ 
    return (Sx*Sx + Sy*Sy + Sz*Sz - Sx*Sy - Sx*Sz - Sy*Sz + 3*Txy*Txy +
3*Txz*Txz + 3*Tyz*Tyz)**0.5

def principalStress(Sx,Sy,Sz,Sxy,Syz,Szx):
    a=((Sx-Sy)**2/4+Sxy**2)**0.5
    S=[(Sx+Sy)/2+a, Sz, (Sx+Sy)/2-a]
    S.sort(reverse=True)
    return S

def principalStress_(Sx,Sy,Sz,Sxy,Syz,Szx):# for test only
    def fn(S,Sx,Sy,Sz,Sxy,Syz,Szx):
        Syz=0.0;Szx=0.0
        # $S**3-(Sx+Sy+Sz)*S**2+(Sx*Sy+Sy*Sz+Sz*Sx-Sxy**2-Syz**2-Szx**2)*S-$ 
         $(Sx*Sy*Sz+2*Sxy*Tyz*Szx-Sx*Tyz**2-Sy*Szx**2-Sz*Sxy**2)$ 
        return S**3-(Sx+Sy+Sz)*S**2+(Sx*Sy+Sy*Sz+Sz*Sx-Sxy**2)*S-(Sx*Sy*Sz-
Szx*Sxy**2)

    from scipy import arange
    from scipy.optimize import fsolve
    S=fsolve(fn, arange(-1000., 1000., 200.), args=(Sx,Sy,Sz,Sxy,Syz,Szx))
    S=list(set([round(i,2) for i in S]))
    S.sort(reverse=True)
    return S

def FOS(S1_2, S1_1, S2_2, S2_1, S3_2, S3_1):
    """Розраховує коефіцієнт запасу втомної міцності за критерієм Сайнса
    S1_2 - головне напруження 1 крок 2 (максимальне навантаження), МПа
    S1_1 - головне напруження 1 крок 1 (мінімальне навантаження)

```

```

"""
sn = 207.0 #границя витривалості
m = 1.0 #коефіцієнт

Sm3 = (S3_2 + S3_1) / 2.
Sa3 = (S3_2 - S3_1) / 2.

Sm2 = (S2_2 + S2_1) / 2.
Sa2 = (S2_2 - S2_1) / 2.

Sm1 = (S1_2 + S1_1) / 2.
Sa1 = (S1_2 - S1_1) / 2.

FOS = (sn - m * (Sm1 + Sm2 + Sm3) / 3.) / (((Sa1 - Sa2) ** 2 + (Sa2 -
Sa3) ** 2 + (Sa3 - Sa1) ** 2) / 2.)*0.5
return FOS

def getResults(nodes):
    "Результати двох кроків для списку вузлів"
    res1=parseLines("      1PSTEP                2                1
1")
    res2=parseLines("      1PSTEP                6                1
2")
    res={}
    for node in nodes:
        SM_1=stressMises(*res1[node])
        S1_1,S2_1,S3_1=principalStress(*res1[node])
        SM_2=stressMises(*res2[node])
        S1_2,S2_2,S3_2=principalStress(*res2[node])
        fos=FOS(S1_2, S1_1, S2_2, S2_1, S3_2, S3_1)
        res[node]=[SM_1,SM_2,fos] # список потрібних результатів
    return res

def minFOSnode(res):
    "вузол з найменшим FOS. res - з getResults"
    minVal=1.e30
    for node in res:
        if res[node][2]<minVal:
            minVal=res[node][2]
            minNode=node
    with open('results.txt','w') as f:
        f.write(str(minVal))
    return minNode,minVal

if __name__=='__main__':
    res=parseLines("      1PSTEP                2                1
1")
    node=1 #res.keys()[0]
    print node,res[node]
    print stressMises(*res[node])
    print principalStress_(*res[node])

```



```

print principalStress(*res[node])
res=getResults([node])
print minFOSnode(res)

```

Лістинг I.5 – main.py

```

#-*- coding: utf-8 -*-
#from __future__ import unicode_literals
import math,sys
from myBaseGeom import *
# Засоби для створення простого GUI
# розкоментувати, якщо потрібна візуалізація
#from OCC.Display.SimpleGui import init_display
#display, start_display, add_menu, add_function_to_menu = init_display()
#display.set_bg_gradient_color(255,255,255,255,255,255) # колір фону

from OCC.BRepBuilderAPI import *
from OCC.TopoDS import *
from OCC.TopExp import *
from OCC.TopAbs import *
from OCC.BRepFilletAPI import BRepFilletAPI_MakeFillet2d
from OCC.ChFi2d import ChFi2d_ChamferAPI,ChFi2d_FilletAPI
from OCC.BRep import BRep_Tool_Pnt
from OCC.BRepAlgoAPI import *
from OCC.BRepAlgo import *
# замість BRepAlgoAPI_Fuse використовувати BRepAlgo_Fuse, щоб результатом
була одна грань
from OCC.BRepBuilderAPI import BRepBuilderAPI_Sewing
from OCC.Precision import precision_Confusion

from gost13877_96params import Rod19
# параметри можуть передаватись через командний рядок:
d={}
for i in sys.argv[1:]: exec i in None,d
d=Rod19(**d)
#d=Rod19(r3n=2.5)
#d=Rod19(d_n=13.12) #12.7,12.84,12.98,13.12,13.26

def translate(f,x1,y1,x2,y2):
    u"""Переміщення з точки в точку"""
    #gp_Trsf2d
    trsf=gp_Trsf()
    trsf.SetTranslation(gp_Pnt(x1,y1,0),gp_Pnt(x2,y2,0)) # переміщення
    f=BRepBuilderAPI_Transform(f,trsf).Shape()
    return f

def rotate(f, angle):
    u"""Поворот навколо осі z на кут angle"""
    trsf=gp_Trsf() # трансформація
    trsf.SetRotation(gp_Ax1(),angle) # поворот
    f=BRepBuilderAPI_Transform(f,trsf).Shape()

```

```

return f

def poly(pts):
    u"""Полігон зі скругленнями і фасками
    Приклад:
    poly(pts=[[ (0,0),1],[ (10,0),(1,1)],[ (5,5),None]])
    тут 1 - радіус скруглення біля вершини,
    (1,1) - фаска біля вершини,
    None або 0 - без фаски чи скруглення біля вершини"""
    gpts=[gp_Pnt(p[0][0],p[0][1],0) for p in pts] # точки вершин
    es=[] # список ребер (без фасок і скруглень)
    p1=gpts[-1] # остання точка
    for p2 in gpts: # для кожної точки
        e=BRepBuilderAPI_MakeEdge(p1,p2).Edge() # створити ребро
        es.append(e) # додати у список
        p1=p2 # попередня точка

    ecs=[] # список містить ребра фаски чи скруглення або None
    for i,p in enumerate(pts): # для кожної точки
        if i==len(pts)-1: j=0 # індекс другого ребра
        else: j=i+1
        pcf=p[1] # розміри фаски чи скруглення
        if type(pcf) in [tuple,list]: # якщо фаска
            ch=ChFi2d_ChamferAPI(es[i],es[j]) # фаска за двома ребрами
            ch.Perform()
            ec=ch.Result(es[i],es[j],pcf[0],pcf[1]) # ребро фаски
            # es[i],es[j] - нові ребра біля фаски
        elif not pcf: # без фаски чи скруглення
            ec=None
        else: # скруглення
            fl=ChFi2d_FilletAPI(es[i],es[j],gp_Pln()) # скруглення за двома
ребрами
            fl.Perform(pcf)
            ec=fl.Result(gpts[i],es[i],es[j]) # ребро скруглення
            # es[i],es[j] - нові ребра біля скруглення
        ecs.append(ec) # додати ребро фаски чи скруглення у список
    en=[] # список усіх остаточних ребер
    for e,ec in zip(es,ecs):
        en.append(e) # додати ребро
        if ec: en.append(ec) # додати ребро фаски чи скруглення, якщо не
None

# # ще один спосіб створення полігону
# mp = BRepBuilderAPI_MakePolygon()
# for p in gpts:
#     mp.Add(p)
# mp.Close()
# w=mp.Wire()

mw=BRepBuilderAPI_MakeWire() # створити контур
for e in en: # для кожного ребра

```

```

        mw.Add(e) # додати в контур
w=mw.Wire() # контур
f=BRepBuilderAPI_MakeFace(w).Face() # грань

# # ще один спосіб створення скруглень
# mf=BRepFilletAPI_MakeFillet2d(f)
# ex = TopExp_Explorer(f, TopAbs_VERTEX)
# hasFillet=False
# while ex.More():
#     v = topods_Vertex(ex.Current())
#     vp=BRep_Tool_Pnt(v)
#     if v.Orientation()==TopAbs_FORWARD:
#         for i in range(len(gpts)):
#             if vp.IsEqual(gpts[i],1e-6) and r[i]!=0:
#                 print vp.X(),vp.Y(),vp.Z()
#                 hasFillet=True
#                 mf.AddFillet(v,r[i])
#         ex.Next()
#     if hasFillet: f=mf.Shape()

return f

def rect(Lx,Ly,c=[0,0,0,0]):
    u"""Прямокутник з першим кутом (нижній лівий) в 0,0
    c - фаски чи скруглення кутів"""
    pts=[[0,0],c[0]],[(Lx,0),c[1]],[(Lx,Ly),c[2]],[(0,Ly),c[3]]
    return poly(pts)

def cut_array(f1,f2,ps):
    u"""Робить масив вирізів поверхню f1 у поверхні f2.
    Список точок вирізів ps=[(0,0),(0,1),(0,2)]"""
    p0=ps[0]
    for p in ps[1:]:
        f2=BRepAlgoAPI_Cut(f2, f1).Shape()
        f1=translate(f1,p0[0],p0[1],p[0],p[1])
        p0=p
    f2=BRepAlgoAPI_Cut(f2, f1).Shape()
    return f2

def thread_points_array(p0,L,s):
    u"""Масив точок для побудови різьби довжиною L та кроком s. Крок може
    бути від'ємний. p0 - початкова точка"""
    ps=[]
    y=p0[1]
    l=0 # поточна довжина різьби
    while l<=L:
        ps.append((p0[0],y))
        y+=s
        l+=abs(s)
    return ps

```

```

def face1():
    u"""Ніпель"""
    h=d.l1n-d.l3n # фаска ніпеля
    f1=rect(d.d_n, d.l1n, [0,(0.577*h,h),0,0]) # ніпель
    f2=rect(d.dn, d.l4n-d.l1n, [0,0,0,0]) # бурт
    f2=translate(f2, 0, 0, 0, d.l1n)
    f3=BRepAlgo_Fuse(f2, f1).Shape()

    f41=rect(d.dn, d.l2n, [0,0,0,0]) # заготовка канавки
    f41=translate(f41, 0, 0, 0, d.l1n-d.l2n)
    f3=BRepAlgo_Fuse(f3, f41).Shape()

    f4=rect(d.dn-d.d1n, d.l2n, [d.r3n,0,0,d.r3n]) # виріз канавки
    f4=translate(f4, 0, 0, d.d1n, d.l1n-d.l2n)
    f5=BRepAlgoAPI_Cut(f3, f4).Shape()

    tn1=math.tan(math.radians(30)) # tan верхнього кута профілю
    tn2=math.tan(math.radians(30)) # tan нижнього кута профілю
    H=2.2 # висота різця
    f6=poly([[0,0],d.r_n],[H,-H*tn2],0],[H,H*tn1],0]) # різець
    f6=translate(f6, 0, 0, -H, 0)
    #display.DisplayShape(f6,update=True,color='red')

    f6=translate(f6, 0, 0, d.dn_, d.ln_)
    a=thread_points_array((d.dn_,d.ln_), d.l1n-d.l2n, -d.p_n)
    f=cut_array(f6,f5,a) # ніпель з різьбою

    #print f.ShapeType() # COMPOUND
    # отримати поверхню
    ex = TopExp_Explorer(f, TopAbs_SHELL)
    f=topods_Shell(ex.Current())
    return f

def face2():
    u"""Муфта"""
    f1=rect(d.dm-d.d1m, d.lm, [0,0,(3,0.8),(1.15,2)]) # муфта
    f1=translate(f1,0,0,d.d1m,-(d.lm-d.l1n))
    h=d.d1m-d.d1_m # фаска
    f2=rect(d.dm-d.d1_m, d.lm-10, [0,0,0,(h,1.73*h)]) # заготовка різьбової
частини
    f2=translate(f2, 0, 0, d.d1_m, -(d.lm-d.l1n))
    f2=BRepAlgo_Fuse(f2, f1).Shape()

    tn1=math.tan(math.radians(30)) # tan верхнього кута профілю
    tn2=math.tan(math.radians(30)) # tan нижнього кута профілю
    H=2.2 # висота різця
    #f3=poly([[0,0],0],[0.3,-0.1],0],[0.3,-0.11],0],[0,-0.21],0])
    f3=poly([[0,H*tn1],0],[H,0),(0.275/0.866, 0.275/0.866)],[(0,-
H*tn2),0]) # різець
    #display.DisplayShape(f3,update=True,color='red')

```

```

f3=translate(f3, 0, 0, d.dm_, d.l2m_)
a=thread_points_array((d.dm_, d.l2m_), d.lm_, -d.p_m)
f=cut_array(f3,f2,a) # муфта з різьбою

#print f.ShapeType() # COMPOUND !!!
# ребро для поділу грані на дві частини
e1=BRepBuilderAPI_MakeEdge(gp_Pnt(d.d1m,d.l1n-5,0),gp_Pnt(d.dm,d.l1n-
5,0)).Edge()

# отримати грань
ex = TopExp_Explorer(f, TopAbs_FACE)
ff=topods_Face(ex.Current())

# розділити грань ребром
from OCC.BRepFeat import BRepFeat_SplitShape
ss=BRepFeat_SplitShape(f)
ss.Add(e1,ff)
f=ss.Shape()
#print f.ShapeType() # COMPOUND

# отримати поверхню
ex = TopExp_Explorer(f, TopAbs_SHELL)
f=topods_Shell(ex.Current())

return f

def mkCompaund(f1,f2):
    u"""Об'єднує форми для експорту в формат BRep"""
    from OCC.BRep import BRep_Builder
    f=TopoDS_Compound()
    bb=BRep_Builder()
    bb.MakeCompound(f)
    bb.Add(f,f1)
    bb.Add(f,f2)
    return f

def findContEdges(f, exPoints):
    u"Шукає усі ребра грані f крім тих, що задані exPoints. Для контактних
    задач. Повертає список ребер у вигляді крайніх точок p1,p2"
    ex = TopExp_Explorer(f, TopAbs_EDGE) # переглядач ребер
    edges=[] # ребра не для контакту як їх крайні точки
    edgesc=[] # ребра для контакту як їх крайні точки
    for p in exPoints:
        x,y=p[1]
        edges.append(findEdge(f,x,y))
    while ex.More():
        e = topods_Edge(ex.Current()) # поточне ребро
        ps=edgeBoundPts(e)
        if (ps[0][0],ps[0][1],ps[1][0],ps[1][1]) not in edges:
            edgesc.append(ps)

```

```

    ex.Next()
    return edgesc

def edgeBoundPts(e):
    u"""Повертає крайні точки ребра"""
    v1=topexp_FirstVertex(e) # перша вершина ребра
    p1=BRep_Tool_Pnt(v1) # точка за вершиною
    v2=topexp_LastVertex(e) # остання вершина ребра
    p2=BRep_Tool_Pnt(v2) # точка за вершиною
    return ((p1.X(),p1.Y()),(p2.X(),p2.Y()))

def drawEdgePts(e):
    u"Рисує крайні точки ребра"
    x1,y1,x2,y2=e
    display.DisplayShape(gp_Pnt(x1,y1,0), color='black')
    display.DisplayShape(gp_Pnt(x2,y2,0), color='black')

def findEdge(f,x,y):
    u"""Шукає ребро поверхні f за точкою на ньому
    Повертає кортеж першої і останньої точок ребер:
    x1,y1,x2,y2"""
    from OCC.BRep import BRep_Tool_Curve
    from OCC.GeomAdaptor import GeomAdaptor_Curve
    from OCC.GeomLib import GeomLib_Tool_Parameter
    px=gp_Pnt(x,y,0)
    #display.DisplayShape(px, color='black')
    ex = TopExp_Explorer(f, TopAbs_EDGE) # переглядач ребер
    while ex.More():
        e = topods_Edge(ex.Current()) # поточне ребро
        c=BRep_Tool_Curve(e) #! зверніть увагу, що результат є кортежем
        gac=GeomAdaptor_Curve(c[0],c[1],c[2]) # крива
(OCC.Geom.Handle_Geom_Curve), перший параметр, другий параметр

        #tp=gac.GetType() # тип кривої 0 - лінія, 1 - коло, ...
        #print type(gac.Line()) # OCC.gp_gp_Lin
        #p = gp_Pnt()
        #gac.D0((c[1]+c[2])/2.0, p) # p - середня точка кривої
        #display.DisplayShape(p, color='black')
        #gac.Line().Contains(px,1e-9) # чи лінія містить точку

        res=GeomLib_Tool_Parameter(c[0],px,1e-9) # ! зверніть увагу, що
        # результат є кортежем. Це не відповідає документації.
        if res[0] and res[1]>=c[1] and res[1]<=c[2]: # якщо точка на кривій
в заданих межах параметрів
            p1=gac.Value(c[1]) # перша точка кривої
            p2=gac.Value(c[2]) # остання точка кривої
            #display.DisplayShape(p1, color='black')
            #display.DisplayShape(p2, color='black')
            return p1.X(),p1.Y(),p2.X(),p2.Y()
    ex.Next()
    return None

```

```

##IsVertexOnLine
##ComputePE
##VertexParameter

def findEdge2(f,x,y):
    u"""Шукає ребро поверхні f за точкою на ньому"""
    from OCC.BRepExtrema import BRepExtrema_ExtPC
    vx=BRepBuilderAPI_MakeVertex(gp_Pnt(x, y, 0)).Vertex() # вершина
    #display.DisplayShape(vx, color='black')
    extr=BRepExtrema_ExtPC() # знаходить відстані від вершини до ребра
    ex = TopExp_Explorer(f, TopAbs_EDGE) # переглядач ребер
    while ex.More(): # для кожного ребра
        e = topods_Edge(ex.Current()) # поточне ребро
        extr.Initialize(e)
        extr.Perform(vx)
        if extr.IsDone(): # якщо екстремуми знайдені
            for i in range(1,extr.NbExt()+1): # для кожного екстремуму
                if extr.SquareDistance(i)<1e-6: # квадрат відстані до
першого екстремуму
                    p=edgeBoundPts(e)
                    return p[0][0],p[0][1],p[1][0],p[1][1]
            ex.Next()
    return None

def drawMesh(es):
    u"""Рисує сітку елементів es"""
    for e in es: # для кожного елемента
        for s in cscx_inp.elements: # для кожної деталі
            # якщо елемент не цієї деталі, то пропустити
            if not e in cscx_inp.elements[s]: continue
            ns=cscx_inp.elements[s][e] # вузли елемента
            ps=[] # точки gp_Pnt
            for n in ns: # для кожного вузла
                x,y,z=cscx_inp.nodes[n] # координати вузла
                ps.append(gp_Pnt(x,y,z)) # додати точку
                display.DisplayShape(ps[-1]) # показати точку
                display.DisplayMessage(ps[-1],str(n),message_color=(0,0,0))
            poly=BRepBuilderAPI_MakePolygon(ps[0],ps[1],ps[2],True).Shape()
            display.DisplayShape(poly) # показати полігон елемента

def visualize(shape):
    u"""Налаштовує параметри візуалізації і показує форму"""
    #from OCC.Display.OCCViewer import Viewer3d
    display.View_Top()
    display.set_bg_gradient_color(255,255,255,255,255,255)
    # сітка
    for x in range(0,50,10):
        for y in range(0,50,10):
            #for z in range(3):
                display.DisplayShape(gp_Pnt(x,y,0))

```

```

display.DisplayShape(shape,update=True)

def saveBrep(shape,filename):
    u"""Зберегти форму у форматі BRep"""
    from OCC.BRepTools import breptools_Write # функція PythonOCC для
запису BRep
    breptools_Write(shape,filename) # зберегти в PythonOCC у форматі BRep

#####

f1=face1() # ніпель
f2=face2() # муфта
f=mkCompaund(f1,f2)
saveBrep(f,"model.brep")
#visualize(f) # візуалізація форми

p={} # словник характерних точок моделі
p[1]=[f2, d.em1[:2]] # нижній торець муфти
p[2]=[f1, d.en1[:2]] # верхній торець ніпеля
p[3]=[f1, d.em4[:2]] # точка на контактній поверхні бурта
p[4]=[f2, d.em4[:2]] # точка на контактній поверхні бурта
p[5]=[f1, d.en2[:2]] # вісь ніпеля
p[6]=[f2, d.em3[:2]] # лінія поділу муфти
p[7]=[f1, d.en3[:2]] # нижній торець ніпеля
p[8]=[f1, d.en4[:2]] # зовнішній циліндр бурта
p[9]=[f2, d.em2[:2]] # #зовнішній циліндр муфти

import ccx_inp
ccx_inp.mesh()
ccx_inp.elset2nset()
lastNode=ccx_inp.appendNode()
#ccx_inp.reverseOrient()
ccx_inp.parse()

def linFinder():
    """Додає лінії у словник характерних точок"""
    for i in p:
        e=findEdge(p[i][0],*p[i][1])
        p[i].append(ccx_inp.findLine(*e))
linFinder()
#print p[6][2] # назва лінії
#drawEdgePts(findEdge(f2,*p[9][1]))

ce1=findContEdges(f=f1, exPoints=[p[7], p[5], p[2], p[8]])
ce2=findContEdges(f=f2, exPoints=[p[9], p[1], p[6]])
master=ccx_inp.findLines(ce1)
#print master # !warning some None
slave=ccx_inp.findLines(ce2)
print len(master), len(set(master)) # чомусь не рівні?
print len(slave), len(set(slave))

```



```

c={}
c["surfSlaveDefinitions"]=ccx_inp.surfDefs(set(slave),"Slave")
#c["surfSlaveDefinitions"]=ccx_inp.surfNodeDefs(set(slave),"Slave")
c["surfMasterDefinitions"]=ccx_inp.surfDefs(set(master),"Master")
c["surfBoltLoadDefinitions"]=ccx_inp.surfByLineEl(p[6][2],"Surface3")
c["surfLoadDefinitions"]=ccx_inp.dloadDefs(p[2][2],"Surface1",d.load2)
c["boundLine1"]=p[1][2]
c["loadLine1"]=p[2][2]
c["boundLine2"]=p[5][2]
c["preTNode"]=lastNode
c["boltLoad"]=d.bolt_load

ccx_inp.writeFinalINP(c)
if not ccx_inp.runCCX(): print "ccx error"
import ccx_out
res=ccx_out.getResults(ccx_inp.nodesByLines(master))
print "FOS=",ccx_out.minFOSnode(res)[1] # мінімальні FOS на поверхні ніпеля
res=ccx_out.getResults(ccx_inp.nodesByLines([p[4][2]]))
print sum([res[n][1] for n in res])/len(res) # середні напруження на бурті
на другому крці

"""
# візуалізація елементів
lnels=set() # множина елементів на контактних лініях
for ln in slave+master:
    if not ln: continue
    lnels.update(set(ccx_inp.elementsByLine(ln)))
#та на лінії BoltLoad
lnels.update(set(ccx_inp.elementsByLine(p[6][2],"Surface2")))
#та на лінії навантаження
lnels.update(set(ccx_inp.elementsByLine(p[2][2],"Surface1")))
drawMesh(lnels)
#drawMesh(ccx_inp.elements['Surface1'])
"""
#start_display()

```

ДОДАТОК Й

Пакет для геометричного моделювання різьб з відхиленнями за допомогою FreeCAD API

Код програм доступний також в GitHub (vkorey/Thread-turning-simulator.

URL: <http://github.com/vkorey/Thread-turning-simulator>). Вміст пакету:

- main.py – головний модуль для побудови геометричної моделі;
- dimsNKT.py – параметри з'єднання гладких НКТ.

Лістинг Й.1 – main.py

```

1  # -*- coding: utf-8 -*-
2  import sys
3  # якщо freecad 64 біт, то python повинен бути 2.7 64 !
4  FREECADPATH = "e:\\FreeCAD 0.17x64\\bin"
5  sys.path.append(FREECADPATH) # шлях до бібліотек FreeCAD
6  import FreeCAD as App # модуль для роботи з програмою
7  import FreeCADGui as Gui # модуль для роботи з GUI
8  import Part # workbench-модуль для створення і керування BRep об'єктами
9  import numpy as np
10 from dims import d
11
12 ##
13 def show(name): # показує форму
14     doc.addObject("Part::Feature",name).Shape=globals()[name]
15
16 def helixPoints(r,h,p,fi,n): # конічна гвинтова лінія
17     "r-менший радіус, h-висота, p-крок, fi - кут (рад), n-кількість
18     точок на витку"
19     a=np.tan(fi)*p/(2*pi)
20     b=p/(2*pi)
21     k=h/p # кількість витків
22     N=int(n*k) # загальна кількість точок
23     t=np.linspace(0,h/b,N) # масив значень t
24     x=(r+a*t)*np.cos(-t) # масив значень x
25     z=(r+a*t)*np.sin(-t) # масив значень y
26     y=b*t # масив значень y
27     # (OY - вісь гвинтової лінії)
28     return zip(t,x,y,z)
29
30 def perror(points):
31     epoints=[] # точки з відхиленнями
32     d=0.2 # параметр трикутного розподілу
33     for t,x,y,z in points:
34         x=x+np.random.triangular(-d,0,d)

```

```

34         y=y+np.random.triangular(-d,0,d)
35         z=z+np.random.triangular(-d,0,d)
36         epoints.append((t,x,y,z))
37     return epoints
38
39 def rebuildSketch(dim, sk):
40     "Перебудовує ескіз і повертає грань. dim - розміри ескізу sk"
41     doc=App.getDocument("Sketches") # об'єкт документа
42     sketch=doc.__getattr__(sk) # ескіз або doc.<назва ескізу>
43     # для кожної пари (обмеження ескізу ID, значення розміру)
44     for k,v in dim.iteritems():
45         sketch.setDatum(k,v) # установити значення розміру
46     doc.recompute() # перебудувати документ
47     w=sketch.Shape.Wires[0] # отримати перший цикл
48     f=Part.Face(w) # створити грань
49     return f
50
51 def revolve(f):
52     """Створює тіло шляхом обертання грані f навколо осі Y"""
53     return f.revolve(App.Vector(0,0,0), App.Vector(0,1,0)) # тіло
54
55 def helix(r,h,p,fi):
56     "Гвинтова лінія. r-радіус, h-висота, p-крок, fi-кут конуса (рад.)"
57     w=Part.makeLongHelix(p,h,r,np.degrees(fi)) # !!! але не makeHelix
58     #FreeCAD-master\src\Mod\Part\App\TopoShape.cpp
59     w.rotate(App.Vector(0,0,0),App.Vector(1,0,0),-90)
60     return w
61
62 def makeThread(f,h,s): # різьба ніпеля/муфти
63     s2=h.makePipeShell(f.Wires,True,True) # тіло спіралі; дозволяє
64     кілька профілів!
65     s=s.cut(s2) # булева операція вирізання
66     return s
67
68 def helix2(points): # гвинтова лінія з відхиленнями
69     pts=[(x,y,z) for t,x,y,z in points]
70     h=Part.makePolygon(pts) # полінія
71     return h
72
73 def makeThread2(f,h,s): # різьба з відхиленнями
74     w=f.Wires[0] # цикл інструмента
75     points=[(v.X,v.Y,v.Z) for v in h.Vertexes] # точки на гвинт. лінії
76     W=[] # профілі в різних точках лінії
77     for x,y,z in points: # для кожної точки
78         w_=w.copy() # копія циклу інструмента
79         t=np.arctan2(-z,x) # кут повороту різця відносно заготовки
80         w_.rotate(App.Vector(0,0,0),App.Vector(0,1,0),np.degrees(t)) #
81     повернути інструмент навколо осі Y
82     w_.translate(App.Vector(x,y,z)) # перемістити інструмент
83     W.append(w_.copy()) # додати профіль до списку W
84     if len(W)>1:

```

```

82         st=Part.makeLoft([W[-2],W[-1]],True) # створити тіло за
двома сусідніми профілями
83         s=s.cut(st) # виріз в заготовці тілом st
84
85     return s,W
86
87 #S={} # форми
88 print "OCC", Part.OCC_VERSION
89 pi=np.pi
90 App.open(u"D:/3/Sketches.FCStd") # відкрити документ
91
92 ## ніпель
93 fA0=rebuildSketch(dim={6:d.fi, 9:d.H}, sk='Sketch') # грань інструмента
94 fA0.translate(d._v2) # перемістити інструмент в поч. позицію
95 sA0=revolve(rebuildSketch(dim={20:d.fi, 16:d.d3/2}, sk='Sketch001')) #
тіло заготовки
96 hA0=helix(r=d._r, h=d.l4-12, p=d.P, fi=d.fi) # гвинтова лінія
97 hA0.translate((0,d.l3-d.l4,0)) # перемістити лінію
98 sA1=makeThread(fA0, hA0, sA0) # ніпель з різьбою
99
100 ## муфта
101 fB2=rebuildSketch(dim={10:(pi-d.fi), 9:d.H}, sk='Sketch002') # tool
102 fB2.translate(d._v2)
103 #f2.rotate(App.Vector(0,0,0),App.Vector(0,1,0),1)
104 sB2=revolve(rebuildSketch(dim={28:d.fi, 17:d.d3/2}, sk='Sketch003'))
105 hB2=helix(r=d._r, h=d.l4, p=d.P, fi=d.fi)
106 hB2.translate((0,d.l3-d.l4,0))
107 sB3=makeThread(fB2, hB2, sB2)
108
109 ## ніпель без відхилень
110 fA3=rebuildSketch(dim={6:d.fi, 9:d.H}, sk='Sketch') # tool
111 fA3.translate((-d.H/2+np.tan(d.fi)*d.P/2, d.P/2, 0))
112 points=helixPoints(r=d._r, h=d.l4-12, p=d.P, fi=d.fi, n=50)
113 hA3=helix2(points)
114 hA3.translate((0,d.l3-d.l4,0))
115 sA3,W=makeThread2(fA3, hA3, sA0)
116
117 ## ніпель з відхиленнями
118 fA3.rotate(App.Vector(0,0,0),App.Vector(0,1,0),-10) # утворити γ
119 points=perror(points) # точки гвинт лінії з відхиленнями
120 hA3=helix2(points) # гвинтова лінія з відхиленнями
121 hA3.translate((0,d.l3-d.l4,0)) # перемістити
122 sA4,W=makeThread2(fA3, hA3, sA0) # ніпель з різьбою
123
124 # булеві операції над плоскими перерізами
125 f=Part.makePlane(400,400,App.Vector(-200,-200,0),App.Vector(0,0,1)) #
площина XY
126 f1=sA3.common(f).Faces[0] # осьовий перетин ніпеля
127 f2=sA4.common(f).Faces[0] # осьовий перетин муфти
128 f3=f1.cut(f2).fuse(f2.cut(f1)) # операція XOR
129 print f3.Area # XOR-площа

```

```

130
131 ##
132 # Наступні команди потрібні тільки для візуалізації створених форм
133 Gui.showMainWindow() # показати головне вікно
134 Gui.activateWorkbench("PartWorkbench")
135 doc=App.newDocument() # створити новий документ
136 # показати форми
137 show('fA0')
138 show('sA1')
139 show('sB3')
140 show('fA3')
141 show('hA3')
142 show('sA3')
143 show('sA4')
144 show('f1')
145 show('f2')
146 show('f3')
147 for w in W[:10]:
148     Part.show(w)
149
150 doc.recompute() # перебудувати
151 Gui.exec_loop() # головний цикл програми

```

Лістинг Й.2 – dimsNKT.py

```

# -*- coding: utf-8 -*-
from math import atan, degrees, tan

class Dim:
    "Клас описує поняття розміру"
    n=0.0 # номінальний розмір
    ei=0.0 # нижнє відхилення
    es=0.0 # верхнє відхилення
    v=0.0 # дійсне значення
    def __init__(self,n,ei,es,doc):
        "конструктор"
        self.v=n
        self.n=n
        self.ei=ei
        self.es=es
        self.__doc__=doc.decode('utf-8')
    def min(self):
        "повертає мінімальний розмір"
        return self.n+self.ei
    def max(self):
        "повертає максимальний розмір"
        return self.n+self.es

nkt114={'D':Dim(114.3,0.0,0.0,"зовнішній діаметр труби"),
        'd':Dim(100.3,0.0,0.0,"внутрішній діаметр труби"),

```

```

'Dm':Dim(132.1,0.0,0.0,"зовнішній діаметр муфти"),
'Ln':Dim(156.0,0.0,0.0,"довжина муфти*"),
'P':Dim(3.175,0.0,0.0,"крок різьби паралельно осі різьби"),
'dsr':Dim(112.566,0.0,0.0,"середній діаметр різьби в основній
площині"),
'd1':Dim(111.031,0.0,0.0,"зовнішній діаметр різьби в площині торця
труби"),
'd2':Dim(107.411,0.0,0.0,"внутрішній діаметр різьби в площині торця
труби"),
'L':Dim(65.0,-3.2,3.2,"загальна довжина різьби труби"),
'l':Dim(52.3,0.0,0.0,"довжина різьби труби до основної площини (з
повним профілем)"),
'l1':Dim(10.0,0.0,0.0,"максимальна довжина збігу різьби труби"),
'd3':Dim(111.219,0.0,0.0,"внутрішній діаметр різьби в площині торця
муфти"),
'd0':Dim(115.9,0.0,0.8,"діаметр циліндричної виточки муфти"),
'l0':Dim(9.5,-0.5,1.5,"глибина виточки муфти"),
'A':Dim(6.5,0.0,0.0,"натяг при згвинчуванні вручну"),
'fi':Dim(atan(1.0/32),0.0,0.0,"кут нахилу (рад)"),
'H':Dim(2.75,0.0,0.0,"висота вихідного профілю"),
'h1':Dim(1.81,-0.1,0.05,"висота профілю різьби"),
'h':Dim(1.734,0.0,0.0,"робоча висота профілю"),
'alfa_2':Dim(30.0,-1.0,1.0,"кут нахилу сторони профілю alfa/2"),
'r':Dim(0.508,0.0,0.045,"радіус заокруглення вершини профілю"),
'r1':Dim(0.432,-0.045,0.0,"радіус заокруглення впадини профілю")}

nkt33={'D':Dim(33.4,0.0,0.0,"зовнішній діаметр труби"),
'd':Dim(22.6,0.0,0.0,"внутрішній діаметр труби"),
'Dm':Dim(42.2,0.0,0.0,"зовнішній діаметр муфти"),
'Ln':Dim(84.0,0.0,0.0,"довжина муфти*"),
'P':Dim(2.54,0.0,0.0,"крок різьби паралельно осі різьби"),
'dsr':Dim(32.065,0.0,0.0,"середній діаметр різьби в основній площині"),
'd1':Dim(32.382,0.0,0.0,"зовнішній діаметр різьби в площині торця
труби"),
'd2':Dim(29.568,0.0,0.0,"внутрішній діаметр різьби в площині торця
труби"),
'L':Dim(29.0,-2.5,2.5,"загальна довжина різьби труби"),
'l':Dim(16.3,0.0,0.0,"довжина різьби труби до основної площини (з
повним профілем)"),
'l1':Dim(8.0,0.0,0.0,"максимальна довжина збігу різьби труби"),
'd3':Dim(31.210,0.0,0.0,"внутрішній діаметр різьби в площині торця
муфти"),
'd0':Dim(35.0,0.0,0.8,"діаметр циліндричної виточки муфти"),
'l0':Dim(8.0,-0.5,1.5,"глибина виточки муфти"),
'A':Dim(5.0,0.0,0.0,"натяг при згвинчуванні вручну"),
'fi':Dim(atan(1.0/32),0.0,0.0,"кут нахилу (рад)"),
'H':Dim(2.2,0.0,0.0,"висота вихідного профілю"),
'h1':Dim(1.412,-0.1,0.05,"висота профілю різьби"),
'h':Dim(1.336,0.0,0.0,"робоча висота профілю"),
'alfa_2':Dim(30.0,-1.0,1.0,"кут нахилу сторони профілю alfa/2"),
'r':Dim(0.432,0.0,0.045,"радіус заокруглення вершини профілю"),

```

```

    'r1':Dim(0.356,-0.045,0.0,"радіус заокруглення впадини профілю"}}

D=nkt114 #nkt33 вибрати типорозмір
class Dims(object): pass
d=Dims() # атрибутами є об'єкти класу Dim, що зручніше ніж словник D
for key,value in D.iteritems():
    setattr(d,key,value)

# дійсні розміри ескіза:
"""
d.D.v=d.D.n/2
d.d.v=d.d.max()/2
d.Dm.v=d.Dm.min()/2
d.Lm.v=d.Lm.n/2
d.P.v=d.P.n
d.dsr.v=d.dsr.n/2
d.d1.v=d.d1.n/2
d.d2.v=d.d2.n/2
d.L.v=d.L.min()
d.l.v=d.l.n
d.l1.v=d.l1.n
d.d3.v=d.d3.n/2
d.d0.v=d.d0.min()/2
d.l0.v=d.l0.max()
d.A.v=d.A.n
d.fi.v=d.fi.n
d.H.v=d.H.n
d.h1.v=d.h1.max()
d.h.v=d.h.n
d.alfa_2.v=d.alfa_2.n
d.r.v=d.r.max()
d.r1.v=d.r1.min()
"""
d.D.v=d.D.n/2
d.d.v=d.d.n/2
d.Dm.v=d.Dm.n/2
d.Lm.v=d.Lm.n/2
d.P.v=d.P.n
d.dsr.v=d.dsr.n/2
d.d1.v=d.d1.n/2
d.d2.v=d.d2.n/2
d.L.v=d.L.n
d.l.v=d.l.n
d.l1.v=d.l1.n
d.d3.v=d.d3.n/2
d.d0.v=d.d0.n/2
d.l0.v=d.l0.n
d.A.v=d.A.n
d.fi.v=d.fi.n
d.H.v=d.H.n
d.h1.v=d.h1.n

```

```
d.h.v=d.h.n
d.alfa_2.v=d.alfa_2.n
d.r.v=d.r.max()
d.r1.v=d.r1.min() #0.3 - для FEA без радіусів при вершині
```

```
# приклад використання:
```

```
#print d.D.v
#print d.D.n
#print d.D.min()
#print d.D.__doc__
```

```
##
```

```
# Або доступ до дійсного значення без атрибута v:
```

```
class F(float, Dim): pass
d=Dims() # атрибутами є дійсні розміри ескізу
for key,value in D.iteritems():
    setattr(d,key,F(value.v))
    getattr(d,key).v=value.v
    getattr(d,key).n=value.n
    getattr(d,key).ei=value.ei
    getattr(d,key).es=value.es
    getattr(d,key).__doc__=value.__doc__
```

```
# приклад використання:
```

```
#print d.D
#print d.D.n
#print d.D.min()
#print d.D.__doc__
```

```
# допоміжні параметри:
```

```
# Увага! не враховано відхилення h1
```

```
d.tanFi=tan(d.fi) # тангенс кута нахилу (зміна радіуса конуса на одиницю довжини)
```

```
d._l=d.L-d.A # відстань від 0 до верхнього торця муфти
```

```
d._l2=d.Lm-d._l # відстань від 0 до нижнього торця муфти
```

```
d._r=d.dsr-d.tanFi*(d._l+d._l2) # радіус середнього діаметра в нижньому торці муфти
```

```
x1,y1 = -d.H/2, 0 # вектор переміщення в 0,0 сер. діам. різця ніпеля
```

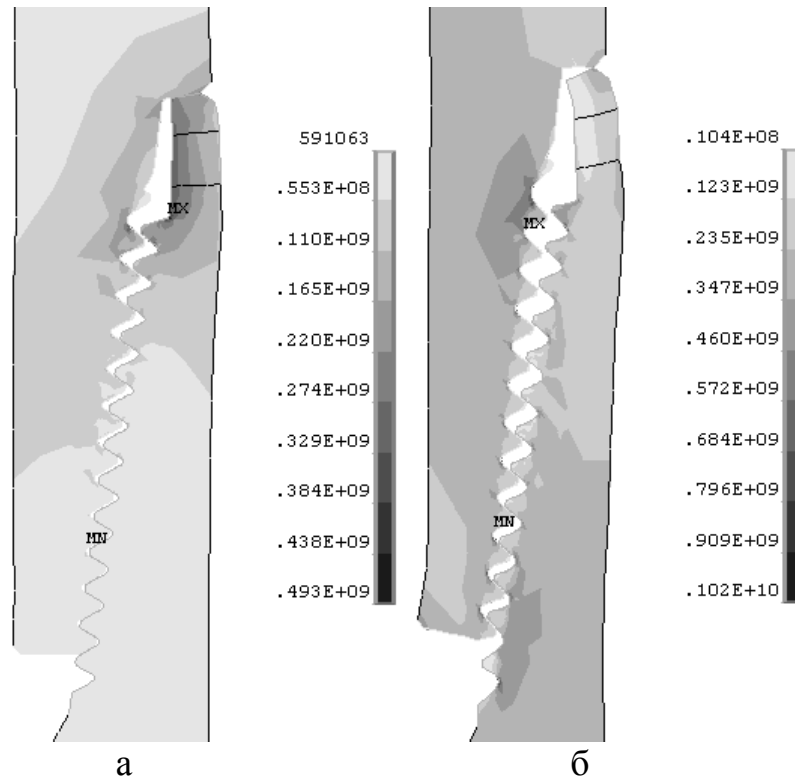
```
x2,y2 = -d.H/2+d.tanFi*d.P/2, d.P/2 # -//- муфти
```

```
d._v1 = x1+d._r, y1-d._l2, 0 # поч. положення різця ніпеля
```

```
d._v2 = x2+d._r, y2-d._l2, 0 # поч. положення різця муфти
```

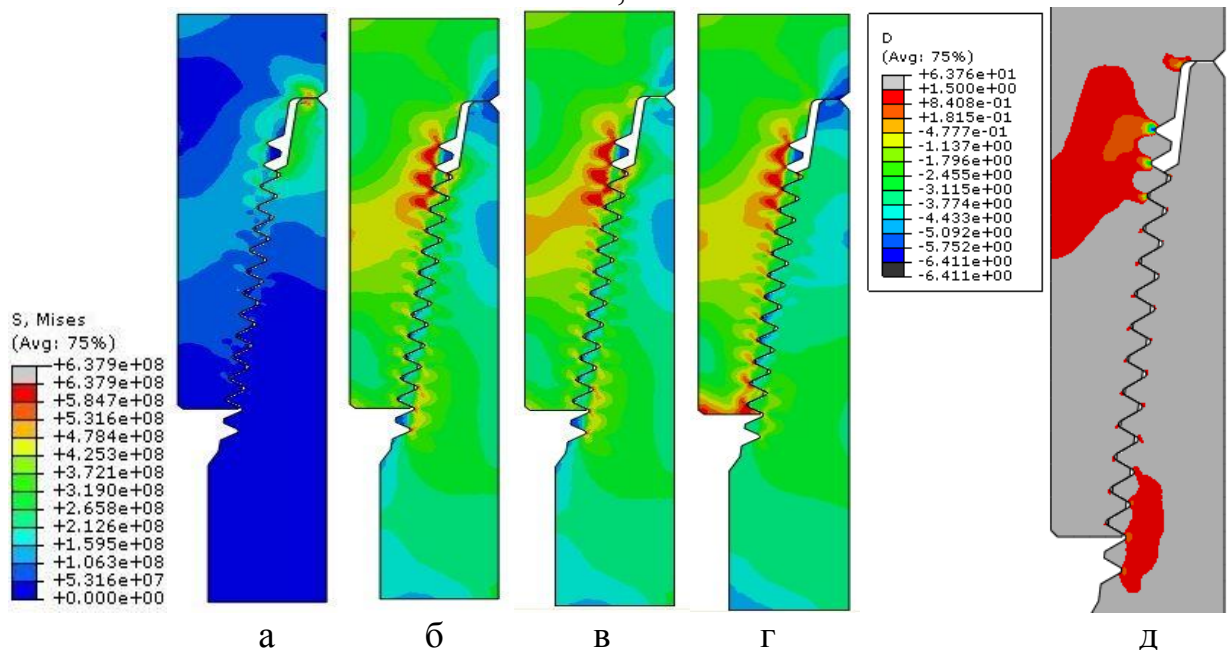

ДОДАТОК К

Результати моделювання замкового РЗ



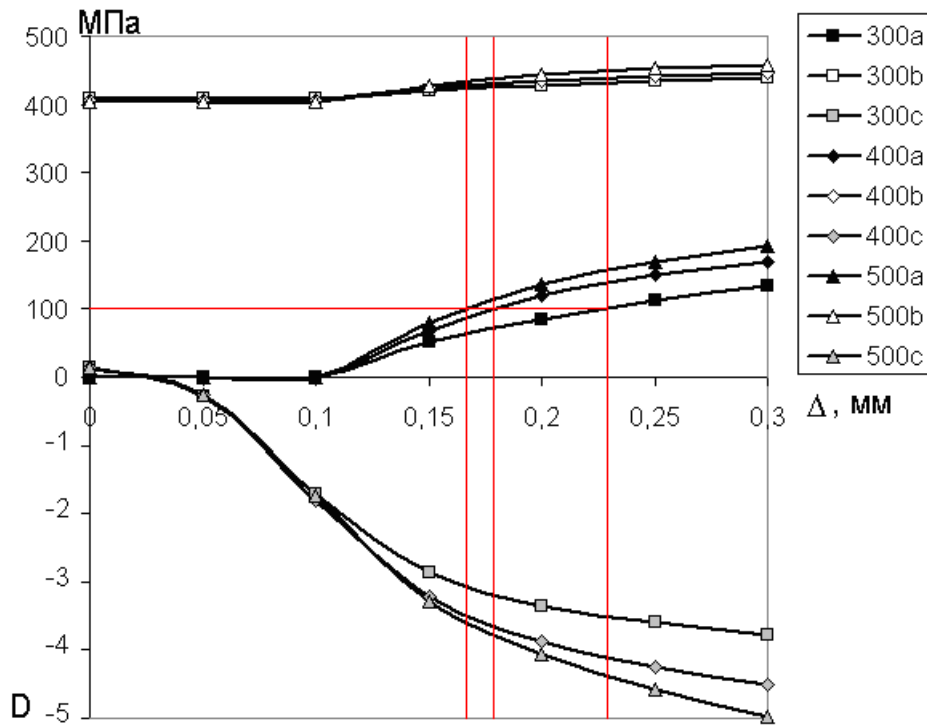
а) – навантаження 0 Н; б) – навантаження 1 МН

Рисунок К.1 – Розподіл напружень σ_m (Па) в пружній моделі РЗ 3-66 замка ЗН-80 з $\Delta=0,1$ мм



а – навантаження 0 МН, $\Delta=0,1$ мм; б – навантаження 1 МН, $\Delta=0,1$ мм; в – навантаження 1 МН, $\Delta=0,2$ мм; г – навантаження 1 МН, $\Delta=0,1$ мм, муфта з пластичнішого матеріалу

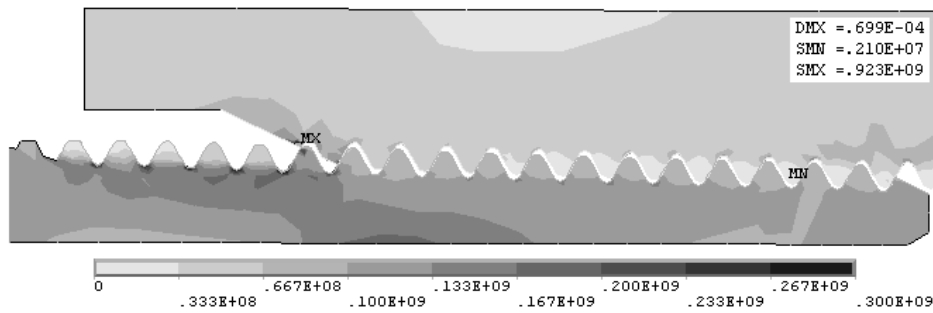
Рисунок К.2 – Розподіл напружень σ_m (Па) та коефіцієнта D для циклу навантаження $\sigma_p=0\dots155$ МПа (д) в пружно-пластичній моделі РЗ 3-66 замка ЗН-80



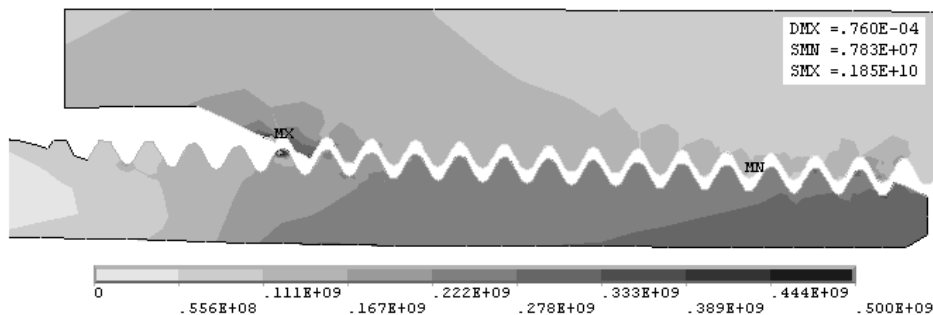
a – величина контактного тиску в місці стику під час максимального навантаження; b – напруження σ_m в першій робочій западині різьби ніпеля під час максимального навантаження; c – коефіцієнт запасу втомної міцності D в першій робочій западині різьби ніпеля

Рисунок К.3 – Діаграма для визначення оптимальної величини згвинчування Δ (мм) РЗ замка ЗН-80 для циклу навантаження $\sigma_p=0\dots155$ МПа і значень σ_t матеріалу муфти 300, 400 і 500 МПа

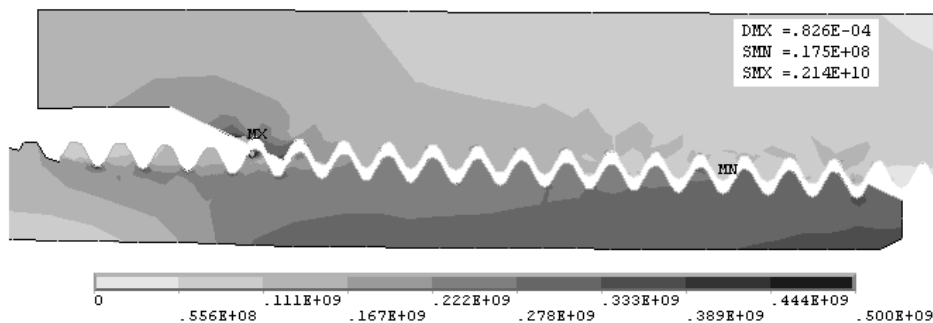
ДОДАТОК Л
Результати моделювання РЗ гладких НКТ умовним діаметром 114 мм (ГОСТ 633-80) з пружною моделлю матеріалу



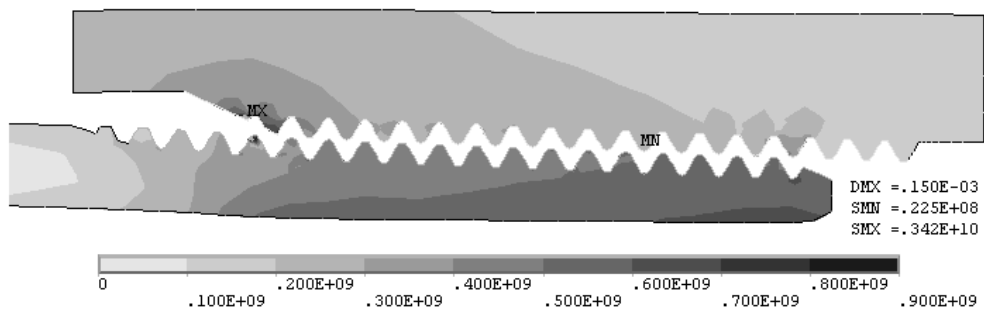
а



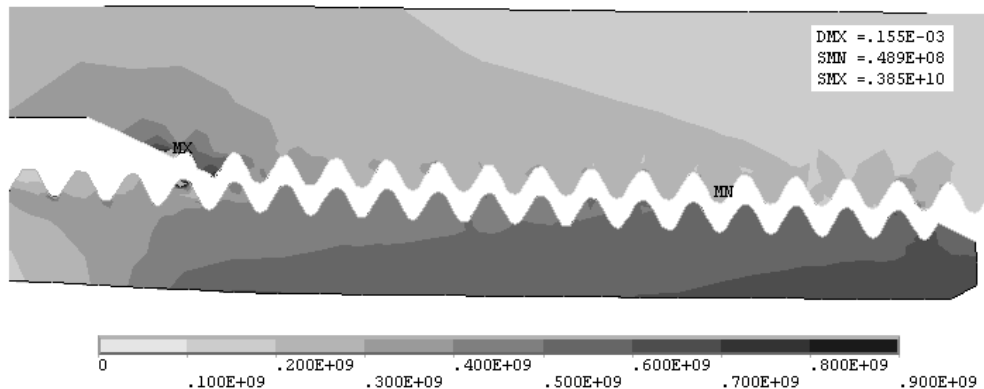
б



в

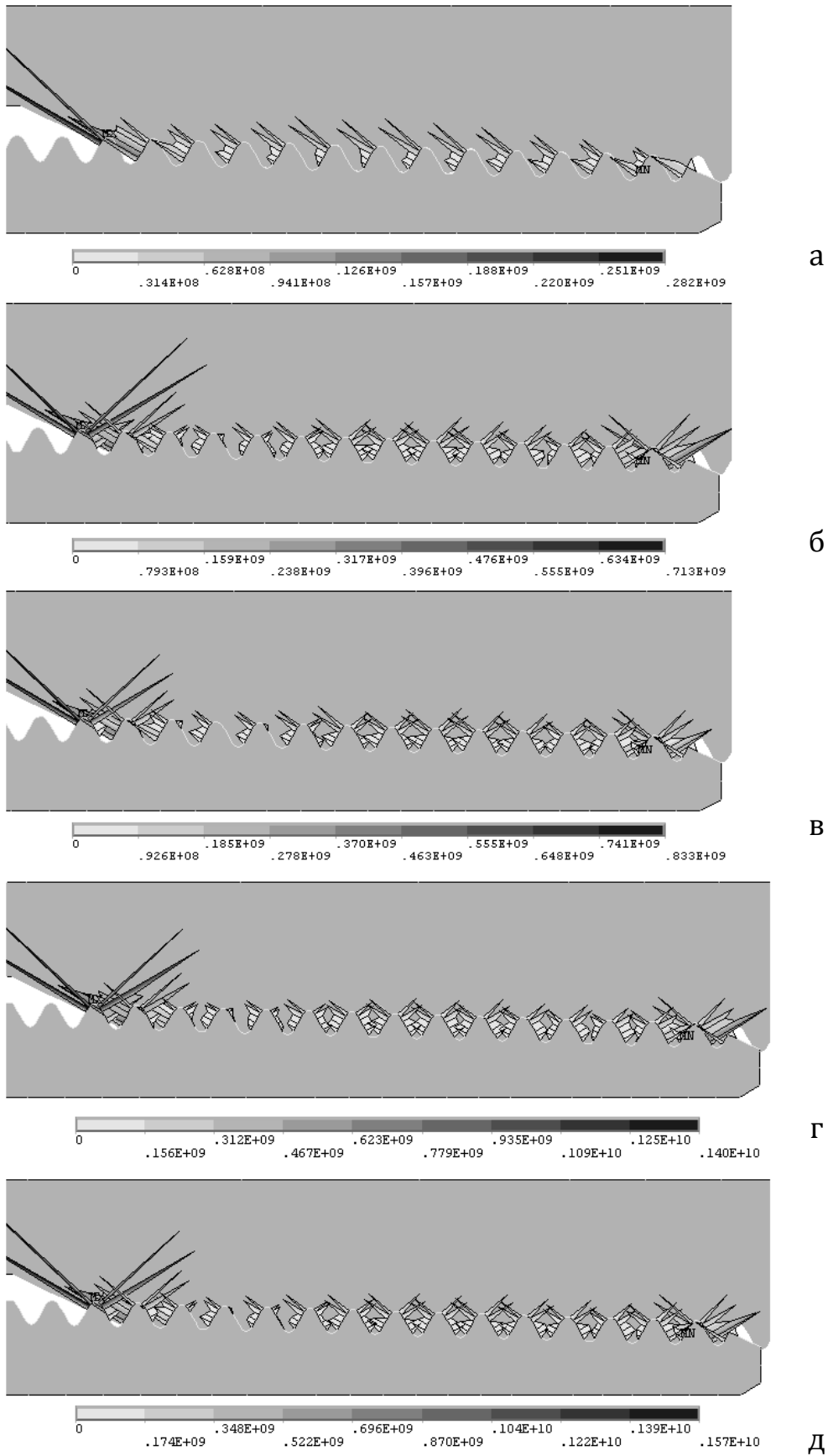


г



д

а) - $A=6,35$ мм; б, в) - $A=3,175$ мм; г, д) - $A=0$ мм; б, г) - $\sigma_p=0$ МПа; а, в, д) - $\sigma_p=100$ МПа
 Рисунок Л.1 – Розподіл σ_M (Па) в муфтовому РЗ НКТ



а) - $A=6,35$ мм; б, в) - $A=3,175$ мм; г, д) - $A=0$ мм;
 б, г) - $\sigma_p=0$ МПа; а, в, д) - $\sigma_p=100$ МПа

Рисунок Л.2 – Значения контактного тиску на витках РЗ НКТ

ДОДАТОК М

Input-файли Abaqus для відтворення гармонічного і динамічного аналізу

Лістинг М.1 – Частина Input-файлу Abaqus для відтворення гармонічного аналізу

```

**Властивості матеріалу
*Material, name=Material-1
*Density
  7.8e-09,
*Elastic
210000., 0.28
** Залежність напруження-деформація (табл.1)
*Plastic
314.,      0.
  349., 0.0013
...
** Властивості контактної взаємодії, коефіцієнт тертя
*Surface Interaction, name=IntProp-1
1.,
*Friction, slip tolerance=0.005
  0.06,
*Surface Behavior, pressure-overclosure=HARD
** Контактні взаємодії поверхонь
*Contact Pair, interaction=IntProp-1, small sliding, type=SURFACE TO
SURFACE
_PickedSurf50, _PickedSurf49
** Перший крок навантажування - згвинчування
*Step, name=Step-1, nlgeom=YES
*Static
1., 1., 1e-05, 1.
** Гранична умова на нижньому торці муфти  $U_y=U_{R_x}=U_{R_z}=0$ 
*Boundary
_PickedSet54, YSYMM
** Навантаження - тиск на верхньому торці -1 МПа
*Dload
_PickedSurf53, P, -1.
** Зменшувати перекриття контактних поверхонь для імітації згвинчування
*Contact Interference, shrink
_PickedSurf50, _PickedSurf49
** Виведення результатів
*Restart, write, frequency=0
*Output, field, variable=PRESELECT
*Output, history, variable=PRESELECT
*End Step
** Другий крок - статичний осьовий розтяг
*Step, name=Step-2, nlgeom=YES
*Static
1., 1., 1e-05, 1.

```

```

** Навантаження - тиск на верхньому торці -155.1 МПа
*Dload
_PickedSurf53, P, -155.1
** Виведення результатів
*Restart, write, frequency=0
*Output, field, variable=PRESELECT
*Output, history, variable=PRESELECT
*End Step
** Третій крок - гармонічний аналіз в діапазоні 10000..20000 Гц
*Step, name=Step-3, nlgeom=NO, perturbation
*Steady State Dynamics, direct, real only, frequency scale=LINEAR, friction
damping=NO
10000., 20000., 500, 1.
** Гармонічне навантаження - дійсна частина тиску на верхньому торці -10
МПа
*Dload, real
Surf-4, P, -10.
** Виведення результатів
*Output, field, variable=PRESELECT
*Output, history, variable=PRESELECT
*End Step

```

Лістинг М.2 – Частина Input-файлу Abaqus для відтворення неявного динамічного аналізу

```

** Опис гармонічного навантаження частотою 55000 рад/с, амплітудою -10 МПа
і середнім значенням -155.1 МПа
*Amplitude, name=Amp-1, definition=PERIODIC
1,          55000.,          0.,          -155.1
           0.,          -10.
** Крок тривалістю 0,005с неявного динамічного аналізу з прямим
інтегруванням
*Step, name=Step-3, nlgeom=YES, inc=1000
*Dynamic,direct,initial=NO
5e-06,0.005,
** Навантаження - змінний тиск на верхньому торці за законом Amp-1
*Dload, op=NEW, amplitude=Amp-1
Surf-4, P, 1.
*Dload, op=NEW
*Restart, write, frequency=0
*Output, field, variable=PRESELECT, frequency=1
*Output, history, variable=PRESELECT, frequency=1
*End Step

```

ДОДАТОК Н

VBA-макрос для обчислення коефіцієнта запасу втомної міцності за критерієм Сайнса в SOLIDWORKS Simulation

Код доступний також в GitHub (vkopey/FOS: VBA macro for SOLIDWORKS API for fatigue safety factor calculation by Sines' method. URL: <http://github.com/vkopey/FOS>).

Лістинг Н.1 – calcFOS.bas

```

Attribute VB_Name = "Macro11"
'Необхідно додати в Tools/References... SolidWorks Simulation 2018 type
library
Dim swApp As Object
Dim COSMOSWORKS As Object
Dim CWorkObject As CosmosWorksLib.CwAddinCallback
Dim ActDoc As CosmosWorksLib.CWModelDoc
Dim StudyMgr As CosmosWorksLib.CWStudyManager
Dim Study As CosmosWorksLib.CWStudy
Dim errCode As Long 'код помилки
Dim CWResult As CosmosWorksLib.cwResults
'Dim LBCMgr As CosmosWorksLib.CWLoadsAndRestrainsManager
'Dim lr As CosmosWorksLib.CWLoadsAndRestrains
Dim Face As SldWorks.Face2

' Розраховує коефіцієнт запасу втомної міцності за критерієм Сайнса
' S1_2 - головне напруження 1 крок 2 (максимальне навантаження), МПа
' S1_1 - головне напруження 1 крок 1 (мінімальне навантаження)
Public Function FOS(S1_2, S1_1, S2_2, S2_1, S3_2, S3_1 As Double)
    sn = 207 '000000 ' границя витривалості
    m = 1 ' коефіцієнт

    Sm3 = (S3_2 + S3_1) / 2
    Sa3 = (S3_2 - S3_1) / 2

    Sm2 = (S2_2 + S2_1) / 2
    Sa2 = (S2_2 - S2_1) / 2

    Sm1 = (S1_2 + S1_1) / 2
    Sa1 = (S1_2 - S1_1) / 2

    FOS = (sn - m * (Sm1 + Sm2 + Sm3) / 3) / Sqr((((Sa1 - Sa2) ^ 2 + (Sa2 -
Sa3) ^ 2 + (Sa3 - Sa1) ^ 2) / 2)
End Function

Sub main()

```

```

Dim s1(1 To 2), s2(1 To 2), s3(1 To 2) As Double ' головні напруження
Set swApp = Application.SldWorks ' об'єкт Solidworks
Set CWorkObject = swApp.GetAddInObject("SldWorks.Simulation") ' об'єкт
Simulation
Set COSMOSWORKS = CWorkObject.COSMOSWORKS
Set ActDoc = COSMOSWORKS.ActiveDoc() ' активний документ COSMOSWORKS
Set StudyMngr = ActDoc.StudyManager() ' менеджер задач
Set Part = swApp.ActiveDoc ' активна деталь

For Each N In Array(6313, 6334, 6349, 198, 186) ' вузол
For i = 1 To 2
StudyMngr.ActiveStudy = i - 1
Set Study = StudyMngr.GetStudy(i - 1) ' задача

' Study.MeshAndRun
'errCode = Study.CreateMesh(0, 4.7, 0.25) ' створити сітку
'runError = Study.RunAnalysis ' виконати задачу
Set CWResult = Study.results ' результати

MaxStep = CWResult.GetMaximumAvailableSteps()
sn = CWResult.GetStress(0, MaxStep, Nothing, 3, errCode) ' масив напружень
в вузлах

s1(i) = sn((N - 1) * 12 + 7) ' перше головне напруження (МПа)
s2(i) = sn((N - 1) * 12 + 8) ' друге
s3(i) = sn((N - 1) * 12 + 9) ' третє
'Debug.Print i, s1(i), s2(i), s3(i)
Next i
Debug.Print N, FOS(s1(2), s1(1), s2(2), s2(1), s3(2), s3(1))
Next N

End Sub

```


ДОДАТОК О

Варіанти конструкції пресового з'єднання полімерного стержня зі сталеву оболонкою

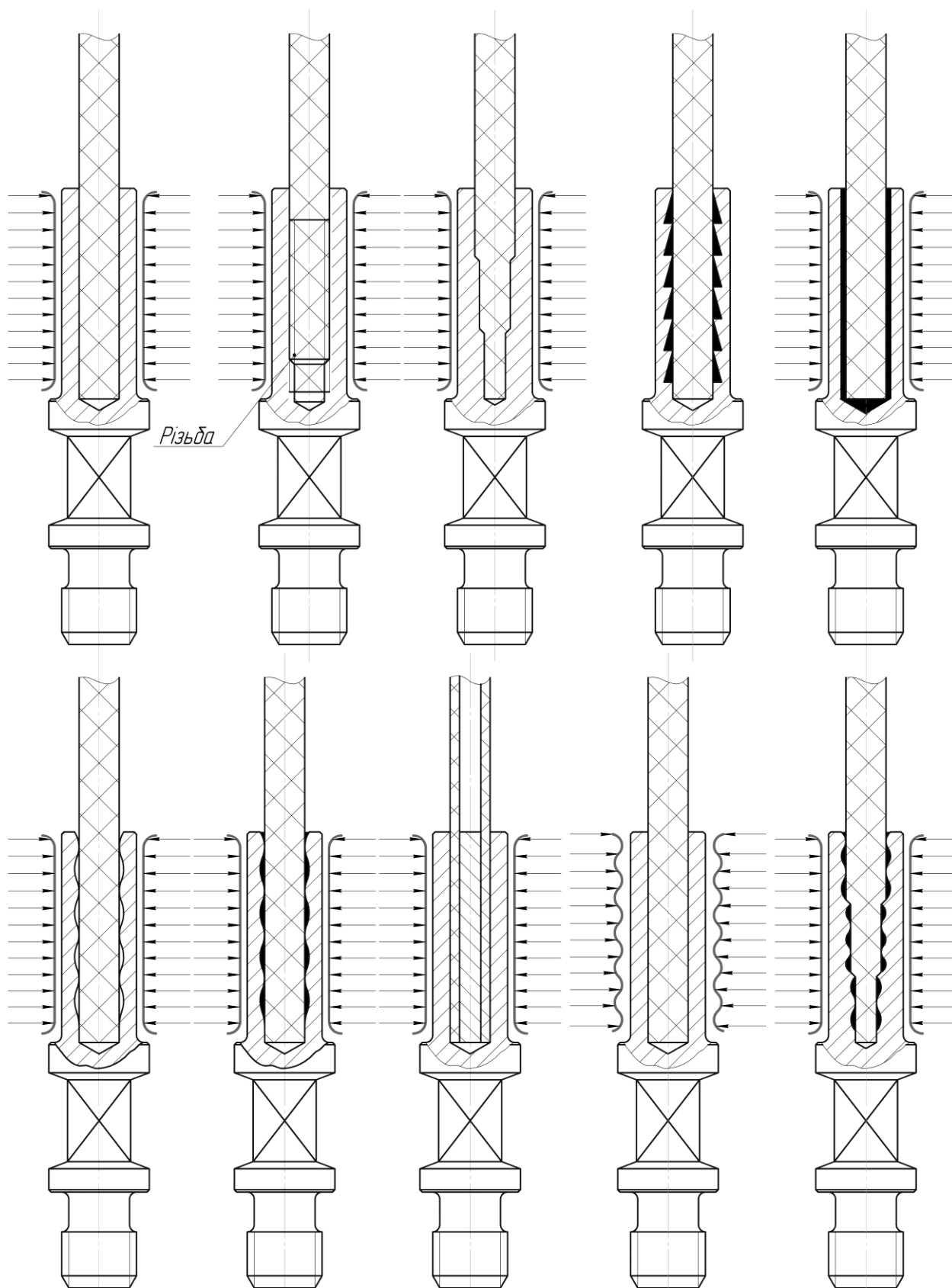


Рисунок О.1 – Варіанти конструкції з'єднання

ДОДАТОК П

**Макрос Abaqus/CAE для розрахунку з'єднання полімерного стержня зі
сталевою оболонкою**

Код доступний також в GitHub (vkorey/Fiberglass-Sucker-Rod-Connection:
FEM model of fiberglass sucker rod connection. URL:
<http://github.com/vkorey/Fiberglass-Sucker-Rod-Connection>).

Лістинг П.1 – main.py

```
# -*- coding: cp1251 -*-
'''Макрос Abaqus CAE для розрахунку з'єднання полімерного стержня зі
сталевою оболонкою'''
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

class Material:
    '''Клас описує поняття матеріалу
    В Abaqus визначається істинна діаграма деформування (див. Stress and
    strain measures)
    E - модуль пружності, Па
    mu - коефіцієнт Пуассона
    st - границя текучості, Па
    et - деформація для st
    sb - істинна границя міцності, Па (sv - умовна границя)
    eb - істинна деформація, яка відповідає границі міцності
    delta - відносне видовження
    psi - відносне звуження
    '''
    def __init__(self,E,mu,st,sv,delta,psi):
        '''конструктор'''
        self.E=E # модуль пружності
        self.mu=mu # коефіцієнт Пуассона
        self.st=st # границя текучості
        self.et=st/E # деформація для st
        self.delta=delta/100.0 # відносне видовження після розриву
```

```

self.psi=psi/100.0 # відносне звуження після розриву
k=0.4 # коефіцієнт(eb=(0.1...0.4,0.2...0.8)delta)
self.sv=sv # границя міцності
self.sb=sv*(1+k*self.delta) # істинна границя міцності
self.eb=log(1+k*self.delta) # істинна деформація, яка відповідає
границі міцності
# істинне напруження і деформація в момент руйнування
self.sk=0.8*self.sv/(1-self.psi) # 0.8-коефіцієнт руйнуючого
навантаження
#self.sk=self.sv*(1+1.35*self.psi)
self.ek=log(1/(1-self.psi))
def bilinear(self):
    '''Повертає словник еластичних і пластичних властивостей'''
    return {'el':((self.E,self.mu),),
            'pl':((self.st,0.0), # білінійна залежність
                  (self.sb,self.eb))} # або (self.sk,self.ek)
def e(self,s,n):
    '''Степенева залежність e(s)
    n - степінь
    ...
    return self.et*(s/self.st)**n
def power(self,k):
    '''Повертає словник еластичних і пластичних властивостей
    k - кількість ліній для апроксимації пластичної ділянки (2,4,8...)
    ...
    #n визначається з умови проходження через точку (eb+et,sb),
n=6...10
n=log((self.eb+self.et)/self.et)/log(self.sb/self.st)
ds=self.sb-self.st
#степенева залежність
k_=float(k)
s_e=[(self.st+i/k_*ds,self.e(self.st+i/k_*ds,n)-self.et) for i in
range(0,k+1)]
#s_e=[(self.st+i*ds,self.e(self.st+i*ds,n)-self.et) for i in
[0.0,0.25,0.5,0.75,1.0]]
s_e.append((self.sk,self.ek)) # додати точку руйнування
return {'el':((self.E,self.mu),),
        'pl':tuple(s_e)}

def set_values(part,feature,par):
    '''
    Присвоює значення параметрам
    Приклад:
    par={'h1':0.0002,'h2':0.00004}
    set_values(part='A1',feature='Shell planar-1',par=par)
    ...
    p=model.parts[part] # деталь
    f=p.features[feature] # елемент
    s=model.ConstrainedSketch(name='__edit__', objectToCopy=f.sketch) #
тимчасовий ескіз

```

```

    p.projectReferencesOntoSketch(filter=COPLANAR_EDGES, sketch=s,
upToFeature=f) # спроекувати
    for k,v in par.iteritems(): # для всіх параметрів
        s.parameters[k].setValues(expression=str(v)) # установити значення
    f.setValues(sketch=s) # установити ескіз
    del s # знищити
    p.regenerate() # оновити деталь

def mesh_all():
    '''Будує сітку скінченних елементів'''
    ra=model.rootAssembly # збірка
    # елементи збірки
    reg=(ra.instances['Nipple-1'],ra.instances['Rod-1'],ra.instances['Tool-
1']) #!
    ra.deleteMesh(regions=reg) # знищити сітку
    ra.generateMesh(regions=reg) # створити сітку

def JobSubmit():
    ''' Виконує задачу '''
    myJob = mdb.jobs['Model-1'] # задача
    myJob.submit() # виконати задачу
    # Чекає поки задача не буде розв'язана
    myJob.waitForCompletion()

def readODB_set(set,step,var,pos=NODAL):
    '''Повертає список результатів у вузлах заданої множини вказаного кроку
set - множина
step - номер кроку
var - змінна:
(('S', INTEGRATION_POINT, ((INVARIANT, 'Mises'), )), )
(('U', NODAL, ((COMPONENT, 'U2'), )), )
pos - позиція: NODAL - для вузлів,INTEGRATION_POINT - для елементів
Приклад: readODB_set(set='Set-1',step='Step-1',var=var)
'''
    # отримати дані
    if pos==NODAL:
        dat=session.xyDataListFromField(odb=myOdb, outputPosition=NODAL,
variable=var, nodeSets=(set.upper(),)) # дані
    if pos==INTEGRATION_POINT:
        dat=session.xyDataListFromField(odb=myOdb,
outputPosition=INTEGRATION_POINT, variable=var, elementSets=(set.upper(),))
    # дані

    step_number=myOdb.steps[step].number # номер кроку

    nframes=[] # містить кількість фреймів в кожному кроці
    for k in myOdb.steps.keys(): # для всіх кроків
        nframes.append(len(myOdb.steps[k].frames))

    nstart_frame=nframes[step_number-2] # номер початкового фрейму кроку

```

```

    nend_frame=int(sum(nframes[step_number-2:step_number])) # номер
кінцевого фрейму кроку
    res=[] # список результатів
    for x in dat: # для всіх вузлів
        #x.data - це ((час,значення),(час,значення)...)
        res.append(x.data[nstart_frame:nend_frame]) # дані тільки з
вказаного кроку

    # видалити тимчасові дані
    for k in session.xyDataObjects.keys():
        del session.xyDataObjects[k]
    return res # повертає список значень

def readODB_set2(set,step,var,pos=NODAL):
    '''Повертає список середніх результатів в вузлах заданої множини
вказаного кроку
(менш універсальна альтернатива readODB_set())
set - множина
step - крок
var - змінна:
('S','Mises')
('S','Pressure')
('U','Magnitude')
('U','U1')
('CPRESS','')
('D','') # коефіцієнт запасу втомної міцності
pos - позиція: NODAL - для вузлів,INTEGRATION_POINT - для елементів
Приклад: readODB_set2(set='Cont',step='Step-1',var=('S','Mises'))
'''
    if pos==NODAL:
        s=myOdb.rootAssembly.nodeSets[set.upper()] # множина вузлів
    if pos==INTEGRATION_POINT:
        s=myOdb.rootAssembly.elementSets[set.upper()] # множина елементів
    m=[] # список середніх результатів з усіх вузлів множини
    for f in myOdb.steps[step].frames: # для кожного фрейму
        fo=f.fieldOutputs[var[0]].getSubset(region=s,position=pos) # дані
        #openOdb(r'C:/Temp/Model-1.odb').steps['Step-
1'].frames[4].fieldOutputs['CPRESS'].getSubset(position=NODAL,
region=openOdb(r'C:/Temp/Model-
1.odb').rootAssembly.nodeSets['CONT']).values[0].data
        res=[] # список результатів
        for v in fo.values: # для кожного вузла/елемента
            if var[1]=='Mises': res.append(v.mises) # додати до списку
результатів
            if var[1]=='S11': res.append(v.data.tolist()[0])
            if var[1]=='S22': res.append(v.data.tolist()[1])
            if var[1]=='S33': res.append(v.data.tolist()[2])
            if var[1]=='S12': res.append(v.data.tolist()[3])
            if var[1]=='Pressure': res.append(v.press)
            if var[0]=='U' and var[1]=='Magnitude': res.append(v.magnitude)
            if var[1]=='U1': res.append(v.data.tolist()[0])

```

```

        if var[1]=='U2': res.append(v.data.tolist()[1])
        if var[0]=='CPRESS': res.append(v.data)
    m.append((f.frameValue, sum(res)/len(res))) # додати середнє з
усіх вузлів
    return m # повертає список значень

def findmax(data):
    '''Повертає максимальне значення в форматі (час, значення)'''
    max=(0,0)
    for x in data:
        if x[1]>max[1]:
            max=x
    return max

import csv
csv_file=open("results.csv", "wb") # відкрити csv файл
writer = csv.writer(csv_file,delimiter = ';') # установити розділювач
writer.writerow(['deltax','st','lc','step_time','S22','step_time','cpress']
) # записати рядок
model=mdb.models['Model-1'] # модель

for deltax in [0.1,0.15,0.2,0.25,0.3]: # цикл для зміни глибини переміщення
штампів
    for lc in [100.0,140.0,180.0]: # цикл для зміни довжини обтискання
        for st in [300.0,400.0,500.0]: # цикл для зміни границі плинності
стали
            # установити значення геометричних параметрів
            set_values(part='Nipple',feature='Shell planar-
1',par={'l1':lc+20,'r1':17})
            set_values(part='Tool',feature='Wire-
1',par={'l1':lc,'l2':lc+10})
            set_values(part='Rod',feature='Shell planar-
1',par={'l1':lc+70})
            model.rootAssembly.regenerate() # оновити
            #створити матеріал

mat_steel=Material(E=210000.0,mu=0.28,st=st,sv=600.0,delta=21.0,psi=56.0)
            # визначити матеріали
            model.materials['Material-
1'].elastic.setValues(table=((mat_steel.E, mat_steel.mu), ))
            model.materials['Material-
1'].plastic.setValues(table=mat_steel.power(8)['pl'])
            model.materials['Material-2'].elastic.setValues(table=((0.1e5,
0.5e5, 0.1e5, 0.22, 0.22, 0.22, 0.04e5, 0.04e5, 0.2e5), ))
            mesh_all() # створити сітку
            model.boundaryConditions['BC-4'].setValues(u1=-deltax)
#гранична умова
            #model.loads['Load-1'].setValues(magnitude=press)
            JobSubmit() # виконати задачу
            myOdb = openOdb(path=model.name + '.odb') # відкрити базу даних
результатів

```

```

        session.viewports['Viewport:
1'].setValues(displayedObject=myOdb)

        # отримати напруження
        x1=readODB_set2(set='Up',step='Step-
2',var=('S','S22'),pos=INTEGRATION_POINT)
        x1max=findmax(x1) # знайти максимальне з усіх фреймів

        # отримати контактний тиск
        x2=readODB_set2(set='Nip',step='Step-1',var=('CPRESS',''))
        x2max=findmax(x2) # знайти максимальне з усіх фреймів

        # отримати вертикальне переміщення
        #x3=readODB_set2(set='Bot',step='Step-2',var=('U','U2'))

        ## отримати напруження
        #var=(('S', INTEGRATION_POINT, ((INVARIANT, 'Mises'), )), )
        #print readODB_set(set='Up',step='Step-2',var=var)
        ## отримати вертикальне переміщення
        #var=(('U', NODAL, ((COMPONENT, 'U2'), )), )
        #print readODB_set(set='Bot',step='Step-2',var=var)

        # записати дані у файл
        writer.writerow([deltax, st, lc, x1max[0], x1max[1], x2max[0],
x2max[1]])
        myOdb.close() # закрити базу даних результатів
        csv_file.close() # закрити файл csv

```

ДОДАТОК Р

Система автоматизованого проектування металополімерних з'єднань на основі вільного програмного забезпечення

Код доступний також в GitHub (vkopey/Fiberglass-Sucker-Rod-Connection: FEM model of fiberglass sucker rod connection. URL: <http://github.com/vkopey/Fiberglass-Sucker-Rod-Connection>).

Таблиця Р.1 – Структура вхідного файлу CalculiX моделі з'єднання тіла склопластикової штанги зі сталевую головою

Секція файлу	Пояснення
*NODE, NSET=NALL 1, 0.000000,5.000000 ...	Визначення вузлів сітки у форматі: номер вузла, x, y
*NSET, NSET=Line1 52 ...	Визначення множини вузлів Line1 у форматі: номер вузла
*ELEMENT, type=CAX4, ELSET=Rod 1, 1,2,3,4 ...	Визначення множини елементів Rod типу CAX4 (4-вузловий осесиметричний елемент) у форматі: номер елемента, вузол 1, вузол 2, вузол 3, вузол 4
*SURFACE, NAME = Line2, TYPE = ELEMENT 33, S4 ...	Визначення поверхні Line2 з граней елементів у форматі: номер елемента, грань елемента
*ELSET,ELSET=Elset1 2 ...	Визначення множини елементів для отримання результатів у форматі: номер елемента
*MATERIAL, NAME=mat1 *ELASTIC,TYPE=ENGINEERING CONSTANTS 210000.0, 0.3	Визначення матеріалу з пружними властивостями у форматі: модуль пружності, коефіцієнт Пуассона або, якщо TYPE=ENGINEERING CONSTANTS: значення параметрів ортотропного матеріалу
*SOLID SECTION, ELSET=Rod, MATERIAL=mat1 1.	Визначення матеріалів для множини елементів Rod
*SURFACE INTERACTION, NAME=Int1 *SURFACE BEHAVIOR, PRESSURE- OVERCLOSURE=LINEAR 1.E7, 3. *FRICTION 0.2, 47000 *CONTACT PAIR, INTERACTION=Int1, ADJUST=Line4, TYPE=SURFACE TO SURFACE Line4, Line2	Визначення взаємодії контактних поверхонь, характеристик поверхні і тертя, контактних поверхонь. Тут параметр ADJUST дозволяє перемістити вибрані вузли Line4 підпорядкованої поверхні Line4 на початку першого кроку так, щоб вони контактували з головною поверхнею Line2.

Кінець таблиці P.1

Секція файлу	Пояснення
*BOUNDARY Line1,1,2,0.0	Визначення граничної умови, яка забороняє переміщення по x і y на вузлах Line1
*AMPLITUDE,NAME=A1 0.0,0.0, ...	Визначення залежності A1 величини від часу у форматі: час, значення величини
*STEP,NLGEOM *STATIC 1E-4,1.0	Визначення першого кроку статичної задачі тривалістю 1 с
*NODE FILE U, *EL FILE S,	Визначення результатів першого кроку, які будуть зберігатись у файл результатів jobname.frd - переміщення і напруження
*END STEP	Кінець визначення першого кроку
*STEP,NLGEOM *STATIC,DIRECT 0.1,1.0	Визначення другого кроку статичної задачі тривалістю 1 с. Параметр DIRECT означає, що підкроки будуть тривати 0,1 с.
*BOUNDARY,AMPLITUDE=A1 Line3,2,2,10.0	Визначення граничної умови, яка переміщує по y вузли Line2 на величину $y(t)=10*A1(t)$ відповідно залежності A1
*EL PRINT,ELSET=Elset1 S	Визначення результатів (напруження в елементах Elset1) другого кроку, які будуть зберігатись у файл результатів jobname.dat
*END STEP	Кінець визначення другого кроку

Лістинг P.1 – fibrod_regular.py – код програми

```
#encoding: utf-8
# скінченно-елементна модель з'єднання з прямокутною регулярною сіткою
from __future__ import division
import os
import numpy as np
import matplotlib.pyplot as plt
from math import pi,sin

class RodConnector:
    """Параметри з'єднання"""
    r0=11.0 # радіус штанги
    r1=18.0 # радіус ніпеля
    l0=150.0 # довжина штанги
    l1=150.0 # довжина ніпеля
    l3=1.875 # зміщення штанги відносно ніпеля вверх
    # рекомендується l3=(l1/n1[1])/2
    n0=3,40 # кількість елементів штанги по x,y
    n1=2,40 # кількість елементів ніпеля по x,y
    y1,y2=10.0, 140.0 # границі вертикальної зони деформування
    args=[0.12, 1.0, 1.0] # параметри деформування для функції dx()
    def dx(self,y):
        """Закон деформування - залежність зміщень (в напрямку до осі)
```

```

точок внутрішньої поверхні ніпеля від ординати точки у"""
a,b,c=self.args
#0.1 # рівномірне
#0.01*y # клиноподібне
#0.2+0.01*y # рівномірне+клиноподібне
#1.-0.0005*(y-70)**2 # параболічне
#1.0+0.5*sin(2*pi*y/30.0+0.0) # гармонічне
#1.0+0.01*y*sin(2*pi*y/30.0+0.0) # гармонічне зі змінною амплітудою
#0.1+(0.2-0.001*y)*sin(2*pi*y/30.0+0.0) # гармонічне зі змінною
амплітудою (обернене)
return self.args[0]

```

```
class Model:
```

```

"""Скінченно-елементна модель з'єднання з прямокутною регулярною сіткою
Увага! Тут нумерація індексів з 0, а у inp-файлі з 1"""
def __init__(self):
    self.ccx=r"c:\CalculiXLauncher-03.1-beta_Windows-
32bit\bin\ccx\ccx212.exe" # повний шлях до розв'язувача
    self.dirccx=os.path.dirname(self.ccx)
    os.chdir(self.dirccx)
    self.filename='fiber' # назва файлу моделі
    self.reset()

def reset(self): # очистити сітку
    self.N=[] # вузли
    self.E=[] # елементи
    self.indE={} # індекси елементів деталей

def indN(self,part):
    """Індекси вузлів деталі part"""
    s=set()
    for i in self.indE[part]:
        s.update(set(self.E[i]))
    return list(s)

def mesh(self,x,y,lx,ly,nx,ny):
    """Будує регулярну сітку на прямокутній області
    x,y # лівий нижній кут
    lx,ly # ширина, висота
    nx,ny - кількість елементів в ширину і висоту"""
    newE=[] # індекси нових елементів
    dx, dy = lx/nx, ly/ny # ширина, висота елемента
    for j in range(ny):
        for i in range(nx):
            #print i,j
            e=[] # елемент
            for dn in [(0,0),(dx,0),(dx,dy),(0,dy)]: # додати 4 вузли
                (проти годинникової стрілки)
                n=[i*dx+dn[0]+x, j*dy+dn[1]+y] # вузол
                if n not in self.N: # якщо такого вузла ще немає
                    self.N.append(n) # додати вузол

```

```

        e.append(self.N.index(n)) # додати номер вузла в
елемент
        self.E.append(e) # додати елемент
        newE.append(len(self.E)-1) # додати індекс елемента
return newE

def getVLineNodes(self,part,x,y1,y2):
    """Повертає індекси вузлів вертикальної лінії"""
    L=[] # вузли однієї лінії
    for i in self.indN(part): # для індексів вузлів деталі part
        n=self.N[i] # вузол
        if n[0]==x and n[1]>=y1 and n[1]<=y2: # якщо належить лінії
            L.append([i,n])
    L.sort(key=lambda x: x[1][1]) # сортувати по y
    L=[i[0] for i in L] # тільки індекси
return L

def getHLineNodes(self,part,y,x1,x2):
    """Повертає індекси вузлів горизонтальної лінії"""
    L=[] # вузли однієї лінії
    for i in self.indN(part): # для індексів вузлів деталі part
        n=self.N[i] # вузол
        if n[1]==y and n[0]>=x1 and n[0]<=x2: # якщо належить лінії
            L.append([i,n])
    L.sort(key=lambda x: x[1][0]) # сортувати по x
    L=[i[0] for i in L] # тільки індекси
return L

def getElements(self,Nodes):
    """Повертає список елементів і їх граней за впорядкованим списком
вузлів лінії"""
    n=0
    E=[] # список елементів
    P={(0,1):'S1',(1,2):'S2',(2,3):'S3',(0,3):'S4'} # грані елемента
    while n<len(Nodes)-1: # кожний вузол крім останнього
        n1,n2=Nodes[n],Nodes[n+1] # пара сусідніх вузлів
        # шукаємо цю пару серед усіх елементів
        for i,e in enumerate(self.E):
            if n1 in e and n2 in e:
                p=sorted((e.index(n1),e.index(n2))) # вузли грані
                E.append((i,P[tuple(p)]))
        n+=1
return E

def deform(self):
    """Деформування сітки лінії
Увага! Величина деформування повинна бути меншою розміру елемента
Увага! Деформувати сітку тільки після визначення потрібних вузлів
функціями getHLineNodes, getVLineNodes"""
    L=self.getVLineNodes(part='nipple',x=rc.r0,y1=0,y2=rc.l1)
    for i in L: # для кожного вузла лінії

```

```

        x,y=self.N[i]
        if y>=rc.y1 and y<=rc.y2: # межі деформування
            self.N[i][0]=x-rc.dx(y) # змістити вузол вліво на величину
dx
def draw(self):
    """Рисує сітку"""
    N=self.N
    for i,e in enumerate(self.E):# [:4]
        X=[N[e[0]][0],N[e[1]][0],N[e[2]][0],N[e[3]][0],N[e[0]][0]]
        Y=[N[e[0]][1],N[e[1]][1],N[e[2]][1],N[e[3]][1],N[e[0]][1]]
        if i in self.indE['nipple']:
            plt.plot(X, Y, 'ko-')
        else:
            plt.plot(X, Y, 'ro-')
    plt.axes().set_aspect('equal')
    plt.show()

def writeINP(self,Lines):
    """Створює inp-файл для CalculiX
    Увага! Нумерація індексів у файлі з 1, тому усі індекси збільшуємо
на 1"""
    Line1,Line2,Line3,Line4,Line5=Lines # списки вузлів потрібних ліній

    s="*NODE, NSET=NALL\n"
    for i,n in enumerate(self.N):
        x,y=n
        s+="%d, %f,%f\n"%(i+1,x,y)

    s+="*NSET, NSET=Line1\n"
    for i in Line1:
        s+="%d\n"%(i+1)

    s+="*NSET, NSET=Line2\n"
    for i in Line2:
        s+="%d\n"%(i+1)

    s+="*NSET, NSET=Line3\n"
    for i in Line3:
        s+="%d\n"%(i+1)

    s+="*NSET, NSET=Line4\n"
    for i in Line4:
        s+="%d\n"%(i+1)

    s+="*NSET, NSET=Line5\n"
    for i in Line5:
        s+="%d\n"%(i+1)

    s+="*ELEMENT, type=CAX4, ELSET=Rod\n" # або CAX4R
    for i,e in enumerate(self.E):

```

```

        if i==self.indE['nipple'][0]:
            s+="*ELEMENT, type=CAX4, ELSET=Nipple\n"
            s+="%d, %d,%d,%d,%d\n"%(i+1,e[0]+1,e[1]+1,e[2]+1,e[3]+1)

s+="*SURFACE, NAME = Line2, TYPE = ELEMENT\n"
for i,j in self.getElements(Line2):
    s+="%d, %s\n"%(i+1, j)

s+="*SURFACE, NAME = Line4, TYPE = ELEMENT\n"
for i,j in self.getElements(Line4):
    s+="%d, %s\n"%(i+1, j)

Elset1=set() # елементи, де потрібні будуть результати
for i,j in self.getElements(Line3)+self.getElements(Line4):
    Elset1.add(i)
s+="*ELSET,ELSET=Elset1\n"
for i in Elset1:
    s+="%d\n"%(i+1)

s+="""
*MATERIAL, NAME=mat1
*ELASTIC,TYPE=ENGINEERING CONSTANTS
10000.0,50000.0,10000.0,0.22,0.22,0.22,4000.0,4000.0
20000.0,295.0
*MATERIAL, NAME=mat2
*ELASTIC
210000.0, 0.3
*SOLID SECTION, ELSET=Rod, MATERIAL=mat1
1.
*SOLID SECTION, ELSET=Nipple, MATERIAL=mat2
1.
*SURFACE INTERACTION, NAME=Int1
*SURFACE BEHAVIOR, PRESSURE-OVERCLOSURE=LINEAR
1.E7, 3.
*FRICTION
0.2, 47000
*CONTACT PAIR, INTERACTION=Int1, ADJUST=Line4, TYPE=SURFACE TO SURFACE
Line4, Line2
*BOUNDARY
Line5,1,1,0.0
*BOUNDARY
Line1,1,2,0.0
*AMPLITUDE,NAME=A1
0.0,0.0,0.1,0.1,0.2,0.2,0.3,0.3,
0.4,0.4,0.5,0.5,0.6,0.6,0.7,0.7,
0.8,0.8,0.9,0.9,1.0,1.0
*STEP,NLGEOM
*STATIC
1E-4,1.0
*NODE FILE
U,

```

```
*EL FILE
S,
*END STEP
```

```
*STEP,NLGEOM
*STATIC,DIRECT
0.01,1.0
*BOUNDARY
**BOUNDARY,AMPLITUDE=A1
Line3,2,2,5.0
*NODE FILE
U,
*EL FILE
S,
** *NODE PRINT,NSET=Line3
** RF
*EL PRINT,ELSET=Elset1
S
*END STEP
"""
```

```
f=open(self.dirccx+"\\"+self.filename+".inp", 'w')
f.write(s)
f.close()
```

```
def runCCX(self):
    """Розраховує задачу в CalculiX"""
    import subprocess
    s=subprocess.check_output([self.ccx, "-i", self.dirccx+"\\"
+self.filename], shell=True)
    L=[ln.strip() for ln in s.splitlines()[-10:]] # останні рядки
виведення
    if "Job finished" in L:
        return 1 # якщо розрахунок успішний
    return 0 # якщо розрахунок не успішний
```

```
def readResult1(self, Nodes):
    """Читає результати для елементів штанги в місці контакту"""
    elset=[str(i+1) for i,j in self.getElements(Nodes)]
    f=open(self.dirccx+"\\"+self.filename+".dat",'r')
    L=f.readlines()
    f.close()
    Y=[] # значення
    inc1=True
    for s in L: # для кожного рядка
        w=s.strip().split(' ') # слова
        w=[i for i in w if i.strip()!=''] # без пустих слів
        if w==[]: continue # пропустити пусті рядки
        if w[0]=='stresses':
            if not inc1: break # тільки для першого інкременту
            inc1=False
```

```

        if w[0] in elset: # якщо елемент у списку
            if w[1]=='3': # integration point (1 для CAХ4R)
                Y.append(float(w[2])) # !!! напруження sxx
    return min(Y) # бо зі знаком -

def readResult2(self, Nodes):
    """Читає результати для елементів штанги в місці верхнього торця"""
    elset=[str(i+1) for i,j in self.getElements(Nodes)]
    f=open(self.dirccx+"\\"+self.filename+".dat", 'r')
    L=f.readlines()
    f.close()
    T=[] # час
    Y=[] # значення
    n=len(elset) # кількість елементів
    for s in L: # для кожного рядка
        w=s.strip().split(' ') # слова
        w=[i for i in w if i.strip()!=''] # без пустих слів
        if w==[]: continue # пропустити пусті рядки
        #print w
        if w[0]=='stresses': #'forces'
            T.append(float(w[-1]))
        if w[0]==elset[0]: # якщо перший елемент списку
            if w[1]=='3': # integration point (1 для CAХ4R)
                Y.append(float(w[3])/n)
        if w[0] in elset[1:]: # якщо не перший елемент списку
            if w[1]=='3': # integration point
                Y[-1]=Y[-1]+float(w[3])/n # середнє по елементам
    plt.plot(T,Y,'ko-')
    plt.show()

    # D=[(b-a)/(T[1]-T[0]) for a,b in zip(Y[:-1],Y[1:])] # похідна
dy/dt
    # print D
    # dmin=200 # мінімальне допустиме (малий наклон дотичної на рис.
означає руйнування з'єднання)
    # for i,d in enumerate(D): # для усіх значень dy/dt
    #     if d<dmin: # якщо менше допустимого
    #         return T[i],Y[i] # то це точка руйнування
    Ymax=max(Y)
    t=T[Y.index(Ymax)]
    if t==T[-1]: print "Увага! Можливо знайдено максимальне напруження.
Збільшіть величину осьової деформації"
    return t, Ymax

def run(self, runCCX=True):
    """Створює модель і виконує задачу"""
    self.reset() # очистити
    # створити сітку
    self.indE['rod']=self.mesh(x=0.0, y=rc.l3, lx=rc.r0, ly=rc.l0,
nx=rc.n0[0], ny=rc.n0[1])
    print len(self.N),len(self.E)

```

```

self.indE['nipple']=self.mesh(x=rc.r0, y=0.0, lx=rc.r1-rc.r0,
ly=rc.l1, nx=rc.n1[0], ny=rc.n1[1])

Line1=self.getHLineNodes('nipple', y=0, x1=0, x2=rc.r1) # низ
Line2=self.getVLineNodes('nipple', x=rc.r0, y1=0, y2=rc.l1) #
контакт
Line3=self.getHLineNodes('rod', y=rc.l0+rc.l3, x1=0, x2=rc.r0) #
верх
Line4=self.getVLineNodes('rod', x=rc.r0, y1=rc.l3, y2=rc.l0+rc.l3) #
# контакт
Line5=self.getVLineNodes('rod', x=0, y1=rc.l3, y2=rc.l0+rc.l3) #
вісь

self.deform() # !!! деформувати сітку тільки після визначення
потрібних вузлів функціями getHLineNodes, getVLineNodes
self.writeINP([Line1,Line2,Line3,Line4,Line5])
if runCCX==False:
    self.draw() # тільки нарисувати сітку і вийти
    return
if self.runCCX(): # розрахувати і повернути результати
    sxx=self.readResult1(Nodes=Line4)
    t,syy=self.readResult2(Nodes=Line3)
    print sxx,t,syy
    return sxx,t,syy

def optimize(self):
    """Виконує послідовність задач для оптимізації"""
    open('result.csv', 'w').close() # створити файл результатів
    for x in [0.10,0.11,0.12,0.13,0.14,0.15]: # значення параметрів
моделі
        rc.args[0]=x # змінити значення параметрів моделі
        sxx,t,syy=self.run() # розрахувати і отримати результати
        with open('result.csv', 'a') as f: # додати результати у файл
            f.write("%f;%f;%f;%f\n"%(x,sxx,t,syy))

##
rc=RodConnector()
mr=Model()
#mr.run(False) # тільки створити модель і показати сітку
mr.run() # виконати задачу
#mr.optimize() # виконати послідовність задач

```


ДОДАТОК С

Програма для обчислення КІН за результатами моделювання МСЕ

Код доступний також в GitHub (vkopey/SIF: Calculation of stress intensity factor (SIF) and fatigue life by FEM results. URL: <http://github.com/vkopey/SIF>).

Лістинг С.1 – Код програми

```
# -*- coding: utf-8 -*-
"""Розрахунок КІН за результатами моделювання МСЕ"""
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

E=2.1e11
G=7.9e10 # модуль зсуву, Па
#G=E/(2+2*nu)
nu=0.28 # коефіцієнт Пуассона
r=0.2/1000 # віддаль від вістря тріщини до місця заміру, м
sigma=56.0e6 # кільцеві напруження, Па
d=5.5/1000 # товщина стінки труби, м

def K(V):
    """Коефіцієнт інтенсивності напружень (КІН), Па*м**0.5
    V - половина величини розкриття тріщини, м"""
    k=3-4*nu # для плоского деформування
    #return np.sqrt(2*np.pi)*2*G*V/((1+k)*np.sqrt(r))
    return np.sqrt(2*np.pi/r)*V*E/(4-4*nu**2) # [Meinhard Kuna]

def K_(s):
    """Коефіцієнт інтенсивності напружень (КІН), Па*м**0.5
    s - напруження біля тріщини, Па"""
    return s*np.sqrt(2*np.pi*r)

def Y(K,a):
    """Поправочна функція"""
    return K/(sigma*np.sqrt(np.pi*a))

def Kt(Y,sigma,a):
    """теоретичне значення КІН, Па*м**0.5
    Y - значення поправочної функції
    sigma - напруження, Па
    a - глибина тріщини, м"""
    return Y*sigma*np.sqrt(np.pi*a)

def Kt1(sigma,b):
```

```

"""Теоретичний КІН [Муракамі, т.2, с.556, табл.9.33]
R - внутрішній радіус труби
t - товщина стінки труби
b - глибина тріщини
a - половина довжини тріщини
p - внутрішній тиск
sigma - кільцеві напруження в трубі
"""

R=62.0/2000
t=5.5/1000
Rt=50.0/1000 # радіус кругового фронту тріщини
a=(Rt**2-(Rt-b)**2)**0.5
p=sigma*t/R
Q=1+1.464*(b/a)**1.65
R0=R+t
G0=np.interp(b/t, [0.2, 0.5, 0.8], [1.147, 1.584, 2.298])
G1=np.interp(b/t, [0.2, 0.5, 0.8], [0.685, 0.839, 1.099])
G2=np.interp(b/t, [0.2, 0.5, 0.8], [0.521, 0.600, 0.739])
G3=np.interp(b/t, [0.2, 0.5, 0.8], [0.432, 0.480, 0.568])
Fe=(t/R)*(R*R/(R0*R0-
R*R))*(2*G0+2*G1*b/R0+3*G2*(b/R0)**2+4*G3*(b/R0)**3)
return Fe*(p*R/t)*(np.pi*b/Q)**0.5

def v(K):
    """Швидкість росту тріщини, м/цикл
    K - КІН, Па*м**0.5"""
    K=K/1.0e6 # перевести МПа*м**0.5
    # параметри діаграми втомного руйнування сталі 20Н2М 3% NaCl [ Копей Б.
Штанги... моногр. рис. 4.5]:
    C=9.14e-13 # м/(МПа*м**0.5)
    n=3.6
    #C=6.0e-17 # повітря
    #n=6.04
    return C*K**n

a=np.array([0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5])/1000 # глибина тріщини, м

def N(sigma,f,popt):
    """Циклічна довговічність, цикли
    sigma - напруження, Па"""
    a_=np.linspace(a.min(), a.max()) # розміри тріщини, м
    return np.trapz(1/v( Kt(f(a_/d,*popt),sigma,a_ ) ), a_) # інтеграл по a

def N_(sigma,f,popt): # інший спосіб інтегрування для перевірки
    A=a.min()
    dN=1000.0
    #dA=0.01/1000 # або
    N=0.0
    while A<=a.max():
        K=Kt(f(A/d,*popt),sigma,A)
        V=v(K)

```

```

    dA=V*dN
    #dN=dA/V # або
    A=A+dA
    N=N+dN
return N

def N1(sigma,material):
    """Кількість циклів з рівняння втоми [Трощенко т.1, с.51]"""
    D={"сталь" :dict(s=210.0e6, N=3.0e6, m=9.00), # середня сталь
      "сталь45" :dict(s=253.0e6, N=1.8e6, m=8.72), # гладкий зразок,
сталь 45, відпал, на повітрі, R=-1 [Трощенко т.1, с.456]
      "сталь45NaCl":dict(s=100.0e6, N=2.5e7, m=2.76), # гладкий зразок,
сталь 45, відпал, 3% NaCl, R=-1 [Трощенко т.1, с.456]
      "20H2M" :dict(s=190.0e6, N=2.5e6, m=5.86), # моно. Рисунок 4.5
      "20H2MNaCl" :dict(s=60.00e6, N=2.0e7, m=2.3), # моно. Рисунок 4.5
      "20H2MNaCl_" :dict(s=125.0e6, N=1.0e6, m=1.86), # моно. Рисунок 3.15
      "20H2M_" :dict(s=370.0e6, N=1.0e6, m=6.09), # Трощенко с.672
      "20H2MNaCl__":dict(s=50.0e6, N=5.0e6, m=3.32, m1=10.3) # Трощенко
с.672
    }
    d=D[material]
    s=d['s'] # ордината точки перелому кривої втоми в лог. коорд.
    N=d['N'] # абсциса точки перелому кривої втоми в лог. коорд.
    m=d['m'] # показник нахилу кривої втоми в лог. коорд.
    psi=0.2 # коефіцієнт чутливості матеріалу до асиметрії циклу
    s=2*s/(1+psi) # перевід з R=-1 в R=0 (максимальне, а не амплітуда)
    if sigma<s:
        if d.has_key('m1'): return (N*s**d['m1'])/sigma**d['m1'] # якщо
крива має 2 переломи
        #else: return 1e8 # включити тільки для візуалізації !
    return (N*s**m)/sigma**m

def S1(n):
    """Напруження з рівняння втоми (обернене до N1 - перевірено) n -
кількість циклів"""
    s=253.0e6
    N=1.8e6
    m=8.72
    sigma=((N*s**m)/n)**(1/m) # ампл. напруження для R=-1, що відповідає n
    psi=0.2 # коефіцієнт чутливості матеріалу до асиметрії циклу
    sigma=sigma/(1+psi) # ампл. напруження для R=0, що відповідає n (з
залежності sa=sa_eq-psi*sm)
    return 2*sigma # макс. напруження для R=0, що відповідає n

e=a/d # відносна глибина тріщини
# половина величини розкриття тріщини, мкм:
# без бандажу
V1=[0.19311,0.30551,0.40903,0.52202,0.65405,0.79376,0.94576,1.09433,1.24091
]
# з бандажем 84 мм

```

```

V2=[0.08682, 0.17087, 0.24629, 0.30835, 0.36911, 0.42522, 0.47333, 0.52429,
0.58763]
# з бандажем 74 мм
#V2=[0.09031, 0.17794, 0.25781, 0.32571, 0.39187, 0.4531, 0.50657, 0.55936,
0.62274]
# з бандажем 84 мм по panetration
#V2=[0.18499, 0.29258, 0.39114, 0.49768, 0.62166, 0.75185, 0.89302,
1.03042, 1.16669]
V1=np.array(V1)/1000000 # в метрах
V2=np.array(V2)/1000000

##
Y1=Y(K(V1),a)
def f1(x,a,b,c): # модель
    return a*x**2+b*x+c
popt1, pcov1 = curve_fit(f1, e, Y1) # апроксимувати нелінійним методом
найменших квадратів
print popt1 # коефіцієнти a,b,c
print "R^2=", np.corrcoef(Y1, f1(e,*popt1))[0, 1]**2 # коефіцієнт
детермінації
print "N(%fПа)=%sigma, np.trapz(1/v(K(V1)), a) # інтегрувати задану
масивом функцію методом трапецій

##
Y2=Y(K(V2),a)
def f2(x,a,b,c,d): # модель
    return a*x**3+b*x**2+c*x+d
popt2, pcov2 = curve_fit(f2, e, Y2) # апроксимувати нелінійним методом
найменших квадратів
print popt2 # коефіцієнти
print "R^2=", np.corrcoef(Y2, f2(e,*popt2))[0, 1]**2 # коефіцієнт
детермінації
print "N(%fПа)=%sigma, np.trapz(1/v(K(V2)),a)

##
plt.figure()
plt.plot(K(V1), v(K(V1))*1e6, 'ko-') # кінетична діаграма втомного
руйнування
#np.savetxt('KDVRa.csv', v(K(V1))*1e6, delimiter=';')
_ =np.loadtxt('KDVRa.csv', delimiter=';')
plt.plot(K(V1), _, 'rs-')
plt.xlabel(u'$K, Пам^{1/2}$'),plt.ylabel(u'$v, мкм/цикл$')
plt.show()

##
plt.figure()
# перевірка - мають збігатись:
plt.plot(a, K(V1), 'ko-')
plt.plot(a, Kt(f1(a/d,*popt1),sigma,a), 'r^-' )
plt.plot(a, Kt1(sigma,a), 'rs-')
plt.xlabel('$a$'),plt.ylabel(u'$K$')

```

```

plt.show()

##
plt.figure()
S=np.linspace(80.0e6, 500.0e6)
plt.plot([np.log10(N(s,f1,popt1)) for s in S], S, 'k-')
#plt.plot([N(s,f1,popt1) for s in S], S, 'k-')

plt.plot([np.log10(N(s,f2,popt2)) for s in S], S, 'k--')
#plt.plot([N(s,f2,popt2) for s in S], S, 'k--')

# зберегти масиви
#np.savetxt('_ .csv', [np.log10(N(s,f2,popt2)) for s in S], delimiter=';')
#нарисувати збережені np.savetxt графіки
_=np.loadtxt('74.csv', delimiter=';')
plt.plot(_, S, 'k:')
_=np.loadtxt('84np.csv', delimiter=';')
plt.plot(_, S, 'r-.')
_=np.loadtxt('73a.csv', delimiter=';')
plt.plot(_, S, 'r-')
_=np.loadtxt('84a.csv', delimiter=';')
plt.plot(_, S, 'r--')

# графіки вТОМИ за рівнянням вТОМИ
import scipy
N1=scipy.vectorize(N1)
S=np.linspace(80.0e6, 500.0e6)
for k,_ in enumerate(["20H2M", "20H2MNaCl"]):
    plt.plot(np.log10(N1(S,_)), S, 'k-.', linewidth=k+1)

plt.xlabel('$\log(N)$'),plt.ylabel(u'$\text{Па}$')
plt.grid()
plt.show()

##
plt.figure()
plt.plot(e, Y1, 'ko') # нарисувати емпіричну залежність
plt.plot(e, f1(e,*popt1), 'k-') # нарисувати апроксимовану залежність
plt.plot(e, Y2, 'k^')
plt.plot(e, f2(e,*popt2), 'k-')
plt.xlabel('$\epsilon$'),plt.ylabel('$Y$')
plt.show()

```

ДОДАТОК Т

**Макрос Abaqus/CAE для оптимізації конструкції ШН за критерієм втомної
міцності з fe-safe**

Код доступний також в GitHub (vkorey/Abaqus-feSafe-optimization: Abaqus CAE macro for design optimization with fe-safe. URL: <http://github.com/vkorey/Abaqus-feSafe-optimization>).

Лістинг Т.1 – rod3D.py

```
# -*- coding: cp1251 -*-
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
import subprocess

def set_values(part,feature,par):
    '''
    Присвоює значення параметрам
    Приклад:
    par={'h1':0.0002,'h2':0.00004}
    set_values(part='A1',feature='Shell planar-1',par=par)
    '''
    p=model.parts[part] # деталь
    f=p.features[feature] # елемент
    s=model.ConstrainedSketch(name='__edit__', objectToCopy=f.sketch) #
    тимчасовий ескіз
    p.projectReferencesOntoSketch(filter=COPLANAR_EDGES, sketch=s,
    upToFeature=f) # спроекувати
    for k,v in par.iteritems(): # для всіх параметрів
        s.parameters[k].setValues(expression=str(v)) # установити значення
    f.setValues(sketch=s) # установити ескіз
    del s # знищити
    p.regenerate() # оновити деталь

def mesh_all(lst):
```

```

'''Будує сітку скінченних елементів
lst - список назв елементів збірки'''
ra=model.rootAssembly #збірка
# елементи збірки
reg=[ra.instances[i] for i in lst]
#reg=(ra.instances['Part-1-1'],ra.instances['Part-2-2'])
ra.deleteMesh(regions=reg) # знищити сітку
ra.generateMesh(regions=reg) # створити сітку

def JobSubmit(job):
    '''Виконує задачу'''
    myJob = mdb.jobs[job] # задача
    myJob.submit() # виконати задачу
    # Чекає поки задача не буде розв'язана
    myJob.waitForCompletion()

def readODB_set(set,step,var,pos=NODAL):
    '''Повертає список результатів у вузлах заданої множини вказаного кроку
set - множина
step - номер кроку
var - змінна:
(('S', INTEGRATION_POINT, ((INVARIANT, 'Mises'), )), )
(('U', NODAL, ((COMPONENT, 'U2'), )), )
pos - позиція: NODAL - для вузлів,INTEGRATION_POINT - для елементів
Приклад: readODB_set(set='Set-1',step='Step-1',var=var)
'''
    #отримати дані
    if pos==NODAL:
        dat=session.xyDataListFromField(odb=myOdb, outputPosition=NODAL,
variable=var, nodeSets=(set.upper(),)) # дані
    if pos==INTEGRATION_POINT:
        dat=session.xyDataListFromField(odb=myOdb,
outputPosition=INTEGRATION_POINT, variable=var, elementSets=(set.upper(),))
# дані

    step_number=myOdb.steps[step].number # номер кроку

    nframes=[] # містить кількість фреймів в кожному кроці
    for k in myOdb.steps.keys(): # для всіх кроків
        nframes.append(len(myOdb.steps[k].frames))

    nstart_frame=nframes[step_number-2] # номер початкового фрейму кроку
    nend_frame=int(sum(nframes[step_number-2:step_number])) # номер
кінцевого фрейму кроку
    res=[] # список результатів
    for x in dat: # для всіх вузлів
        #x.data - це ((час,значення),(час,значення)...)
        res.append(x.data[nstart_frame:nend_frame]) # дані тільки з
вказаного кроку

    # видалити тимчасові дані

```

```

for k in session.xyDataObjects.keys():
    del session.xyDataObjects[k]
return res # повертає список значень

def readODB_set_(set,var):
    '''Повертає список результатів в вузлах заданої множини
    (для змінних fe-safe)
    set - множина
    var - змінна:
    (('LOGLife-Repeats', ELEMENT_NODAL), )
    (('FOS@Life=Infinite', ELEMENT_NODAL), )
    (('%%Failure@Life=5E6-Repeats', ELEMENT_NODAL), )
    Приклад: readODB_set_(set='Set-1',var=var)
    '''

    # отримати дані
    dat=session.xyDataListFromField(odb=myOdb, outputPosition=NODAL,
    variable=var, nodeSets=(set.upper(),)) # дані

    res=[] # список результатів
    for x in dat: # для всіх вузлів
        #x.data - це ((час,значення),(час,значення)...)
        res.append(x.data) # дані

    # видалити тимчасові дані
    for k in session.xyDataObjects.keys():
        del session.xyDataObjects[k]
    return res # повертає список значень

def readODB_set2(set,step,var,pos=NODAL):
    '''Повертає список середніх результатів в вузлах заданої множини
    вказаного кроку
    (менш універсальна альтернатива readODB_set())
    set - множина
    step - крок
    var - змінна:
    ('S','Mises')
    ('S','Pressure')
    ('U','Magnitude')
    ('U','U1')
    ('CPRESS','')
    ('D','') # коефіцієнт запасу втомної міцності
    ('LOGLife-Repeats', '')
    ('FOS@Life=Infinite', '')
    ('%%Failure@Life=5E6-Repeats', '')
    pos - позиція: NODAL - для вузлів,INTEGRATION_POINT - для елементів
    Приклад: readODB_set2(set='Cont',step='Step-1',var=('S','Mises'))
    '''

    if pos==NODAL:
        s=myOdb.rootAssembly.nodeSets[set.upper()] # множина вузлів
    if pos==INTEGRATION_POINT:
        s=myOdb.rootAssembly.elementSets[set.upper()] # множина елементів

```



```

m=[] # список середніх результатів з усіх вузлів множини
for f in myOdb.steps[step].frames: # для кожного фрейму
    fo=f.fieldOutputs[var[0]].getSubset(region=s,position=pos) # дані
    #openOdb(r'C:/Temp/Model-1.odb').steps['Step-
1'].frames[4].fieldOutputs['CPRESS'].getSubset(position=NODAL,
region=openOdb(r'C:/Temp/Model-
1.odb').rootAssembly.nodeSets['CONT']).values[0].data
    res=[] # список результатів
    for v in fo.values: # для кожного вузла/елемента
        if var[1]=='Mises': res.append(v.mises) # додати до списку
результатів
        if var[1]=='S11': res.append(v.data.tolist()[0])
        if var[1]=='S22': res.append(v.data.tolist()[1])
        if var[1]=='S33': res.append(v.data.tolist()[2])
        if var[1]=='S12': res.append(v.data.tolist()[3])
        if var[1]=='Pressure': res.append(v.press)
        if var[0]=='U' and var[1]=='Magnitude': res.append(v.magnitude)
        if var[1]=='U1': res.append(v.data.tolist()[0])
        if var[1]=='U2': res.append(v.data.tolist()[1])
        if var[0]=='CPRESS': res.append(v.data)
    m.append((f.frameValue, sum(res)/len(res))) # додати середнє з
усіх вузлів
return m # повертає список значень

def findmax(data):
    '''Повертає максимальне значення в форматі (час, значення)'''
    max=(0,0)
    for x in data:
        if x[1]>max[1]:
            max=x
    return max

def runFeSafe(input_odb,input_stlx,output_odb):
    '''Виконує аналіз втомної міцності у fe-safe
input_odb - назва вхідного файлу результатів Abaqus (без розширення
.odb)
input_stlx - назва файлу моделі fe-safe (без розширення .stlx)
output_odb - назва вихідного файлу результатів Abaqus (без розширення
.odb)
'''
    s=r'd:\Program Files\Safe_Technology\fe-safe\version.6.2\exe\fe-
safe_cl.exe -s j=c:\1\{iodb}.odb b=c:\1\{istlx}.stlx o=c:\1\{oodb}.odb'
    s=s.format(iodb=input_odb, istlx=input_stlx, oodb=output_odb)
    # виконує обчислення в fe-safe та чекає завершення
    subprocess.Popen(s).communicate()

def writeLDFfile(filename,lst):
    '''Змінює вміст файлу визначення навантаження LDF fe-safe
filename - ім'я файлу
lst - список рядків-даних
'''

```

```

f=open(filename, "w")
s=""
# .ldf file created by fe-safe compliant product 6.2-01[mswin]

INIT
transitions=Yes
END

# Block number 1
BLOCK n=1, scale=1
lh=0 1 , ds=1, scale=1
lh=0 {x} , ds=2, scale=1
END

"" # шаблон файлу
s=s.format(x=lst[0]) # вміст файлу з шаблону
f.write(s)
f.close()

import csv
csv_file=open("results.csv", "wb") # відкрити csv файл
writer = csv.writer(csv_file,delimiter = ';')
writer.writerow(['r','load','LogLife','FOS','%Failure']) # записати рядок
model mdb.models['Model-3'] # модель

for rad in [0.067,0.25,0.5,0.75,1.0]: # цикл для зміни значення параметру r
# установити значення геометричного параметра r
set_values(part='Part-1',feature='Solid revolve-1',par={'r':rad})
#set_values(part='Part-1',feature='Shell planar-1',par={'r':rad})
model.rootAssembly.regenerate() # оновити модель
mesh_all(['Part-1-1']) # створити сітку скінченних елементів
#model.loads['Load-2'].setValues(magnitude=load)
JobSubmit('Job-3') # виконати задачу

for load in [0.1,0.15,0.2,0.25,0.3]: # цикл для зміни значення
навантаження згину
writeLDFfile('model3.ldf',[str(load)]) # змінити LDF файл fe-safe
oodb='results'+str(int(rad*1000))+'_'+str(int(load*100)) # назва
бази даних результатів
runFeSafe('Job-3','model3',oodb) # виконати fe-safe

#myOdb = openOdb(path=model.name + '.odb') # відкрити базу даних
результатів
myOdb = openOdb(path=oodb + '.odb') # відкрити базу даних
результатів
session.viewports['Viewport: 1'].setValues(displayedObject=myOdb)

# отримати логарифм довговічності у множині вузлів Set-1
var=(('LOGLife-Repeats', ELEMENT_NODAL), )
x1=readODB_set_(set='Set-1',var=var)

```

```

x1min = min([x[0][1] for x in x1]) # знайти мінімальне значення

# отримати коефіцієнт запасу у множині вузлів Set-1
var= (('FOS@Life=Infinite', ELEMENT_NODAL), )
x2=readODB_set_(set='Set-1',var=var)
# x2=[((3.0, 1.62),), ((3.0, 2.0),), ...,((час,значення),)]
x2min = min([x[0][1] for x in x2]) # знайти мінімальне значення

# отримати відсоток відмов у множині вузлів Set-1
var= (('%%Failure@Life=5E6-Repeats', ELEMENT_NODAL), )
x3=readODB_set_(set='Set-1',var=var)
x3max = max([x[0][1] for x in x3]) # знайти максимальне значення

...

# отримати напруження
x1=readODB_set2(set='Up',step='Step-
2',var=('S', 'S22'),pos=INTEGRATION_POINT)
x1max=findmax(x1) # знайти максимальне з усіх фреймів

# отримати контактний тиск
x2=readODB_set2(set='Nip',step='Step-1',var=('CPRESS', ''))
x2max=findmax(x2) # знайти максимальне з усіх фреймів

# отримати вертикальне переміщення
#x3=readODB_set2(set='Bot',step='Step-2',var=('U', 'U2'))

## отримати напруження
#var= (('S', INTEGRATION_POINT, ((INVARIANT, 'Mises'), )), )
#print readODB_set(set='Up',step='Step-2',var=var)
## отримати вертикальне переміщення
#var= (('U', NODAL, ((COMPONENT, 'U2'), )), )
#print readODB_set(set='Bot',step='Step-2',var=var)
'''

# записати дані у файл
writer.writerow([rad,load,x1min,x2min,x3max])
myOdb.close() # закрити базу даних результатів

csv_file.close() # закрити файл csv

```

ДОДАТОК У

Засоби оптимізації конструкції протектора

Таблиця У.1 – План експерименту і результати симуляції

№	b	r1	r2	L/2	a	y2	y1	y2n	y1n
1	0,02514	0,02159	0,02159	0,036591	0,37664	4204,253	2,443136	-0,746227364	0,537861
2	0,02514	0,02159	0,02159	0,036591	0,67002	3618,773	1,362398	-1,418270929	-1,07477
3	0,02514	0,02159	0,02159	0,053409	0,37664	5811,499	2,397281	1,098653089	0,469438
4	0,02514	0,02159	0,02159	0,053409	0,67002	5003,99	1,311918	0,171751743	-1,1501
5	0,02514	0,02159	0,03841	0,036591	0,37664	4110,578	2,452658	-0,853752246	0,552068
6	0,02514	0,02159	0,03841	0,036591	0,67002	3537,188	1,374097	-1,511918895	-1,05732
7	0,02514	0,02159	0,03841	0,053409	0,37664	5717,824	2,41894	0,991127185	0,501756
8	0,02514	0,02159	0,03841	0,053409	0,67002	4922,403	1,32	0,078102744	-1,13804
9	0,02514	0,03841	0,02159	0,036591	0,37664	4205,495	2,920309	-0,74480156	1,249878
10	0,02514	0,03841	0,02159	0,036591	0,67002	3619,83	1,64	-1,417057616	-0,66055
11	0,02514	0,03841	0,02159	0,053409	0,37664	5812,742	2,734789	1,100078846	0,973053
12	0,02514	0,03841	0,02159	0,053409	0,67002	5005,047	1,51	0,172965011	-0,85453
13	0,02514	0,03841	0,03841	0,036591	0,37664	4111,82	2,9415	-0,852326442	1,281498
14	0,02514	0,03841	0,03841	0,036591	0,67002	3538,245	1,652804	-1,510705582	-0,64144
15	0,02514	0,03841	0,03841	0,053409	0,37664	5719,066	2,754014	0,992552931	1,00174
16	0,02514	0,03841	0,03841	0,053409	0,67002	4923,46	1,519298	0,079316023	-0,84065
17	0,03986	0,02159	0,02159	0,036591	0,37664	4689,444	2,46011	-0,189299732	0,563189
18	0,03986	0,02159	0,02159	0,036591	0,67002	4034,958	1,343304	-0,940552945	-1,10326
19	0,03986	0,02159	0,02159	0,053409	0,37664	6296,69	2,398508	1,655579618	0,471267
20	0,03986	0,02159	0,02159	0,053409	0,67002	5420,173	1,298565	0,649468637	-1,17002
21	0,03986	0,02159	0,03841	0,036591	0,37664	4651,76	2,461645	-0,232555768	0,565478
22	0,03986	0,02159	0,03841	0,036591	0,67002	4002,121	1,356735	-0,978244266	-1,08322
23	0,03986	0,02159	0,03841	0,053409	0,37664	6259,006	2,390605	1,612323617	0,459475
24	0,03986	0,02159	0,03841	0,053409	0,67002	5387,337	1,314174	0,611777672	-1,14673
25	0,03986	0,03841	0,02159	0,036591	0,37664	4689,444	3,070677	-0,189299744	1,474251
26	0,03986	0,03841	0,02159	0,036591	0,67002	4034,958	1,647086	-0,940552945	-0,64997
27	0,03986	0,03841	0,02159	0,053409	0,37664	6296,69	2,922793	1,655579618	1,253585
28	0,03986	0,03841	0,02159	0,053409	0,67002	5420,174	1,580314	0,649469108	-0,74961
29	0,03986	0,03841	0,03841	0,036591	0,37664	4651,76	3,06592	-0,232555768	1,467153
30	0,03986	0,03841	0,03841	0,036591	0,67002	4002,121	1,665305	-0,978244266	-0,62279
31	0,03986	0,03841	0,03841	0,053409	0,37664	6259,006	2,920903	1,612323617	1,250764
32	0,03986	0,03841	0,03841	0,053409	0,67002	5387,337	1,597182	0,611778142	-0,72444
33	0,015	0,03	0,03	0,045	0,523333	4217,64	1,795891	-0,730861214	-0,42793
34	0,05	0,03	0,03	0,045	0,523333	5396,368	1,949208	0,622143574	-0,19916
35	0,0325	0,01	0,03	0,045	0,523333	4865,889	2,469018	0,013233251	0,57648
36	0,0325	0,05	0,03	0,045	0,523333	4865,905	2,506527	0,013250767	0,632449
37	0,0325	0,03	0,01	0,045	0,523333	4914,347	1,894558	0,068855389	-0,28071
38	0,0325	0,03	0,05	0,045	0,523333	4791,199	1,931681	-0,072500796	-0,22531
39	0,0325	0,03	0,03	0,025	0,523333	3086,55	2,051168	-2,029184368	-0,04702
40	0,0325	0,03	0,03	0,065	0,523333	6645,228	1,801189	2,055650192	-0,42003
41	0,0325	0,03	0,03	0,045	0,174444	5732,07	4,200831	1,007480051	3,160619
42	0,0325	0,03	0,03	0,045	0,872222	3999,708	0,980021	-0,981014215	-1,64534
43	0,0325	0,03	0,03	0,045	0,523333	4865,889	1,904474	0,013232918	-0,26591
44	0,0325	0,03	0,03	0,045	0,523333	4865,889	1,906308	0,013232918	-0,26317

Таблиця У.2 – Значення даних y , регресійної моделі $f(x)$ та відхилення

№	$\omega_1=1, \omega_2=-1$			$\omega_1=1, \omega_2=0$			$\omega_1=0, \omega_2=-1$		
	y	$f(x)$	$f(x)-y$	y	$f(x)$	$f(x)-y$	y	$f(x)$	$f(x)-y$
1	1,284088	1,232064	-0,05202	0,537861	0,51763	-0,02023	0,746227	0,746764	0,000537
2	0,343499	0,279873	-0,06363	-1,07477	-1,08826	-0,01349	1,418271	1,421052	0,002781
3	-0,62922	-0,68258	-0,05336	0,469438	0,432628	-0,03681	-1,09865	-1,09818	0,000477
4	-1,32185	-1,31336	0,008485	-1,1501	-1,17327	-0,02317	-0,17175	-0,17138	0,000369
5	1,40582	1,38297	-0,02285	0,552068	0,51763	-0,03444	0,853752	0,844012	-0,00974
6	0,454603	0,430779	-0,02382	-1,05732	-1,08826	-0,03095	1,511919	1,507112	-0,00481
7	-0,48937	-0,53167	-0,0423	0,501756	0,432628	-0,06913	-0,99113	-0,99672	-0,00559
8	-1,21614	-1,16246	0,053683	-1,13804	-1,17327	-0,03523	-0,0781	-0,08111	-0,00301
9	1,994679	1,944897	-0,04978	1,249878	1,256379	0,006502	0,744802	0,746764	0,001963
10	0,756512	0,68058	-0,07593	-0,66055	-0,69362	-0,03308	1,417058	1,421052	0,003994
11	-0,12703	0,030256	0,157282	0,973053	1,105156	0,132103	-1,10008	-1,09818	0,001903
12	-1,02749	-0,91266	0,114836	-0,85453	-0,84485	0,009679	-0,17297	-0,17138	0,001582
13	2,133824	2,095803	-0,03802	1,281498	1,256379	-0,02512	0,852326	0,844012	-0,00831
14	0,869264	0,831486	-0,03778	-0,64144	-0,69362	-0,05218	1,510706	1,507112	-0,00359
15	0,009187	0,181162	0,171975	1,00174	1,105156	0,103416	-0,99255	-0,99672	-0,00416
16	-0,91997	-0,76175	0,158218	-0,84065	-0,84485	-0,0042	-0,07932	-0,08111	-0,00179
17	0,752488	0,83272	0,080232	0,563189	0,576713	0,013525	0,1893	0,183867	-0,00543
18	-0,16271	-0,11947	0,04324	-1,10326	-1,02918	0,074082	0,940553	0,939701	-0,00085
19	-1,18431	-1,23259	-0,04828	0,471267	0,491711	0,020444	-1,65558	-1,66107	-0,00549
20	-1,81949	-1,86338	-0,04389	-1,17002	-1,11418	0,055837	-0,64947	-0,65273	-0,00327
21	0,798034	0,927898	0,129864	0,565478	0,576713	0,011235	0,232556	0,234957	0,002402
22	-0,10498	-0,02429	0,080685	-1,08322	-1,02918	0,054041	0,978244	0,979604	0,00136
23	-1,15285	-1,13741	0,015436	0,459475	0,491711	0,032236	-1,61232	-1,60577	0,006554
24	-1,75851	-1,7682	-0,00969	-1,14673	-1,11418	0,032547	-0,61178	-0,60862	0,003159
25	1,663551	1,545553	-0,118	1,474251	1,361492	-0,11276	0,1893	0,183867	-0,00543
26	0,290581	0,281236	-0,00934	-0,64997	-0,58851	0,061461	0,940553	0,939701	-0,00085
27	-0,40199	-0,51976	-0,11776	1,253585	1,210268	-0,04332	-1,65558	-1,66107	-0,00549
28	-1,39908	-1,46267	-0,06359	-0,74961	-0,73973	0,009872	-0,64947	-0,65273	-0,00326
29	1,699708	1,640731	-0,05898	1,467153	1,361492	-0,10566	0,232556	0,234957	0,002402
30	0,355458	0,376413	0,020956	-0,62279	-0,58851	0,034275	0,978244	0,979604	0,00136
31	-0,36156	-0,42458	-0,06302	1,250764	1,210268	-0,0405	-1,61232	-1,60577	0,006554
32	-1,33621	-1,36749	-0,03128	-0,72444	-0,73973	-0,0153	-0,61178	-0,60862	0,00316
33	0,302929	0,166522	-0,13641	-0,42793	-0,36358	0,064349	0,730861	0,738782	0,007921
34	-0,8213	-0,85902	-0,03772	-0,19916	-0,16838	0,030781	-0,62214	-0,62335	-0,0012
35	0,563247	0,434041	-0,12921	0,57648	0,859609	0,283129	-0,01323	-0,0127	0,000536
36	0,619198	0,751464	0,132266	0,632449	0,664302	0,031854	-0,01325	-0,0127	0,000554
37	-0,34956	-0,30379	0,045775	-0,28071	-0,26598	0,014725	-0,06886	-0,06817	0,000681
38	-0,15281	-0,02618	0,126627	-0,22531	-0,26598	-0,04067	0,072501	0,071385	-0,00112
39	1,982167	1,92202	-0,06015	-0,04702	-0,02198	0,025037	2,029184	2,034029	0,004844
40	-2,47568	-2,25199	0,223686	-0,42003	-0,35983	0,0602	-2,05565	-2,05062	0,005029
41	2,153139	2,143922	-0,00922	3,160619	3,175576	0,014957	-1,00748	-1,00654	0,000941
42	-0,66433	-0,91896	-0,25464	-1,64534	-1,72875	-0,08341	0,981014	0,981145	0,000131
43	-0,27914	-0,16499	0,114156	-0,26591	-0,26598	-7,2E-05	-0,01323	-0,0127	0,000536
44	-0,27641	-0,16499	0,11142	-0,26317	-0,26598	-0,00281	-0,01323	-0,0127	0,000536

Код програми та SOLIDWORKS-модель доступні в GitHub (vkopecy/protector-for-sucker-rod: SOLIDWORKS model of protector for sucker rod and programs for optimization. URL: <http://github.com/vkopecy/protector-for-sucker-rod>).

Лістинг У.1 – Код VBA-макросу для обчислення середньої площі тертя

Sub Makro()

```

Const swDocPART = 1
Const swDocASSEMBLY = 2
Const swDocDRAWING = 3

Dim swApp As Object
Dim Part As Object
Dim face As Object
'Dim massProps As Variant
Dim R1, R2, L, a, b, f As Double
Dim i As Integer
R1min = 10: R1max = 50: R1step = 5
R2min = 10: R2max = 50: R2step = 5
Lmin = 20: Lmax = 60: Lstep = 10
amin = 10: amax = 50: astep = 5
bmin = 15: bmax = 50: bstep = 5
fmin = 1: fmax = 10: fstep = 2
MyPath = CurDir
MyPath = "C:\Protector"

Set swApp = CreateObject("SldWorks.Application")
Set Part = swApp.OpenDoc(MyPath + "\ModelVBA.SLDPRT", swDocPART)
If Part Is Nothing Then
    Exit Sub
Else
    Set Part = swApp.ActivateDoc("ModelVBA.SLDPRT")
End If
i = 3
For R1 = R1min To R1max Step R1step
For R2 = R2min To R2max Step R2step
For L = Lmin To Lmax Step Lstep
For a = amin To amax Step astep
For b = bmin To bmax Step bstep
For f = fmin To fmax Step fstep
Part.Parameter("D1@Filet1").SystemValue = R1 / 1000
Part.Parameter("D1@Filet2").SystemValue = R2 / 1000
Part.Parameter("D1@Extrude2").SystemValue = L / 1000
Part.Parameter("D1@c_sketch").SystemValue = (a * 3.14) / 180
Part.Parameter("D3@schemfer").SystemValue = b / 1000

```

```
Part.Parameter("D1@w_sketch").SystemValue = 0.056 - (f / 1000)
Part.EditRebuild
Cells(i, 1).Value = R1: Cells(2, 1).Value = R1
Cells(i, 2).Value = R2: Cells(2, 2).Value = R2
Cells(i, 3).Value = L: Cells(2, 3).Value = L
Cells(i, 4).Value = a: Cells(2, 4).Value = a
Cells(i, 5).Value = b: Cells(2, 5).Value = b
Cells(i, 6).Value = f: Cells(2, 6).Value = f

'massProps = Part.GetMassProperties
'Cells(i, 7).Value = 2 * massProps(3)

Set face = Part.GetEntityByName(1, 2)
Cells(i, 8).Value = face.GetArea
Set face = Part.GetEntityByName(2, 2)
Cells(i, 9).Value = 2 * face.GetArea
i = i + 1
Next f
Next b
Next a
Next L
Next R2
Next R1
End Sub
```

ДОДАТОК Ф

Абстрактна модель ІС

Таблиця Ф.1 – Опис класів елементів ІС на інших рівнях ієрархії

Рівень, біполярні множини ознак	Назва класу, приклади елементів та їх інформаційної продукції
<p><i>Рівень 1.1</i> 0: {1, 2, 3, 4}</p> <p>1: {5, 6, 7, 8}</p>	<p><i>Етап проектів.</i> Центр етапу відповідає максимуму <i>A, M</i>. Підтримка проектування виробу. Дослідно-конструкторська та дослідно-технологічна робота. Системи підтримки прийняття проектних рішень. Системи автоматизованого проектування (системи CAD і CAE).</p> <p><i>Етап реалізацій.</i> Центр етапу відповідає максимуму <i>O, Pe</i>. Підтримка виробництва і експлуатації. Управління ланцюгом постачань (SCM-системи). Створення супроводжуючої документації (IETM-системи), навчання персоналу.</p>
<p><i>Рівень 1.2</i> 0: {8, 1, 2, 3}</p> <p>1: {4, 5, 6, 7}</p>	<p><i>Теоретичний етап.</i> Теоретичні методи. Науково дослідна робота над створенням продукції.</p> <p><i>Практичний етап.</i> Емпіричні методи. Застосування результатів науково-дослідної роботи.</p>
<p><i>Рівень 1.3</i> 0: {7, 8, 1, 2}</p> <p>1: {3, 4, 5, 6}</p>	<p><i>Етап концепцій.</i> Центр етапу відповідає максимуму <i>I, B</i>. Абстрактні моделі, гіпотези, пошук рішень.</p> <p><i>Етап технологічний.</i> Центр етапу відповідає максимуму <i>P, K</i>. Конкретні рішення. Інтегровані автоматизовані системи управління виробництвом. Управління ресурсами підприємства (ERP-системи).</p>
<p><i>Рівень 1.4</i> 0: {6, 7, 8, 1}</p> <p>1: {2, 3, 4, 5}</p>	<p><i>Етап експлуатації і вимог.</i> Експлуатаційні моделі виробу. Інтегровані автоматизовані системи управління експлуатацією. Дослідження експлуатації. Експлуатаційна документація. Управління відносинами з клієнтами (CRM-системи).</p> <p><i>Етап конструкторсько-технологічний.</i> Конструкторсько-технологічні моделі. Конструкторсько-технологічна документація. Технологічна підготовка (системи CAD і CAM). Поставлення продукції на виробництво (підготовка виробництва, освоєння виробництва).</p>
<p><i>Рівень 2.1</i> 0: {1, 2, 5, 6}</p> <p>1: {3, 4, 7, 8}</p>	<p><i>Класи.</i> Подання об'єктів абстрактно, як їх класи. {1, 2} - <i>етап науково-дослідницький</i> (пошуково-аналітичний, дослідження та обґрунтування розроблення). {5, 6} - <i>етап серійного виробництва і дослідження збуту</i>. <i>Об'єкти.</i> Подання об'єктів конкретно, як екземпляри класів. {3, 4} - <i>етап підготовки виробництва</i> (швидке прототипування, дослідне виробництво, САМ-системи). {7, 8} - <i>етап збуту і експлуатації</i>.</p>

Кінець таблиці Ф.1

Рівень, біполярні множини ознак	Назва класу, приклади елементів та їх інформаційної продукції
<p><i>Рівень 2.2</i> 0: {8, 1, 4, 5}</p> <p>1: {2, 3, 6, 7}</p>	<p><i>Ірраціональні етапи.</i> Стохастичні, евристичні методи, еволюційні алгоритми, нейронні мережі, когнітивне моделювання, МАІС. {8, 1} - <i>етап генерації ідей і вимог</i> (виявлення проблеми, розробки варіантів і моделей прийняття рішення, ескізний проект). {4, 5} - <i>етап прийняття рішення і контролю</i> (організаційно-планова підготовка виробництва). <i>Раціональні етапи.</i> Детерміновані методи, МФПС, експертні системи логічного виведення. {2, 3} - <i>етап аналізів і мотивів</i> (пошук рішення і оцінка альтернатив, технічний проект). {6, 7} - <i>етап реалізації та її оцінки</i> (експлуатація, економічні методи оцінювання, аналіз витрат і вигод).</p>
<p><i>Рівень 3.1</i> 0: {1, 3, 5, 7} 1: {2, 4, 6, 8}</p>	<p><i>Процеси.</i> Орієнтація на процеси в ІС. Цілі нечіткі. Каузальний підхід. <i>Цілі.</i> Орієнтація на цілі в ІС. Процес не важливий. Аксіологічний підхід. Початок ЖЦ завжди має ознаку 0, а кінець - 1 (рис. 2).</p>
<p>0: {1а, 2а, 3а, 4а, 5а, 6а, 7а, 8а} 1: {1б, 2б, 3б, 4б, 5б, 6б, 7б, 8б}</p>	<p><i>Потік 1.</i> Загальні перспективи. Проект першої черги. Статичні властивості об'єктів. Статичні моделі або дискретні динамічні моделі. Зосередження на цілях. <i>Потік 2.</i> Точні прогнози. Проект другої черги. Динамічні властивості об'єктів. Динамічні неперервні моделі. Зосередження на методах досягнення цілей.</p>
<p>0: {1а, 2б, 3б, 4а, 5а, 6б, 7б, 8а} 1: {1б, 2а, 3а, 4б, 5б, 6а, 7а, 8б}</p>	<p><i>Властивості.</i> Опис і моделі властивостей об'єктів. <i>Відношення.</i> Опис і моделі відношень між об'єктами.</p>
<p>0: {1а, 2а, 3б, 4б, 5б, 6б, 7а, 8а} 1: {1б, 2б, 3а, 4а, 5а, 6а, 7б, 8б}</p>	<p><i>Зміст.</i> Евристичні методи оцінки перспектив або прогнозів. Зміст понять. Абстрактні поняття. <i>Обсяг.</i> Евристичні методи висування вимоги і прийняття рішення. Обсяг понять. Конкретні поняття.</p>
<p>0: {1а, 2а, 3а, 4а, 5б, 6б, 7б, 8б} 1: {1б, 2б, 3б, 4б, 5а, 6а, 7а, 8а}</p>	<p><i>Кількість.</i> Детерміновані методи аналізу і синтезу. Формально-логічні методи. Кількісні методи. Об'єктивні оцінки. <i>Якість.</i> Детерміновані методи постановки цілей або оцінки результатів. Методи експертного оцінювання. Якісні методи. Суб'єктивні оцінки.</p>

Таблиця Ф.2 – Бінарні відношення між елементами класів рівня 2.1

{1,2},{1,2}	{3,4},{3,4}	{5,6},{5,6}	{7,8},{7,8}	1.Тотожність
{1,2},{3,4}	{3,4},{5,6}	{5,6},{7,8}	{7,8},{1,2}	2.Пряма послідовність
{1,2},{5,6}	{3,4},{7,8}	{5,6},{1,2}	{7,8},{3,4}	3.Антипослідовність
{1,2},{7,8}	{3,4},{1,2}	{5,6},{3,4}	{7,8},{5,6}	4.Обернена послідовн.

Таблиця Ф.3 – Бінарні відношення між елементами класів I, B, \dots, Pe

I,I	B,B	A,A	M,M	P,P	K,K	O,O	Pe,Pe	Тотожність 1
I,B	B,I	A,M	M,A	P,K	K,P	O,Pe	Pe,O	Синхронність 2
I,A	B,M	A,P	M,K	P,O	K,Pe	O,I	Pe,B	Пряма послідовність 1
I,M	B,A	A,K	M,P	P,Pe	K,O	O,B	Pe,I	Пряма послідовність 2
I,P	B,K	A,O	M,Pe	P,I	K,B	O,A	Pe,M	Антипослідовність 1
I,K	B,P	A,Pe	M,O	P,B	K,I	O,M	Pe,A	Антипослідовність 2
I,O	B,Pe	A,I	M,B	P,A	K,M	O,P	Pe,K	Обернена послідовність 1
I,Pe	B,O	A,B	M,I	P,M	K,A	O,K	Pe,P	Обернена послідовність 2

ДОДАТОК X

Код експертної системи з проблем надійності та довговічності різьбових з'єднань

Файл `assertedFacts.csv`, що доступний в GitHub (`vkopey/TreadsKB: Expert system for problems of reliability and durability of threaded connections`. URL: <http://github.com/vkopey/TreadsKB>), згенерований експертною системою автоматично та містить факти бази знань (461 факт) у вигляді триплетів (суб'єкт; предикат; об'єкт). Для зменшення об'єму цього коду в лістингу X.1 використані такі позначення:

- 1 – Base\Фактор\Конструкційні елементи;
- 2 – Base\Фактор\Покриття;
- 3 – Base\Фактор\Пошкодження;
- 4 – Base\Фактор\Геометричні параметри;
- 5 – Base\Фактор\Матеріал;
- 6 – Base\Фактор\Складання;
- 7 – Base\Фактор\Деталі;
- 8 – Base\Фактор\Міцність;
- 9 – Base\Фактор\Жорсткість;
- 10 – Base\Фактор\Технологія;
- 11 – Base\Фактор\Тертя;
- 12 – Base\Фактор\Загальні напрямки підвищення надійності;
- 13 – Base\Фактор\Середовище;
- 14 – Base\Фактор\Напруження;
- 15 – Base\Фактор\Температура;
- 16 – Base\Фактор\Масштабний фактор;
- 17 – Base\Фактор\Навантаження;
- 18 – Base\Фактор\Концентрація напружень;
- 19 – Base\Фактор\Пошкодження при транспортуванні;
- 20 – Base\Фактор\Стопоріння;
- 21 – Base\Фактор\Згвинчуваність;
- 22 – Base\Фактор\Герметичність;
- 23 – Base\Фактор\Маса;
- 24 – Base\Фактор\Залишкові напруження;
- 25 – Base\Фактор\Рухомість з'єднання;
- 26 – Base\Фактор\Змащення;
- 27 – Base\Фактор\Пошкодження при транспортуванні;
- 28 – Base\Фактор\Розкриття стику.

Редактор коду експертної системи доступний в GitHub (`vkopey/TreePyKB: Knowledge base editor`. URL: <http://github.com/vkopey/TreePyKB>). Готовий до виконання код експертної системи, зібраний в один модуль `generatedKBcode.py`, доступний повністю у GitHub (`vkopey/TreadsKB: Expert system for problems of`

reliability and durability of threaded connections. URL: <http://github.com/vkopey/TreadsKB>). Він містить усі факти бази знань, що відповідають триплетам в лістингу X.1, правила і машину виведення та запити. Код цього модуля з невеликою частиною фактів бази знань наведено у лістингу X.2.

Лістинг X.1 – assertedFacts.csv (триплети бази знань)

```

1\Гайка з осьовими прорізами;isCause;18\Нерівномірне навантаження по виткам
різьби\Низьке
1\Герметизуючі елементи;isCause;22
1\Зарізьбова канавка;isCause;8\За статичного навантаження\Низька
1\Зарізьбова канавка;isCause;8\За циклічного навантаження\Висока
1\Опорні поверхні\Конічні;isCause;14\Згину\Зменшення
1\Опорні поверхні\Сферичні;isCause;14\Згину\Зменшення
1\Різьба\Асиметрична різьба;isCause;18\Нерівномірне навантаження по виткам
різьби\Низьке
1\Різьба\Збіг;isCause;8\За статичного навантаження\Висока
1\Різьба\Збіг;isCause;8\За циклічного навантаження\Низька
1\Різьба\Змінний середній діаметр різьби гайки;isCause;18\Нерівномірне
навантаження по виткам різьби\Низьке
1\Різьба\Крок\Змінний крок;isCause;8\За циклічного навантаження\Висока
1\Різьба\Крок\Неоднаковий крок\Збільшення кроку
гайки;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке
1\Різьба\Крок\Неоднаковий крок\Зменшення кроку
болта;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке
1\Різьба\Напряв гвинтової лінії\Ліва;Not;1\Різьба\Напряв гвинтової
лінії\Права
1\Різьба\Обтиск останніх витків різьби муфти;isCause;18\Нерівномірне
навантаження по виткам різьби\Низьке
1\Різьба\Податливі витки\Перші витки гайки;isCause;18\Нерівномірне
навантаження по виткам різьби\Низьке
1\Різьба\Розтиск останніх витків різьби ніпеля;isCause;18\Нерівномірне
навантаження по виткам різьби\Низьке
1\Різьба\Розтиск перших витків різьби муфти;isCause;18\Нерівномірне
навантаження по виткам різьби\Низьке
1\Різьба\Фаски\Зріз перших витків різьби муфти;isCause;18\Нерівномірне
навантаження по виткам різьби\Низьке
1\Різьба\Форма профілю\Кругла;isCause;8\За циклічного навантаження\Висока
1\Різьба\Форма профілю\Прямокутна;isCause;18\Нерівномірне навантаження по
виткам різьби
1\Різьба\Форма профілю\Спеціальна;isCause;8\За циклічного
навантаження\Висока
1\Різьба\Форма профілю\Трикутна;isCause;3\Залишкова деформація\Деформація
тіла гайки
1\Різьба\Форма профілю\Упорна;isCause;18\Нерівномірне навантаження по
виткам різьби

```

1\Різьба\Форма профілю\Упорна;isCause;3\Залишкова деформація\Деформація тіла гайки\Зменшення

1\Різьба\Характер поверхні\Конічна;isCause;22

1\Розташування різьбової частини болта\Вільні витки болта;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке

1\Розташування різьбової частини болта\Різьба болта утоплена в гайку;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке

1\Форма головки болта\Двухрадіусна галтель;isCause;18\Під головкою болта\Низька

1\Форма головки болта\Оптимальна;isCause;8\За циклічного навантаження\Висока

1\Центруючі елементи;isCause;14\Згину\Зменшення

2;isCause;3\Заїдання\Зменшення

2\Багат шарове мідь-нікель, хром;isCause;15\Робоча температура\Максимальна\600

2\Багат шарове мідь-нікель;isCause;15\Робоча температура\Максимальна\600

2\Електрохімічна обробка;isCause;5\Насичення атомарним воднем

2\Зносостійкі;isCause;12\Захист від механічного спрацювання деталей

2\Кадмієве з хроматуванням;isCause;11\На різьбі\Мале

2\Кадмієве з хроматуванням;isCause;15\Робоча температура\Максимальна\200

2\Кадмієве з хроматуванням;isCause;2\Електрохімічна обробка

2\Корозійностійкі;isCause;3\Корозійна\Низька

2\Мідне;isCause;11\Зміна коефіцієнта тертя під час повторних згинчуваннях\Збільшується

2\Мідне;isCause;15\Робоча температура\Максимальна\600

2\Нікелеве;isCause;11\Зміна коефіцієнта тертя під час повторних згинчуваннях\Збільшується

2\Нікелеве;isCause;15\Робоча температура\Максимальна\900

2\Оксидне анодизаційне з хроматуванням;isCause;15\Робоча температура\Максимальна\200

2\Оксидне з кислих розчинів;isCause;15\Робоча температура\Максимальна\200

2\Оксидне;isCause;11\Зміна коефіцієнта тертя під час повторних згинчуваннях\Збільшується

2\Оксидне;isCause;15\Робоча температура\Максимальна\200

2\Оксидне;isCause;24\Стиску

2\Олов'яне;isCause;11\На різьбі\Мале

2\Олов'яне;isCause;15\Робоча температура\Максимальна\150

2\Оптимальне;isCause;8\За циклічного навантаження\Висока

2\Пластичне;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке

2\Свинцеве;isCause;11\На різьбі\Мале

2\Срібне;isCause;11\На різьбі\Мале

2\Срібне;isCause;15\Робоча температура\Максимальна\600

2\Травлення під час нанесення покриттів;isCause;5\Насичення атомарним воднем

2\Фосфатне;isCause;15\Робоча температура\Максимальна\200

2\Фосфатне;isCause;2\Травлення під час нанесення покриттів

2\Цинкове з хроматуванням;isCause;15\Робоча температура\Максимальна\300

2\Цинкове;isCause;11\Зміна коефіцієнта тертя під час повторних згинчуваннях\Збільшується

2\Цинкове;isCause;15\Робоча температура\Максимальна\200

2\Цинкове;isCause;2\Електрохімічна обробка

3\Від високошвидкісного навантаження;SubClassOf;3
 3\Втомна тріщина;isCause;3\Корозійна
 3\Втомна тріщина;SubClassOf;3
 3\Втомна тріщина\В останньому витку муфти;SubClassOf;3\Втомна тріщина
 3\Втомна тріщина\В першому витку ніпеля;SubClassOf;3\Втомна тріщина
 3\Втомна тріщина\Під головкою болта;SubClassOf;3\Втомна тріщина
 3\Заїдання;isCause;11\На різьбі\Велике
 3\Заїдання;isCause;11\На упорному бурті\Велике
 3\Заїдання;isCause;14\Кручення
 3\Заїдання;isCause;3\Знос
 3\Заїдання;Not;3\Заїдання\Зменшення
 3\Заїдання;SubClassOf;3
 3\Залишкова деформація;isCause;3\Заїдання
 3\Залишкова деформація;isCause;3\Самовідгвинчування
 3\Залишкова деформація;SubClassOf;3
 3\Залишкова деформація\Деформація тіла гайки;Not;3\Залишкова
 деформація\Деформація тіла гайки\Зменшення
 3\Залишкова деформація\Деформація тіла гайки;SubClassOf;3\Залишкова
 деформація
 3\Залишкова деформація\Зріз витків;Not;3\Залишкова деформація\Зріз
 витків\Зменшення
 3\Залишкова деформація\Зріз витків;SubClassOf;3\Залишкова деформація
 3\Залишкова деформація\Зріз стержня;Not;3\Залишкова деформація\Зріз
 стержня\Зменшення
 3\Залишкова деформація\Зріз стержня;SubClassOf;3\Залишкова деформація
 3\Залишкова деформація\Обрив стержня;SubClassOf;3\Залишкова деформація
 3\Знос;isCause;22\Низька
 3\Знос;isCause;3\Залишкова деформація\Зріз витків
 3\Знос;SubClassOf;3
 3\Знос\Абразивний;SubClassOf;3\Знос
 3\Знос\Адгезійний;SubClassOf;3\Знос
 3\Знос\Втомний;SubClassOf;3\Знос
 3\Знос\Ерозійний;SubClassOf;3\Знос
 3\Знос\Зменшення;Not;3\Знос
 3\Знос\Механохімічний;SubClassOf;3\Знос
 3\Знос\Фретинг-корозійний;SubClassOf;3\Знос
 3\Знос\Фретинговий;SubClassOf;3\Знос
 3\Корозійна;isCause;18
 3\Корозійна;isCause;22\Низька
 3\Корозійна;isCause;3\Заїдання
 3\Корозійна;isCause;3\Знос
 3\Корозійна;isCause;8\Низька
 3\Корозійна;SubClassOf;3
 3\Корозійна\Низька;Not;3\Корозійна
 3\Корозійне розтріскування;isCause;3\Крихкий злам
 3\Корозійне розтріскування;isCause;8\За статичного навантаження\Низька
 3\Корозійне розтріскування;SubClassOf;3
 3\Крихкий злам;SubClassOf;3
 3\Крихкий злам\Уповільнене крихке руйнування;SubClassOf;3\Крихкий злам
 3\Крихкий злам\Уповільнене крихке руйнування\Низьке;Not;3\Крихкий
 злам\Уповільнене крихке руйнування

3\Розгерметизація;SubClassOf;3
 3\Розгерметизація\Заобігання;Not;3\Розгерметизація
 3\Самовідгвинчування;isCause;3\Залишкова деформація\Зріз витків
 3\Самовідгвинчування;SubClassOf;3
 3\Самовідгвинчування\Зменшення;Not;3\Самовідгвинчування
 4\Допуск розміру\Великий;isCause;4\Посадка\З зазором
 4\Допуск розміру\Малий;isEffect;6\Селекційне складання
 4\Допуск розміру\Малий;Not;4\Допуск розміру\Великий
 4\Допуск форми\Перекоос опорних поверхонь;isCause;14\Згину
 4\Допуск форми\Перекоос опорних поверхонь;isCause;3\Крихкий злам\Уповільнене крихке руйнування
 4\Допуск форми\Перекоос опорних поверхонь\Високоміцних сталей;isCause;8\За циклічного навантаження\Низька
 4\Допуск форми\Перекоос опорних поверхонь\Зменшення;Not;4\Допуск форми\Перекоос опорних поверхонь
 4\Допуск форми\Перекоос опорних поверхонь\Зменшення;Not;4\Допуск форми\Перекоос опорних поверхонь\Великий
 4\Допуск форми\Перекоос осей;isCause;14\Згину
 4\Допуск форми\Перекоос осей;Not;4\Допуск форми\Перекоос осей\Малий
 4\Допуск форми\Чутливість до перекоосу;isCause;8\За статичного навантаження\Низька
 4\Допуск форми\Чутливість до перекоосу;isCause;8\За циклічного навантаження\Низька
 4\Допуск форми\Чутливість до перекоосу;Not;4\Допуск форми\Нечутливість до перекоосу
 4\Посадка\З зазором;isCause;25
 4\Посадка\З зазором\Зменшення діаметральних зазорів;isCause;18\Низька
 4\Посадка\З зазором\Зменшення діаметральних зазорів;isCause;3\Заїдання
 4\Посадка\З зазором\Зменшення діаметральних зазорів;isCause;8\За циклічного навантаження\Висока
 4\Посадка\З натягом;isCause;25\Низька
 4\Посадка\З натягом;isCause;3\Заїдання
 4\Посадка\З натягом;isCause;6\Момент згвинчування\Неоптимальний\Великий
 4\Посадка\Оптимальна;isCause;18\Низька
 4\Посадка\Оптимальна;isCause;8\За циклічного навантаження\Висока
 4\Посадка\Перехідна;isCause;25\Низька
 4\Розмір\Відношення діаметру до кроку\Велике;isCause;18\Нерівномірне навантаження по виткам різьби
 4\Розмір\Діаметр зарізьбової канавки\Великий;isCause;8\За статичного навантаження\Висока
 4\Розмір\Діаметр зарізьбової канавки\Великий;isCause;8\За циклічного навантаження\Низька
 4\Розмір\Діаметр зарізьбової канавки\Великий;Not;4\Розмір\Діаметр зарізьбової канавки\Малий
 4\Розмір\Діаметр різьби\Великий;isCause;3\Залишкова деформація\Зріз витків\Зменшення
 4\Розмір\Діаметр різьби\Великий\З концентратором напружень;isCause;8\За циклічного навантаження\Низька
 4\Розмір\Довжина гайки\Велика;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке

4\Розмір\Довжина гайки\Велика;isCause;3\Залишкова деформація\Зріз витків\Зменшення
 4\Розмір\Довжина головки болта\Велика;isCause;18\Під головкою болта\Низька
 4\Розмір\Довжина зарізьбової канавки\Велика;isCause;9\Болта\Мала
 4\Розмір\Довжина згвинчування\Мала;isCause;3\Залишкова деформація\Зріз витків
 4\Розмір\Крок різьби\Великий;isCause;3\Залишкова деформація\Зріз витків\Зменшення
 4\Розмір\Крок різьби\Оптимальний;isCause;8\За циклічного навантаження\Висока
 4\Розмір\Кут профілю\Великий;isCause;3\Залишкова деформація\Деформація тіла гайки
 4\Розмір\Кут профілю\Великий;Not;4\Розмір\Кут профілю\Малий
 4\Розмір\Кут профілю\Малий;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке
 4\Розмір\Кут профілю\Малий;isCause;3\Залишкова деформація\Зріз витків
 4\Розмір\Перекриття витків\Велике;isCause;8
 4\Розмір\Перекриття витків\Мале;isCause;3\Залишкова деформація\Зріз витків
 4\Розмір\Перекриття витків\Мале;Not;4\Розмір\Перекриття витків\Велике
 4\Розмір\Радіус впадин різьби\Збільшення;isCause;18\Низька
 4\Розмір\Радіус впадин різьби\Збільшення;isCause;3\Крихкий злам\Уповільнене крихке руйнування\Низьке
 4\Розмір\Типорозмір різьби\Великий;isCause;16\Великі розміри
 4\Розмір\Типорозмір різьби\Великий;isCause;23\Велика
 4\Розмір\Типорозмір різьби\Великий;Not;4\Розмір\Типорозмір різьби\Малий
 4\Розмір\Товщина гайки\Велика;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке
 4\Розмір\Товщина гайки\Велика;isCause;8
 4\Розмір\Товщина гайки\Велика;Not;4\Розмір\Товщина гайки\Мала
 4\Розмір\Товщина гайки\Мала;isCause;1\Різьба\Форма профілю\Упорна
 4\Шорсткість\Висока;isCause;18
 4\Шорсткість\Висока;isCause;3\Заїдання
 4\Шорсткість\Висока;isCause;3\Знос
 4\Шорсткість\Висока;isCause;3\Корозійна
 4\Шорсткість\Низька;isCause;18\Низька
 4\Шорсткість\Низька;isCause;3\Самовідгвинчування\Зменшення
 4\Шорсткість\Низька;Not;4\Шорсткість\Висока
 4\Шорсткість\Низька\Леговані сталі;isCause;18\Низька
 4\Шорсткість\Низька\Леговані сталі;isCause;8\За циклічного навантаження\Висока
 5\Алюмінієві сплави;isCause;23\Мала
 5\Алюмінієві сплави\Гайки;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке
 5\Берилієві сплави;isCause;10\Метод виготовлення\Накатування
 5\Берилієві сплави;isCause;23\Мала
 5\Берилієві сплави;isCause;3\Корозійна\Низька
 5\Берилієві сплави;isCause;5\Алюмінієві сплави\Гайки
 5\Берилієві сплави;isCause;5\Токсичність
 5\Берилієві сплави;isCause;5\Чутливість до концентрації напружень
 5\Бесемєрівська сталь;isCause;5\Наявність азоту
 5\Бесемєрівська сталь;isCause;5\Наявність фосфору

5\Високоміцні сталі болтів;isCause;15\Робоча температура\Висока
 5\Високоміцні сталі болтів;isCause;3\Залишкова деформація\Зріз
 стержня\Зменшення
 5\Високоміцні сталі болтів;isCause;3\Корозійне розтріскування
 5\Високоміцні сталі болтів;isCause;3\Самовідгвинчування\Зменшення
 5\Високоміцні сталі болтів;isCause;4\Допуск форми\Чутливість до перекосу
 5\Високоміцні сталі болтів;isCause;5\Пластичність\Низька
 5\Високоміцні сталі болтів;isCause;8\За циклічного навантаження\Висока
 5\Високоміцні сталі болтів\Границя міцності\1100-1400;isCause;15\Робоча
 температура\Максимальна\400
 5\Високоміцні сталі болтів\Границя міцності\1100-1400;isCause;8\За
 циклічного навантаження\Висока
 5\Високоміцні сталі болтів\Границя міцності\1100-1600;isCause;8\За осьового
 навантаження
 5\Високоміцні сталі болтів\Границя міцності\1800-2100;isCause;8\За
 зрізуючого навантаження
 5\Високоміцні сталі болтів\За високих моментів згвинчування і агресивного
 середовища;isCause;3\Корозійне розтріскування
 5\Високоміцні сталі болтів\За високих моментів згвинчування і агресивного
 середовища;And;5\Вуглець фосфор азот на границях зерен
 5\Високоміцні сталі болтів\За високих моментів згвинчування і агресивного
 середовища;And;6\Момент згвинчування\Неоптимальний\Великий
 5\Високоміцні сталі болтів\За високих моментів згвинчування і агресивного
 середовища;And;13\Корозійне
 5\Високоміцні сталі болтів\Зарізьбова канавка;isCause;8
 5\Високоміцні сталі болтів\Зниження водневої крихкості;isCause;8
 5\Високоміцні сталі болтів\Кадміювання;isCause;8
 5\Високоміцні сталі болтів\Накатування;isCause;10\Метод
 виготовлення\Накатування\Стійкість інструмента\Низька
 5\Високоміцні сталі болтів\Накатування;isCause;8\За циклічного
 навантаження\Висока
 5\Високоміцні сталі болтів\Полірування;isCause;8
 5\Високоміцні сталі болтів\Спеціальна термічна обробка високоміцних болтів
 з легованих сталей;isCause;8
 5\Високоміцні сталі болтів\Сталь07X16H6;isCause;15\Робоча
 температура\Мінімальна\ -196
 5\Високоміцні сталі болтів\Сталь07X16H6;isCause;5\Пластичність\Висока
 5\Високоміцні сталі болтів\Сталь07X16H6;isCause;5\Холодноламкість\Низька
 5\Високоміцні сталі болтів\Сталь07X16H6;isCause;8\За циклічного
 навантаження\Висока
 5\Високоміцні сталі болтів\Сталь07X16H6;SubClassOf;5\Високоміцні сталі
 болтів
 5\Високоміцні сталі болтів\Сталь07X16H6\Обробка
 холодом;isCause;5\Чутливість до концентрації напружень\Низька
 5\Високоміцні сталі болтів\Сталь1X15H4AM3-
 Ш;isCause;5\Холодноламкість\Низька
 5\Високоміцні сталі болтів\Сталь1X15H4AM3-Ш;isCause;8\За циклічного
 навантаження\Висока
 5\Високоміцні сталі болтів\Сталь1X15H4AM3-Ш;SubClassOf;5\Високоміцні сталі
 болтів

5\Високоміцні сталі болтів\Сталь1Х15Н4АМ3-Ш\Обробка
 холодом;isCause;5\Чутливість до концентрації напружень\Низька
 5\Вуглець у поверхневих шарах;Not;5\Вуглець у поверхневих шарах\Зменшення
 5\Вуглець у поверхневих шарах\Неоптимальний;isCause;8\За циклічного
 навантаження\Низька
 5\Вуглець у поверхневих шарах\Оптимальний;isCause;8\За циклічного
 навантаження\Висока
 5\Вуглець у поверхневих шарах\Оптимальний;Not;5\Вуглець у поверхневих
 шарах\Неоптимальний
 5\Вуглець фосфор азот на границях зерен;isCause;3\Корозійне розтріскування
 5\Газонасочений шар;isCause;3\Крихкий злам\Уповільнене крихке руйнування
 5\Жароміцні сплави\Висока робоча температура;isCause;3\Заїдання
 5\Жароміцні сплави\Висока робоча температура;isCause;4\Допуск
 форми\Чутливість до перекосу
 5\Жароміцні сплави\Висока робоча температура;isCause;5\Чутливість до
 концентрації напружень
 5\Коефіцієнт лінійного
 розширення\Гайки\Великий;isCause;3\Заїдання\Зменшення
 5\Корозійностійкі матеріали;isCause;3\Корозійна\Низька
 5\Корозійностійкі матеріали\Сталі;isCause;3\Заїдання
 5\Корозійностійкі матеріали\Сталі;SubClassOf;5\Корозійностійкі матеріали
 5\Кремнієві сталі;isCause;3\Заїдання\Зменшення
 5\Крихкий поверхневий шар;isCause;3\Крихкий злам\Уповільнене крихке
 руйнування
 5\Крихкість;isCause;3\Крихкий злам
 5\Матеріали з метастабільною структурою і малою
 пластичністю;isCause;3\Крихкий злам\Уповільнене крихке руйнування
 5\Матеріали з метастабільною структурою і малою
 пластичністю;isCause;5\Пластичність\Низька
 5\Надвисокоміцні сталі;isCause;3\Залишкова деформація\Зріз
 стержня\Зменшення
 5\Надвисокоміцні сталі;isCause;5\Пластичність\Низька
 5\Насичення атомарним воднем;isCause;3\Крихкий злам\Уповільнене крихке
 руйнування
 5\Насичення атомарним воднем;isCause;5\Крихкість
 5\Наявність азоту;isCause;5\Крихкість
 5\Наявність фосфору;isCause;5\Крихкість
 5\Ніобій;isCause;5\Холодноламкість\Низька
 5\Пластичність\Висока\Гайки;isCause;18\Нерівномірне навантаження по виткам
 різьби\Низьке
 5\Пластичність\Низька;isCause;3\Крихкий злам\Уповільнене крихке руйнування
 5\Пластичність\Низька;isCause;5\Чутливість до концентрації напружень
 5\Пластичність\Низька;Not;5\Пластичність\Висока
 5\Пластмаси;isCause;22
 5\Пластмаси;isCause;3\Розгерметизація\Запобігання
 5\Пластмаси;isCause;5\Корозійностійкі матеріали
 5\Пластмаси;isCause;5\Електричний опір\Великий
 5\Повзучість;isCause;3\Залишкова деформація
 5\Повзучість;isCause;3\Самовідгвинчування
 5\Протекторний захист;isCause;3\Корозійна\Низька
 5\Співвідношення матеріалів\Неоптимальне;isCause;3\Заїдання

5\Співвідношення матеріалів\Пластична гайка;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке
 5\Співвідношення матеріалів\Різні механічні властивості болта і гайки;isCause;3\Залишкова деформація\Зріз витків
 5\Старіння сталі;isCause;5\Пластичність\Низька
 5\Старіння сталі\Низьковуглецевої;isCause;8
 5\Структура матеріалу\Покращення;isCause;8\За циклічного навантаження\Висока
 5\Термообробка\В захисній атмосфері;isCause;5\Вуглець у поверхневих шарах\Зменшення
 5\Термообробка\Відпуск;SubClassOf;5\Термообробка
 5\Термообробка\Відпуск\Збільшення температури відпуску;isCause;4\Допуск форми\Нечутливість до перекосу
 5\Термообробка\Відпуск\Недостатній після гартування;isCause;5\Крихкість
 5\Термообробка\Гартування;SubClassOf;5\Термообробка
 5\Термообробка\Гартування\Перегрів при гартуванні;isCause;5\Крихкість
 5\Термообробка\Хіміко-термічна обробка;isCause;8\За циклічного навантаження\Висока
 5\Термообробка\Хіміко-термічна обробка;SubClassOf;5\Термообробка
 5\Термообробка\Хіміко-термічна обробка\Азотування\Високий момент згвинчування;isCause;3\Крихкий злам
 5\Технологічні дефекти матеріалів;isCause;18
 5\Технологічні дефекти матеріалів;isCause;5\Крихкість
 5\Титанові сплави;isCause;11\На різьбі\Велике
 5\Титанові сплави;isCause;11\На упорному бурті\Велике
 5\Титанові сплави;isCause;15\Робоча температура\Максимальна\350-550
 5\Титанові сплави;isCause;23\Мала
 5\Титанові сплави;isCause;24\Розтягу\Чутливість
 5\Титанові сплави;isCause;3\Залишкова деформація\Зріз витків\Зменшення
 5\Титанові сплави;isCause;3\Крихкий злам\Уповільнене крихке руйнування\Низьке
 5\Титанові сплави;isCause;3\Крихкий злам\Уповільнене крихке руйнування
 5\Титанові сплави;isCause;4\Допуск форми\Нечутливість до перекосу
 5\Титанові сплави;isCause;5\Корозійності матеріали
 5\Титанові сплави;isCause;5\Пластичність\Низька
 5\Титанові сплави;isCause;5\Чутливість до концентрації напружень
 5\Титанові сплави;isCause;8
 5\Титанові сплави;isCause;8\За зрізуючого навантаження
 5\Холодноламкість;isCause;8\За статичного навантаження\Низька
 5\Холодноламкість;Not;5\Холодноламкість\Низька
 5\Чутливість до концентрації напружень;isCause;18
 5\Чутливість до концентрації напружень;Not;5\Чутливість до концентрації напружень\Низька
 6\Згвинчування в нагрітому стані;isCause;4\Посадка\З натягом
 6\Контргайка затянута великим моментом;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке
 6\Контроль затягування;isCause;6\Момент згвинчування\Оптимальний
 6\Контроль затягування\Контроль за видовженням болта;SubClassOf;6\Контроль затягування
 6\Контроль затягування\Контроль за кутом повороту;SubClassOf;6\Контроль затягування

6\Контроль затягування\Контроль за моментом згвинчування;SubClassOf;6\Контроль затягування
 6\Момент згвинчування\Неоптимальний;isCause;3\Втомна тріщина
 6\Момент згвинчування\Неоптимальний\Великий;isCause;22
 6\Момент згвинчування\Неоптимальний\Великий;isCause;3\Залишкова деформація
 6\Момент згвинчування\Неоптимальний\Великий;isCause;8\За циклічного навантаження\Висока
 6\Момент згвинчування\Неоптимальний\Великий;Not;6\Момент згвинчування\Неоптимальний\Малий
 6\Момент згвинчування\Неоптимальний\Малий;isCause;3\Самовідгвинчування
 6\Момент згвинчування\Оптимальний;isCause;8\За циклічного навантаження\Висока
 6\Момент згвинчування\Оптимальний;Not;6\Момент згвинчування\Неоптимальний
 6\Очищення різьби;isCause;11\На різьбі\Мале
 6\Очищення різьби;isCause;21\Добра
 6\Попереднє пластичне деформування перших витків;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке
 6\Селекційне складання;isCause;4\Посадка\З зазором\Зменшення діаметральних зазорів
 6\Тертя при згвинчуванні;isCause;6\Момент згвинчування\Неоптимальний\Великий
 6\Уникнення перекосів;isCause;4\Допуск форми\Перекос осей\Малий
 6\Фіксація болта при згвинчуванні;isCause;14\Кручення\Зменшення
 6\Часте згвинчування розгвинчування;isCause;3\Знос
 7\Болт\Болт під розвертку;isCause;14\Зрізу\Зменшення
 7\Гайка\Гайка розтягу-стиску;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке
 7\Гвинтова вставка;isCause;12\Зменшення механічного спрацювання витків
 7\Гвинтова вставка;isCause;18\Нерівномірне навантаження по виткам різьби\Низьке
 7\Захисний ковпачок;isCause;19\Зменшення
 7\З'єднані деталі\Мала кількість;isCause;3\Самовідгвинчування\Зменшення
 7\Компенсатори температурних деформацій;isCause;17\Температурне\Зменшення
 7\Протектори;isCause;3\Знос\Зменшення
 7\Сферична шайба;isCause;14\Згину\Зменшення
 7\Шайба\З низьковуглецевої сталі;isCause;4\Допуск форми\Перекос опорних поверхонь\Зменшення
 7\Шайба\Пружинна;isCause;3\Самовідгвинчування\Зменшення
 7\Шайба\Сферична;isCause;14\Згину\Зменшення
 8\За зрізуючого навантаження;isCause;3\Залишкова деформація\Зріз стержня\Зменшення
 8\За зрізуючого навантаження;SubClassOf;8
 8\За осьового навантаження;SubClassOf;8
 8\За статичного навантаження;SubClassOf;8
 8\За статичного навантаження\Висока;Not;8\За статичного навантаження\Низька
 8\За статичного навантаження\Низька;isCause;3\Залишкова деформація
 8\За статичного навантаження\Низька;isCause;3\Крихкий злам
 8\За циклічного навантаження;SubClassOf;8
 8\За циклічного навантаження\Висока;Not;8\За циклічного навантаження\Низька
 8\За циклічного навантаження\Низька;isCause;3\Втомна тріщина
 8\Низька;Not;8

9\Болта;isCause;14\Розтягу\Болта
 9\Болта;Not;9\Болта\Мала
 9\З'єднуваних деталей;isCause;14\Розтягу\Болта\Зменшення
 9\З'єднуваних деталей;Not;9\З'єднуваних деталей\Мала
 9\З'єднуваних деталей\Мала;isCause;3\Крихкий злам\Уповільнене крихке
 руйнування
 9\З'єднуваних деталей\Мала;isCause;3\Самовідгвинчування
 9\З'єднуваних деталей\Мала;isCause;8\За циклічного навантаження\Низька
 10\Метод виготовлення\Накатування;isCause;24\Стиску
 10\Метод виготовлення\Накатування;isCause;3\Крихкий злам\Уповільнене крихке
 руйнування\Низьке
 10\Метод виготовлення\Накатування\Стійкість інструмента\Низька;Not;10\Метод
 виготовлення\Накатування\Стійкість інструмента
 10\Метод виготовлення\Обкатування після нарізання;isCause;24\Стиску
 10\Метод виготовлення\Піскоструминна обробка;isCause;3\Крихкий
 злам\Уповільнене крихке руйнування\Низьке
 10\Метод виготовлення\Полірування;isCause;3\Крихкий злам\Уповільнене крихке
 руйнування\Низьке
 10\Попередній статичний розтяг високоміцних болтів;isCause;8\За циклічного
 навантаження\Висока
 10\Режими обробки\Оптимальні;isCause;24\Стиску
 10\Режими обробки\Оптимальні;isCause;4\Допуск розміру\Малий
 10\Режими обробки\Оптимальні;isCause;8
 10\Технологічні дефекти;isCause;18
 10\Технологічні дефекти;isCause;3\Крихкий злам\Уповільнене крихке
 руйнування
 10\Технологічні дефекти;isCause;8\За статичного навантаження\Низька
 11\Зміна коефіцієнта тертя під час повторних
 згвинчуваннях\Збільшується;Not;11\Зміна коефіцієнта тертя під час повторних
 згвинчуваннях\Зменшується
 11\На різьбі\Велике;isCause;21\Погана
 11\На різьбі\Велике;isCause;3\Заїдання
 11\На різьбі\Велике;Not;11\На різьбі\Мале
 11\На різьбі\Мале;isCause;21\Добра
 11\На упорному бурті\Велике;isCause;21\Погана
 11\На упорному бурті\Велике;isCause;3\Заїдання
 11\На упорному бурті\Велике;Not;11\На упорному бурті\Мале
 11\На упорному бурті\Мале;isCause;21\Добра
 12\Вибір оптимальних допусків і шорсткості;isCause;18\Низька
 12\Вибір оптимальної технології;isCause;24\Стиску
 12\Вибір оптимальної технології;isCause;4\Шорсткість\Низька
 12\Запобігання заїдань;isCause;3\Заїдання\Зменшення
 12\Запобігання розгерметизації;isCause;22
 12\Захист від корозійно-втомного і статичного руйнування;isCause;8
 12\Захист від корозійного руйнування;isCause;3\Корозійна\Низька
 12\Захист від механічного спрацювання деталей;isCause;3\Знос\Зменшення
 12\Захист від середовища;isCause;13\Корозійне\Захист
 12\Захист при транспортуванні;isCause;19\Зменшення
 12\Зменшення концентрації напружень;isCause;18\Низька
 12\Підбір матеріалів і термообробки;isCause;3\Корозійна\Низька
 12\Підбір матеріалів і термообробки;isCause;8

12\Попередження самовідгвинчування;isCause;3\Самовідгвинчування\Зменшення
 12\Центрування різьби запобігання згину;isCause;14\Згину\Зменшення
 13\Інгібіторний захист;isCause;3\Корозійна\Низька
 13\Корозійне;isCause;3\Корозійна
 13\Корозійне;isCause;3\Корозійне розтріскування
 13\Корозійне;isCause;8\За циклічного навантаження\Низька
 13\Корозійне;Not;13\Корозійне\Захист
 14\Згину;isCause;3\Втомна тріщина
 14\Згину\Зменшення;Not;14\Згину
 14\Зрізу;isCause;3\Залишкова деформація\Зріз стержня
 14\Зрізу;Not;14\Зрізу\Зменшення
 14\Кручення;Not;14\Кручення\Зменшення
 14\Розтягу\Болта;isCause;3\Крихкий злам\Уповільнене крихке руйнування
 14\Розтягу\Болта;Not;14\Розтягу\Болта\Зменшення
 15\Робоча температура\Висока;isCause;17\Температурне
 15\Робоча температура\Висока;isCause;3\Заїдання
 15\Робоча температура\Висока;isCause;5\Повзучість
 15\Робоча температура\Висока;isCause;5\Чутливість до концентрації напружень
 15\Робоча температура\Висока;isCause;8\За циклічного навантаження\Низька
 15\Робоча температура\Висока;Not;15\Робоча температура\Низька
 15\Робоча температура\Низька;isCause;5\Пластичність\Низька
 15\Робоча температура\Низька;isCause;5\Холодноламкість
 15\Температура за залишкових напруженнях стиску\Висока;isCause;8\За
 циклічного навантаження\Низька
 16\Великі розміри;isCause;8\За циклічного навантаження\Низька
 16\Великі розміри;Not;16\Малі розміри
 17\Вібрація;isCause;3\Самовідгвинчування
 17\Згинаюче;isCause;14\Згину
 17\Згинаюче;isCause;22\Низька
 17\Згинаюче;isCause;3\Втомна тріщина
 17\Зрізаюче;isCause;14\Зрізу
 17\Зрізаюче;isCause;3\Залишкова деформація\Зріз стержня
 17\Крутне;isCause;14\Кручення
 17\Крутне\Догвинчування;isCause;3\Залишкова деформація\Зріз витків
 17\Крутне\Догвинчування;isCause;3\Самовідгвинчування\Зменшення
 17\Крутне\Догвинчування;SubClassOf;17\Крутне
 17\Крутне\Розгвинчування;isCause;3\Самовідгвинчування
 17\Крутне\Розгвинчування;SubClassOf;17\Крутне
 17\Осьове\Розтягу;isCause;14\Розтягу
 17\Осьове\Розтягу;isCause;3\Залишкова деформація\Зріз витків
 17\Осьове\Розтягу;isCause;3\Залишкова деформація\Обрив стержня
 17\Осьове\Розтягу;Not;17\Осьове\Стиску
 17\Температурне;isCause;17\Осьове\Розтягу
 17\Температурне;Not;17\Температурне\Зменшення
 17\Тиск;isCause;13\Корозійне
 17\Удар;isCause;3\Від високошвидкісного навантаження
 17\Циклічне;isCause;3\Втомна тріщина
 17\Швидкісне\Низьковуглецева сталь;isCause;8
 18;isCause;3\Крихкий злам\Уповільнене крихке руйнування
 18;isCause;8\За статичного навантаження\Низька
 18;isCause;8\За циклічного навантаження\Низька

```

18;Not;18\Низька
18\Нерівномірне навантаження по виткам різьби;isCause;18
18\Нерівномірне навантаження по виткам різьби;isCause;3\Втомна тріщина\В
першому витку ніпеля
18\Нерівномірне навантаження по виткам різьби;Not;18\Нерівномірне
навантаження по виткам різьби\Низьке
18\Під головкою болта;isCause;3\Втомна тріщина\Під головкою болта
18\Під головкою болта;Not;18\Під головкою болта\Низька
18\Під головкою болта;SubClassOf;18
20;isCause;3\Самовідгвинчування\Зменшення
20\Жорстке;SubClassOf;20
20\Клеєм;SubClassOf;20
20\Фрикційне;SubClassOf;20
21\Погана;isCause;14\Кручення
21\Погана;isCause;3\Знос
21\Погана;Not;21\Добра
22;isCause;13\Корозійне\Захист
22;Not;22\Низька
22\Низька;isCause;3\Розгерметизація
23\Мала;Not;23\Велика
24\Розтягу;isCause;8\За циклічного навантаження\Низька
24\Розтягу;Not;24\Стиску
24\Розтягу\Чутливість;isCause;8\За циклічного навантаження\Низька
24\Стиску;isCause;8\За циклічного навантаження\Висока
24\Стиску\За високої температури;And;15\Робоча температура\Висока
24\Стиску\За високої температури;And;24\Стиску
24\Стиску\За високої температури;isCause;8\За циклічного
навантаження\Низька
25;isCause;3\Самовідгвинчування
25;Not;25\Низька
26;isCause;11\На різьбі\Мале
26;isCause;11\На упорному бурті\Мале
26;isCause;12\Зменшення механічного спрацювання витків
26;isCause;3\Заїдання\Зменшення
26\Відсутність;Not;26
27;isCause;18
27;Not;19\Зменшення
28;isCause;22\Низька
28;isCause;8\За циклічного навантаження\Низька

```

Лістинг X.2 – generatedKBcode.py

```

# -*- coding: CP1251 -*-
import os
import copy

class Property(object):
    '''Клас, який описує властивість об'єкта'''

```



```

def __init__(self, subj, name, inverseName='', functional=False,
symmetric=False, transitive=False):
    '''Конструктор'''
    self.subj=subj # суб'єкт властивості
    self.name=name # назва властивості
    self.inverseName=inverseName # назва інверсної властивості
    self.functional=functional # властивість функціональна
    self.symmetric=symmetric # властивість симетрична
    self.transitive=transitive # властивість транзитивна
    self.set=set() # множина значень властивості
    self.subj.__setattr__(self.name,self) # установити атрибут
властивості для суб'єкта

def add(self,*args):
    '''Додає об'єкт або кортеж об'єктів в множину'''
    for obj in args: # для всіх об'єктів в args
        if self.functional: # якщо властивість функціональна
            self.set.clear() # очистити множину
            self.set.add(obj) # додати об'єкт в множину

def get(self, showTransitive=False):
    '''Повертає множину значень властивості
(showTransitive=True - транзитивної властивості)'''
    def getTransitive(subj,s=set()):
        '''Повертає множину значень транзитивної властивості.
Рекурсивна'''
        if hasattr(subj, self.name): # якщо subj має атрибут
self.name
            # для усіх об'єктів в властивості з назвою self.name
            for obj in subj.__dict__[self.name].set:
                if obj not in s: # якщо obj немає в множині s
                    s.add(obj) # додати об'єкт в множину
                    s=getTransitive(obj,s) # рекурсія
            return s # повертає множину
        # якщо властивість транзитивна і показувати транзитивні
        if self.transitive and showTransitive:
            # множина значень транзитивної властивості
            s=getTransitive(self.subj)
            s.discard(self.subj) # вилучити self.subj з множини, якщо є
            return s
        # інакше повернути множину значень властивості
        else: return self.set

KB={} # словник бази знань
programDir="D:\!My_doc\Python_projects\TreePyKB"

class X(object):
    '''Базовий клас онтології'''
    def __init__(self): # конструктор
        # ці властивості дозволяють створювати нові об'єкти за допомогою
логічних операцій

```

```

    Property(subj=self,name='And') # властивість 'And'
    Property(subj=self,name='Or') # властивість 'Or'
    Property(subj=self,name='Not',symmetric=True) # властивість 'Not'
    Property(subj=self,name='SubClassOf',transitive=True) # властивість
'SubClassOf'
    self.doc="" # довідка
KB[r""""Base"""]=X; del X

```

```

class X(KB[r""""Base"""]):
    '''Клас, який описує посилання на джерело'''
    def __init__(self): # конструктор
        KB[r""""Base"""].__init__(self)
        # властивість 'є посиланням'
        Property(subj=self,name='isReference',inverseName='hasReference')
KB[r""""Base\Джерело"""]=X; del X

```

```

class X(KB[r""""Base"""]):
    '''Клас, який описує залежність'''
    def __init__(self, xy=None, relative=None, xName='x', yName='y'):
        KB[r""""Base"""].__init__(self)
        # властивість 'є залежністю'
        Property(subj=self,name='isDependence',inverseName='hasDependence')
        self.xy=xy # дані у вигляді [(x1,y1),(x2,y2),(x3,y3)]
        self.relative=relative # відносна залежність (збільшує, зменшує, є
екстремум)
        self.xName=xName; self.yName=yName # назви осей
    def plot(self):
        '''Рисує графік залежності'''
        from matplotlib import rcParams, pyplot # бібліотека matplotlib
        rcParams['text.usetex']=False
        rcParams['font.sans-serif'] = ['Arial']
        rcParams['font.serif'] = ['Arial'] # шрифт для вводу кирилиці
        x,y=[p[0] for p in self.xy],[p[1] for p in self.xy] # розділити
дані
        pyplot.plot(x,y,'b-') #крива
        pyplot.title(unicode('')) # заголовок
        pyplot.xlabel(unicode(self.xName)) # надпис осі x
        pyplot.ylabel(unicode(self.yName)) # надпис осі y
        pyplot.grid(True) # сітка
        pyplot.show() # показати рисунок
    def interp(self,x,reverse=False):
        '''Знаходить значення Y лінійною інтерполяцією
        якщо reverse=True, то знаходить значення X'''
        # якщо reverse=True, поміняти X і Y місцями
        if reverse: data=[(p[1],p[0]) for p in self.xy]
        else: data=self.xy
        lines=[] # список ліній залежності
        p1=data[0] # перша точка
        for p2 in data[1:]: # для усіх точок крім першої
            lines.append((p1,p2)) # створити лінію з пар сусідніх точок
            p1=p2

```

```

results=[] # список результатів
for line in lines: # для всіх ліній залежності
    # координати точок лінії
    x1,y1,x2,y2=line[0][0],line[0][1],line[1][0],line[1][1]
    if x>=x1 and x<=x2 or x<=x1 and x>=x2: # якщо x в межах [x1,x2]
        results.append((x-x1)*(y2-y1)/(x2-x1)+y1) # знайти точку
перетину x з лінією
    return results
KB[r""Base\Залежність""]=X; del X

class X(KB[r""Base""]):
    def __init__(self):
        KB[r""Base""].__init__(self)
        # властивість 'параметр'
        Property(subj=self,name='parameter')
KB[r""Base\Модель""]=X; del X

class X(KB[r""Base\Модель""]):
    def __init__(self):
        KB[r""Base\Модель""].__init__(self)
    def create(self):
        "цю функцію необхідно викликати після створення об'єктів KB"

self.d={ # словник геометричних параметрів
'd_n':"зовнішній діаметр різьби ніпеля",
'd2_n':"середній діаметр різьби ніпеля",
'd1_n':"внутрішній діаметр різьби ніпеля",
'r_n':"радіус западин різьби ніпеля",
'dn':"діаметр бурта ніпеля",
'd1n':"діаметр зарізьбової канавки ніпеля",
'l1n':"довжина ніпеля",
'l2n':"довжина зарізьбової канавки ніпеля",
'l3n':"довжина ніпеля без фаски на різьбі",
'l4n':"довжина ніпеля з буртом",
'r3n':"радіус скруглень зарізьбової канавки ніпеля",
'd_m':"зовнішній діаметр різьби муфти",
'd2_m':"середній діаметр різьби муфти",
'd1_m':"внутрішній діаметр різьби муфти",
'dm':"зовнішній діаметр муфти",
'd1m':"внутрішній діаметр опорної поверхні муфти",
'lm':"довжина муфти",
'd0':"діаметр тіла штанги",
'p_n':"крок різьби ніпеля",
'p_m':"крок різьби муфти"
}

self.p={ # словник інших параметрів
'delta_ln':"величина збільшення довжини ніпеля",
'material1':"назва матеріалу з бібліотеки матеріалів",
'material2':"назва матеріалу з бібліотеки матеріалів",
'bolt_load':"осьова деформація муфти під час згвинчування (мм)",

```

```

'sigma1':"напруження в тілі штанги для кроку 1 (Па)",
'sigma2':"напруження в тілі штанги для кроку 2 (Па)"
}

```

```

def createAbaqusModel(self):
    """Створює модель Abaqus. Створює тимчасовий файл з даними для
    передачі їх скрипту Abaqus. Виконує скрипт в Abaqus"""
    import os,pickle,tempfile,subprocess
    path=os.path.join(os.getcwd(), self.name) # шлях до каталогу
    data={'d':self.d,'p':self.p,'path':path}
    name=os.path.join(tempfile.gettempdir(),"data4AbaqusScript.tmp")
    f=open(name, "wb") # відкрити бінарний файл для запису
    pickle.dump(data,f) # законсервувати дані у файлі
    f.close() # закрити файл

    print "Abaqus CAE started. Please wait"
    # виконує скрипт в Abaqus та чекає завершення
    subprocess.Popen(r'C:\SIMULIA\Abaqus\6.11-3\exec\abq6113.exe cae
noGUI=gost13877_96AbaqusMain.py').communicate()
    #os.system(r'start /WAIT abaqus cae noGUI=gost13877_96Abaqus.py')
    print "Abaqus CAE finished"
KB[r""Base\Модель\ГОСТ 13877-96""]=X; del X

```

```

class X(KB[r""Base\Модель\ГОСТ 13877-96""]):
    def __init__(self):
        KB[r""Base\Модель\ГОСТ 13877-96""].__init__(self)
    def create(self):
        "цю функцію необхідно викликати після створення об'єктів KB"
        KB[r""Base\Модель\ГОСТ 13877-96""].create(self)

d={
'd_n':(27, -0.48, -0.376),
'd2_n':(25.35, -0.204, -0.047),
'd1_n':(24.25, 0, -0.415),
'r_n':(0.28, 0, 0.08),
'dn':(38.1, -0.25, 0.13),
'd1n':(23.24, -0.13, 0.13),
'l1n':(36.5, 0, 1.6),
'l2n':(15, 0.2, 1),
'l3n':(32, 0, 1.5),
'l4n':(48, -1, 1.5),
'r3n':(3, 0, 0.8),
'd_m':(27, 0, 0.27),
'd2_m':(25.35, 0, 0.202),
'd1_m':(24.25, 0, 0.54),
'dm':(41.3, -0.25, 0.13),
'd1m':(27.43, 0, 0.25),
'lm':(102, -1, 1),
'd0':(19.1, -0.41,0.2),
'p_n':(2.54,0,0),
'p_m':(2.54,0,0)
}

```

```

    }

    for k in d:
        self.d[k]=[self.d[k]]+list(d[k])

    p={
        'delta_ln':0,
        'material1':"40fesafe",
        'material2':"40fesafe",
        'bolt_load':-0.1,
        'sigma1':1,
        'sigma2':170.0e+6
    }

    for k in p:
        self.p[k]=[self.p[k]]+[p[k]]
KB[r""Base\Модель\ГОСТ 13877-96\ШН 19""]=X; del X

class X(KB[r""Base""]):
    "Клас описує поняття розміру"
    def __init__(self):
        KB[r""Base""].__init__(self)
    def create(self,doc,n,ei,es,v):
        "створює об'єкт"
        self.doc=doc # довідка
        self.n=n # номінальний розмір
        self.ei=ei # нижнє відхилення
        self.es=es # верхнє відхилення
        self.v=v # дійсне значення
    def min(self):
        "повертає мінімальний розмір"
        return self.n+self.ei
    def max(self):
        "повертає максимальний розмір"
        return self.n+self.es
KB[r""Base\Параметр""]=X; del X

class X(KB[r""Base""]):
    '''Клас, який описує факт (триплет) у вигляді
    суб'єкт-предикат-об'єкт'''
    def __init__(self): # конструктор
        KB[r""Base""].__init__(self)
        # властивість 'має посилання'
        Property(subj=self,name='hasReference',inverseName='isReference')
        # властивість 'має залежність'
        Property(subj=self,name='hasDependence',inverseName='isDependence')
    def create(self,subjName,propName,objName):
        self.subjName=subjName # назва суб'єкта
        self.propName=propName # назва предиката (властивість)
        self.objName=objName # назва об'єкта
        # додати значення в властивість, якщо немає

```

```

        KB[self.subjName].__dict__[self.propName].add(KB[self.objName])
KB[r""Base\Факт""]=X; del X

class X(KB[r""Base""]): # успадковує клас Base
    '''Клас, який описує фактор'''
    def __init__(self): # конструктор
        KB[r""Base""].__init__(self)
        # транзитивна властивість 'є причиною'

Property(subj=self,name='isCause',inverseName='isEffect',transitive=True)
    # транзитивна властивість 'є наслідком'

Property(subj=self,name='isEffect',inverseName='isCause',transitive=True)
KB[r""Base\Фактор""]=X; del X

#####
KB[r""Base\Джерело\И.А.Биргер, Г.Б.Иосилевич. Резьбовые и фланцевые
соединения""]=KB[r""Base\Джерело""]()

KB[r""Base\Модель\ГОСТ 13877-96\ШН
19\bolt_load\0.1""]=KB[r""Base\Модель\ГОСТ 13877-96\ШН 19""]()

KB[r""Base\Модель\ГОСТ 13877-96\ШН
19\bolt_load\0.1\delta_ln\10""]=KB[r""Base\Модель\ГОСТ 13877-96\ШН
19""]()

KB[r""Base\Фактор\Концентрація напружень""]=KB[r""Base\Фактор""]()

KB[r""Base\Фактор\Міцність""]=KB[r""Base\Фактор""]()

KB[r""Base\Фактор\Міцність\За циклічного
навантаження""]=KB[r""Base\Фактор""]()

KB[r""Base\Фактор\Міцність\За циклічного
навантаження\Низька""]=KB[r""Base\Фактор""]()

KB[r""Base\Фактор\Технологія""]=KB[r""Base\Фактор""]()

KB[r""Base\Фактор\Технологія\Технологічні
дефекти""]=KB[r""Base\Фактор""]()

KB[r""Base\Фактор\Концентрація
напружень""].__dict__["isCause"].add(KB[r""Base\Фактор\Міцність\За
циклічного навантаження\Низька""])

KB[r""Base\Фактор\Технологія\Технологічні
дефекти""].__dict__["isCause"].add(KB[r""Base\Фактор\Концентрація
напружень""])
#####

# правила виведення

```

```

def allFactsSet(showTransitive=True):
    '''Повертає множину всіх фактів (аксіом) бази знань.
    Факти у вигляді кортежу (суб'єкт, предикат, об'єкт)'''
    factsSet=set() # множина фактів
    for k in KB.values(): # для усіх концептів
        if k.__class__.__name__==r"""\Base\Фактор""": # якщо це фактор
            for p in k.__dict__: # для усіх атрибутів
                if k.__dict__[p].__class__.__name__=='Property': # якщо
атрибут властивість
                    for obj in k.__dict__[p].get(showTransitive): # для
усіх значень властивості
                        factsSet.add((k,p,obj)) # додати факт в множину
фактів
    return factsSet

for k,v in KB.iteritems():
    v.name=k # для всіх об'єктів в KB задає атрибут name і присвоює йому
значення ключа в KB
    v.codes={} # властивість codes - словник вихідних кодів
    if v.__class__.__name__=='type': # якщо це клас
        v.__name__=k # для всіх класів в KB присвоює __name__ значення
ключа в KB

assertedFacts=allFactsSet(False) # введені факти
print len(assertedFacts)
import csv
writer = csv.writer(open("assertedFacts.csv", "wb"),delimiter = ';') #
відкрити csv файл
for x in assertedFacts:
    writer.writerow([x[0].name,x[1],x[2].name]) # записати в файл csv

while True: # цикл для застосування правил
    beforeFacts=allFactsSet() # кількість фактів до
    print str(len(beforeFacts))+ ' facts. Iteration for apply rules...'

    # логічне виведення для інверсних і симетричних властивостей
    for k in KB.values(): # для усіх концептів
        for p,pv in k.__dict__.iteritems(): # для усіх атрибутів
            if pv.__class__.__name__=='Property': # якщо атрибут
властивість
                for obj in pv.get(True): # для усіх значень властивості

                    if pv.inverseName!='': # якщо є інверсна властивість
                        # додати його в інверсну властивість об'єкта
                        obj.__dict__[pv.inverseName].add(pv.subj)

                    if pv.symmetric: # якщо властивість симетрична
                        # додати його в аналогічну властивість об'єкта
                        obj.__dict__[pv.name].add(pv.subj)

    # правила виведення:

```

```

# SubClassOf(?x, ?y) & isCause(?y, ?p) -> isCause(?x, ?p)
# SubClassOf(?x, ?y) & isEffect(?y, ?p) -> isEffect(?x, ?p)
for k in KB.values(): # для усіх концептів
    if k.__class__.__name__==r""Base\Фактор"": # якщо це фактор
        for bk in k.SubClassOf.get(True): # для усіх базових концептів
            (класів)
                for x in bk.isCause.get(True):
                    k.isCause.add(x)
                for x in bk.isEffect.get(True):
                    k.isEffect.add(x)

# правила виведення:
# Not(?x, ?nx) & isCause(?nx, ?ny) & Not(?y, ?ny) -> isCause(?x, ?y)
# Not(?x, ?nx) & isEffect(?nx, ?ny) & Not(?y, ?ny) -> isEffect(?x, ?y)
for k in KB.values(): # для усіх концептів
    if k.__class__.__name__==r""Base\Фактор"": # якщо це фактор
        for nk in k.Not.get(): # для усіх заперечень концепту
            for e in nk.isCause.get(True): # для усіх наслідків
                заперечення
                    for ne in e.Not.get(): # для усіх заперечень наслідків
                        заперечення
                            k.isCause.add(ne) # концепт є їх причиною
            for e in nk.isEffect.get(True): # для усіх причин
                заперечення
                    for ne in e.Not.get(): # для усіх заперечень причин
                        заперечення
                            k.isEffect.add(ne) # концепт є їх наслідком

    afterFacts=allFactsSet() # кількість фактів після
    # перервати цикл, якщо кількість фактів до і після застосування правил
    рівна
    if beforeFacts==afterFacts: break

allFacts=allFactsSet() # усі факти
print len(allFacts)
import csv
writer = csv.writer(open("allFacts.csv", "wb"),delimiter = ';') # відкрити
csv файл
for x in allFacts:
    writer.writerow([x[0].name,x[1],x[2].name]) # записати в файл csv

# блок запитів до бази знань
print "Відповідь на запит:"
for x in KB[r""Base\Фактор\Міцність\За циклічного
навантаження\Низька""].isEffect.get(True):
    print x.name

```


ДОДАТОК Ц

Приклад PLM системи різьбових з'єднань

Код доступний також в GitHub (vkorey/ThreadsPLM. URL: <http://github.com/vkorey/ThreadsPLM>). Вміст пакету:

- main.py – головний модуль,
- tools.py – утиліти,

класи:

- class_Agent.py,
- class_Property.py,
- class_Datalog.py,
- class_Dependence.py,
- class_DependenceMulti.py,
- class_Fact.py,
- class_Factor.py,
- class_Model.py,
- class_Parameter.py,

агенти:

- Datalog_inference.py,
- Dependence_dependence1.py,
- Dependence_dependence2.py,
- Dependence_dependence3.py,
- DependenceMulti_dependence4.py,
- Fact_конц_напр_спричинює зменшення цикл_міцності.py,
- Factor_відсутність корозійного пошкодження.py,
- Factor_збільшення довжини зарізьбової канавки.py,
- Factor_збільшення радіуса скруглень зарізьбової канавки.py,
- Factor_збільшення циклічної міцності.py,
- Factor_зменшення циклічної міцності.py,

- Factor_концентрація напружень.py,
- Factor_корозійна язва.py,
- Factor_корозійне пошкодження.py,
- Model_ШН19ГОСТ13877-96.py,
- Model_ШН19ГОСТ13877-96_r3n=2.5.py,
- Parameter_довжина зарізьбової канавки.py,
- Parameter_коефіцієнт запасу втомної міцності.py,
- Parameter_логарифм циклічної довговічності.py,
- Parameter_радіус скруглень зарізьбової канавки.py,
- Property_factorHigh.py,
- Property_factorLow.py,
- Property_isCause.py,
- Property_isContraryOf.py,
- Property_isEffect.py,
- Property_objecT.py,
- Property_source.py,
- Property_subClassOf.py,
- Property_subject.py,
- Property_Xpar.py,
- Property_Ypar.py.

Лістинг Ц.1 – main.py

```

#-*- coding: utf-8 -*-
"Це приклад роботи з системою"
from tools import *

files=getFiles()
createClasses(files)
createKB(files)
#loadKB()

applyRules(n=10)
#saveKB()

```

ЛІСТИНГ Ц.2 – tools.py

```

#-*- coding: utf-8 -*-
import sys,os,re
import numpy as np
KB={}
systemencoding=sys.getfilesystemencoding() #'mbscs'

def getFiles(ex={'main.py','tools.py'} ):
    "Повертає список файлів поточного каталогу в довільному порядку"
    files=os.listdir(os.getcwd())
    files=[f for f in files if f[-3:]==' .py' and f not in ex]
    #print 'files:',files
    return files

def createClasses(files):
    files=[f for f in files if f.startswith('class_')]
    while files:
        f=files.pop(0)
        try:
            execfile(f.encode(systemencoding), globals()) #KB
        except NameError:
            files.append(f)

def createKB(files):
    "Створює систему агентів з модулів files"
    files=[f for f in files if not f.startswith('class_')]
    for f in files:
        cls,_,name=f.partition('_')
        name=name[:-3].encode('utf-8')
        #if name in KB: continue
        obj=globals()[cls](name)
        obj.__dict__['KB']=KB
        KB[obj.__name__]=obj
        execfile(f.encode(systemencoding), obj.__dict__)

def getClassObjects(clsName):
    return [k for k in KB if KB[k].__class__.__name__==clsName]

def KBToPropFacts(prop):
    "Повертає усі факти-триплети KB з властивістю prop (prop має бути set)"
    facts=set()
    for k in KB:
        if hasattr(KB[k], prop):
            if KB[k].FunctionalProperty:
                facts.add((k, prop, KB[k].__dict__[prop]))
            else:
                for v in KB[k].__dict__[prop]:
                    facts.add((k, prop, v))
    return facts

```

```

def KBToFacts():
    "Повертає усі факти-триплети KB"
    facts=set()
    props=getClassObjects('Property')
    for p in props:
        facts.update(KBToPropFacts(p))
    return facts

def factToKB(subj,pred,obj):
    "Оновлює значення властивості об'єкта за триплетом"
    #потрібна також перевірка на Domain і Range!
    if not(KB.has_key(pred) and KB[pred].__class__==Property): return
#Property in KB[pred].__class__.__mro__
    if KB.has_key(subj) and hasattr(KB[subj],pred):
        if KB[pred].FunctionalProperty:
            KB[subj].__dict__[pred]=obj
        else:
            KB[subj].__dict__[pred].add(obj)

def factsToKB(facts):
    "Оновлює всю KB за списком фактів-триплетів"
    for s,p,o in facts:
        factToKB(s,p,o)

def runDatalog(facts, rules, predicates):
    """виконує логічне виведення в pyDatalog. Повертає список триплетів.
    facts - список Datalog-фактів,
    rules - список Datalog-правил,
    predicates - список предикатів, для яких будуть шукатись факти"""

    #if not predicates:
    #    predicates={p for s,p,o in facts}|{r.split('(')[0] for r in rules}

    from pyDatalog.pyDatalog import assert_fact, load, ask, clear
    code='\n'.join(facts)+'\n'+'\n'.join(rules) # факти і правила
    load(code)
    allFacts=set()
    for pred in predicates:
        # try:
        res=ask('%s(X,Y)%pred).answers
        # except AttributeError: #Predicate without definition
        #     continue
        for subj,obj in res:
            allFacts.add((subj.encode('utf-8'),pred.encode('utf-
8'),obj.encode('utf-8')))
            print subj,pred,obj
    clear()
    return allFacts

def subClasses(n, s=set()):
    "Множина усіх підоб'єктів об'єкта n. Рекурсивна."

```

```

for k in KB:
    if n in KB[k].subClassOf:
        s.add(k)
        s.update(subClasses(k, s))
return s

def setAttrSubClasses(name, attr, value, ch=False):
    "Установлює значення value усім атрибутам attr усіх підкласів об'єкта
    name, якщо атрибута немає"
    for k in subClasses(name, set()):
        if not hasattr(KB[k], attr):
            setattr(KB[k], attr, value)
            ch=True
    return ch

def applyRules(lst=KB, n=5):
    "Застосовує правила до кожного агента з lst n раз, поки є зміни"
    ch=True
    while ch and n>0: # поки правила створюють зміни
        ch=False
        print '*ApplyRules*'
        for k in lst.keys():
            if KB[k].active and KB[k].rule(): ch=True
        n-=1

def saveKB(lst=KB, new=True):
    import shelve
    from dill import Pickler
    shelve.Pickler = Pickler
    KBp=shelve.open("shelve.dat")
    if new: KBp.clear()
    for k in lst:
        KBp[k]=KB[k]
    KBp.close()

def loadKB():
    import shelve
    from dill import Unpickler
    shelve.Unpickler = Unpickler
    KBp=shelve.open("shelve.dat")
    for k in KBp:
        KB[k]=KBp[k]
    KBp.close()

def find(regex=".*", lst=KB):
    """"Повертає список назв, які відповідають регулярному виразу""""
    res=[]
    po=re.compile(regex, re.IGNORECASE | re.UNICODE)
    for k in lst:
        mo=po.search(k.decode('utf-8'))
        if mo:

```

```

        res.append(k)
        print k
    res.sort() # сортувати
    return res

def createFile(cls,name):
    fname=cls+'_'+name+'.py'
    fname=fname.decode('utf-8') # в Юнікод
    if os.path.exists(fname):
        print 'Error! File exists'
        return
    t='' '#-*- coding: utf-8 -*-'
    """"%s_%s""""
    '''%(cls,name)
    with open(fname,'w') as f:
        f.write(t)
    return fname

def autoKey(start):
    """"Генерує унікальний ключ для KB шляхом додавання цілого числа до
start""""
    k=start+'0'
    n=0
    while k in KB.keys():
        n+=1
        k=start+str(n)
    return k

def isSibling(a,b):
    "Словники рівні або відрізняється знач. не більше ніж одного парам."
    if a==b: return True
    if set(a.keys()) ^ set(b.keys()): return False # різниця ключів
    d=set(a.items()) ^ set(b.items()) # різниця
    return len(d)==2

def query1():
    for v in KB["концентрація напружень"].isEffect:
        print v
    #print KB["isCause"].__doc__

def query2():
    import matplotlib.pyplot as plt
    d=KB['dependence2']
    print d.linregress()
    d.plot(plt)

def query3():
    import networkx as nx
    G = nx.DiGraph()
    for i in KB:
        if KB[i].__class__.__name__=='Factor':

```

```

        for j in KB[i].isCause:
            G.add_edge(i.decode('utf-8'), j.decode('utf-8'),
label='isCause')

    pr=nx.pagerank(G) # for DiGraph only
    pr={k:'%s\n%f'%(k,v) for k,v in pr.iteritems()}
    G=nx.relabel_nodes(G, pr)
    nx.drawing.nx_agraph.write_dot(G,'graph.dot')
    os.system(r'"d:\Program Files\Graphviz2.38\bin\dot.exe" -Tsvg graph.dot
-o graph.svg') # конвертуємо в SVG
    #nx.write_graphml_lxml(G, "graph.graphml")
##
def query4():
    def proToEdges(pro):
        if KB[pro].FunctionalProperty:
            G.add_edge(i.decode('utf-8'), KB[i].__dict__[pro].decode('utf-
8'), label=pro)
            return
        for j in KB[i].__dict__[pro]:
            G.add_edge(i.decode('utf-8'), j.decode('utf-8'), label=pro)

import networkx as nx
G = nx.MultiDiGraph()
for i in KB:
    if KB[i].__class__.__name__=='Factor':
        proToEdges('isCause')
        proToEdges('isEffect')
        proToEdges('subClassOf')
        proToEdges('isContraryOf')
    if KB[i].__class__.__name__=='Fact':
        proToEdges('object')
        proToEdges('subject')
    if KB[i].__class__.__name__=='Parameter':
        proToEdges('factorHigh')
        proToEdges('factorLow')
    if KB[i].__class__.__name__=='Dependence':
        proToEdges('Xpar')
        proToEdges('Ypar')
        proToEdges('source')
nx.drawing.nx_agraph.write_dot(G,'graph.dot')
replaceLongStr('graph.dot')
os.system(r'"d:\Program Files\Graphviz2.38\bin\dot.exe" -Tsvg graph.dot
-o graph.svg') # конвертуємо в SVG
##
def replaceLongStr(file='graph.dot'):
    def repl(mo):
        s=mo.group(1) # рядок знайденої групи
        n = 20 # кількість символів у рядку
        s=r'\n'.join([s[i:i+n] for i in range(0, len(s), n)])
        return s
    import re

```

```

all=open(file,'r').read()
po=re.compile(r'(".*?")', re.UNICODE|re.DOTALL)
all=re.sub(po, repl, all.decode('utf-8'))
f=open(file,'w')
f.write(all.encode('utf-8'))
f.close()

```

Лістинг Ц.3 – class_Agent.py

```

#-*- coding: utf-8 -*-
class Agent(object):
    "Інтелектуальний агент"
    def __init__(self, name):
        self.__name__=name
        self.active=True

    def rule(self):
        "Правило поведінки агента. Повертає 1, якщо його застосування
        призвело до змін"
        return False #у іншому випадку повертає False або нічого не
        повертає

```

Лістинг Ц.4 – class_Property.py

```

#-*- coding: utf-8 -*-
class Property(Agent):
    """Property. Базовий клас властивостей"""
    def __init__(self, name):
        super(Property,self).__init__(name)
        self.inverseOf=None
        self.domain=set()
        self.range=set()
        self.SymmetricProperty=False
        self.TransitiveProperty=False
        self.FunctionalProperty=False

    def datalogRules(self):
        r=set()
        if self.inverseOf: r.add('%s(X,Y) <= %s(Y,X)'%(self.inverseOf,
self.__name__))
        if self.SymmetricProperty: r.add('%s(X,Y) <=
%s(Y,X)'%(self.__name__, self.__name__))
        return r

```

Лістинг Ц.5 – class_Datalog.py

```

#-*- coding: utf-8 -*-
class Datalog(Agent):
    """Datalog."""
    def __init__(self, name):
        super(Datalog,self).__init__(name)

```



```

self.oldfacts=set()

def getDatalogFacts(self,facts):
    "Повертає список datalog-фактів зі списку фактів-триплетів"
    dfacts=[]
    for k, p, v in facts:
        df='+%s(u"%s",u"%s")'%(p,k.decode('utf-8'),v.decode('utf-8'))
        dfacts.append(df)
    return dfacts

def getDatalogRules(self):
    r=set()
    for k in KB:
        if hasattr(KB[k],'datalogRules'):
            r.update(KB[k].datalogRules())
    return r

def rule(self):
    props=getClassObjects('Property')
    facts=KBToFacts()
    dfacts=self.getDatalogFacts(facts)
    drules=self.getDatalogRules()
    allFacts=runDatalog(dfacts, drules, props)
    factsToKB(allFacts)
    n=allFacts-self.oldfacts
    self.oldfacts=allFacts
    return len(n)

```

Лістинг Ц.6 – class_Dependence.py

```

1 #-*- coding: utf-8 -*-
2 class Dependence(Agent):
3     """Dependence. Базовий клас статистичних залежностей"""
4     def __init__(self, name):
5         super(Dependence,self).__init__(name)
6         self.Xpar=None # незалежна змінна
7         self.Ypar=None # залежна змінна
8         self.X=[]
9         self.Y=[]
10        self.source=None
11
12    def plot(self,plt):
13        try:
14            a=self.linregress()
15            plt.plot(self.X,self.Y,'ko')
16            x=np.array([min(self.X),max(self.X)])
17            y=a[0]*x+a[1]
18            plt.plot(x,y,'k-')
19            plt.xlabel('X')
20            plt.ylabel('Y')
21            plt.show()

```

```

22     except:
23         pass
24
25     def linregress(self):
26         from scipy import stats
27         return stats.linregress(self.X, self.Y)
28
29     def fromModels(self): # повертає точки з моделей
30         if KB.get(self.source).__class__.__name__!='Model': return
31         models=getClassObjects('Model')
32         for k in models:
33             m=KB[k]
34             if not m.isSibling(self.source): continue
35             if self.Xpar not in m.paramsIn: continue
36             if self.Ypar not in m.paramsOut: continue
37             y=m.paramsOut[self.Ypar]
38             if y==None: continue
39             x=m.paramsIn[self.Xpar]
40             try:
41                 i=self.X.index(x)
42                 self.Y[i]=y # додати значення Y
43             except ValueError:
44                 self.X.append(x) # додати точку
45                 self.Y.append(y)
46         XY=zip(self.X, self.Y)
47         XY.sort()
48         self.X=[x for x,y in XY]
49         self.Y=[y for x,y in XY]
50
51     def modelExist(self, x): # модель зі значенням x існує?
52         models=getClassObjects('Model')
53         for m in models:
54             if KB[m].paramsIn.get(self.Xpar)==x:
55                 return True
56         return False
57
58     def toModels(self): # динамічно створити моделі, якщо y=None
59         if KB.get(self.source).__class__.__name__!='Model': return
60         for x,y in zip(self.X,self.Y):
61             if y!=None or self.modelExist(x): continue
62             k=autoKey("model")
63             KB[k]=KB[self.source].__class__(k)
64             KB[k].paramsIn[self.Xpar]=x
65             print 'Model created'
66
67     def rule(self):
68         self.fromModels()
69         self.toModels()
70         if None in self.Y: return
71         if not len(self.Y)>1: return
72         slope, intercept, r_value, p_value, std_err=self.linregress()

```

```

73     #print slope, intercept, r_value, p_value, std_err
74     if r_value**2<0.5: return False
75
76     try:
77         if slope>0: # додатна кореляція
78             KB[KB[self.Xpar].factorHigh].isCause.add(
79                 KB[self.Ypar].factorHigh)
80         else:
81             KB[KB[self.Xpar].factorLow].isCause.add(
82                 KB[self.Ypar].factorHigh)
83     except KeyError:
84         pass

```

ЛІСТИНГ Ц.7 – class_DependenceMulti.py

```

#-*- coding: utf-8 -*-
class DependenceMulti(Agent):
    """DependenceMulti"""
    def __init__(self, name):
        super(DependenceMulti,self).__init__(name)
        self.Xpars=None # незалежні змінні
        self.Ypars=None # залежні змінні
        self.X=[]
        self.Y=[]
        self.source=None

    def toDataFrame(self):
        import pandas as pd
        XY=self.X+self.Y
        return pd.DataFrame(data = zip(*XY), columns=self.Xpars+self.Ypars)

    def fromDataFrame(self,df):
        for i,xp in enumerate(self.Xpars):
            self.X[i]=[x for x in df[xp]]
        for i,yp in enumerate(self.Ypars):
            self.Y[i]=[y for y in df[yp]]

    def linregress(self, yp): # лінійна регресія для параметра yp
        X=np.array(self.X).T
        Y=self.Y[self.Ypars.index(yp)]
        from sklearn import linear_model
        reg = linear_model.LinearRegression()
        reg.fit(X, Y)
        return reg.coef_, reg.score(X, Y)

    def byModels(self):
        if KB.get(self.source).__class__.__name__!='Model': return
        df=self.toDataFrame()
        model=KB[self.source].__class__('tmp')
        print 'Temporary model created'

```

```

for i in df.index: # for all points
    for xp in self.Xpars:
        model.paramsIn[xp]=df[xp][i]
        print xp, '=', df[xp][i]
    for yp in self.Ypars:
        model.paramsOut[yp]=None
    model.rule()
    for yp in self.Ypars:
        df[yp][i]=model.paramsOut[yp]
        print yp, '=', df[yp][i]
self.fromDataFrame(df)

def simModel(self,x,yp): # симуляція моделі
    model=KB[self.source].__class__('tmp')
    print 'Temporary model created', x
    for i,xp in enumerate(self.Xpars):
        model.paramsIn[xp]=x[i]
    model.rule()
    print 'y=',model.paramsOut[yp]
    return model.paramsOut[yp]

def optimize(self,yp="коефіцієнт запасу втомної міцності",
bounds=[[2.5,3.5],[12.7, 13.26]], maximize=True):
    from scipy.optimize import differential_evolution # мінімізація
    s = -1 if maximize else 1
    #res=minimize(lambda x,yp: s*self.simModel(x,yp), x0=[sum(p)/2.0
for p in bounds], args=(yp,), method="L-BFGS-B", bounds=bounds,
options=dict(maxfun=10,eps=0.1))
    res=differential_evolution(lambda x,yp: s*self.simModel(x,yp),
args=(yp,), bounds=bounds, maxiter=5, popsize=5, tol=0.1)
    return res

def rule(self):
    if None not in [item for sublist in self.Y for item in sublist]:
return
    self.byModels()

```

Лістинг Ц.8 – class_Fact.py

```

#-*- coding: utf-8 -*-
class Fact(Agent):
    """Fact. Базовий клас фактів"""
    def __init__(self, name):
        super(Fact,self).__init__(name)
        self.subject=None
        self.predicate=None
        self.object=None
        self.sources=set()
    def rule(self):
        for k in KB:
            if k!=self.predicate: continue

```

```

if KB[k].__class__.__name__!='Property': continue
if self.subject not in KB[k].domain: continue
if self.object not in KB[k].range: continue
try:
    if KB[k].FunctionalProperty:
        KB[self.subject].__dict__[k]=self.object
    else:
        KB[self.subject].__dict__[k].add(self.object)
    break
except KeyError:
    pass

```

Лістинг Ц.9 – class_Factor.py

```

1 #-*- coding: utf-8 -*-
2 class Factor(Agent):
3     """Factor. Базовий клас факторів"""
4     def __init__(self, name):
5         super(Factor,self).__init__(name)
6         self.subClassOf=set()
7         self.isCause=set()
8         self.isEffect=set()
9         self.isContraryOf=set()
10        # 'subClassOf(O,B) :- subClassOf(A,B),subClassOf(O,A)'
11        # def rule(self):
12        #     "Правило установлює атрибути в підкласах, якщо їх немає"
13        #     from tools import setAtrSubClasses
14        #     return any( [setAtrSubClasses(__name__,"isCause",set()),
15        #                 setAtrSubClasses(__name__,"isEffect",set()) ])
16
17        # def rule(self):
18        #     "Тільки для тестування прямого виведення. Ви можете
19        #     протестувати цей алгоритм шляхом виключення агентів Datalog
20        #     (active=False)."
21        #     for k in KB:
22        #         if KB[k].__class__.__name__!='Factor': continue
23        #         if k not in self.isCause: continue
24        #         if self.__name__ not in KB[k].__dict__['isEffect']:
25        #             KB[k].__dict__['isEffect'].add(self.__name__)
26        #         return True

```

Лістинг Ц.10 – class_Model.py

```

#-*- coding: utf-8 -*-
class Model(Agent):
    """Model.ШН19 ГОСТ 13877-96"""
    def __init__(self, name):
        super(Model,self).__init__(name)
        self.paramsIn={"радіус скруглень зарізьбової канавки":3.5,
                       "зовнішній радіус різьби ніпеля":13.26}

```

```

self.paramsOut={'коефіцієнт запасу втомної міцності':None}
def parseOutput(self,s,t):
    "Читає результат у змінну t: parseOutput(s,t='FOS')"
    t=t+'='
    for r in s.splitlines():
        if r.startswith(t):
            return float(r.partition(t)[2])
def isSibling(self,model):
    "Моделі відрізняється знач. не більше ніж одного парам."
    a=self.paramsIn
    b=KB[model].paramsIn
    return isSibling(a,b)
def rule(self):
    if None not in self.paramsOut.values(): return
    from subprocess import check_output
    p=r"e:\Anaconda2\python.exe"
    m=r"e:\!Kopey_Documents\Python_projects\ThreadsOCC\main.py"
    cnvName={"радіус скруглень зарізьбової канавки":"r3n",
            "зовнішній радіус різьби ніпеля":"d_n",
            "коефіцієнт запасу втомної міцності":"FOS"}
    args=[cnvName[k]+'='+str(v) for k,v in self.paramsIn.iteritems()]
    s=check_output([p, m]+args, shell=True)
    for k in self.paramsOut:
        self.paramsOut[k]=self.parseOutput(s,cnvName[k])

```

Лістинг Ц.11 – class_Parameter.py

```

#-*- coding: utf-8 -*-
class Parameter(Agent):
    """Parameter. Базовий клас ..."""
    def __init__(self, name):
        super(Parameter,self).__init__(name)
        self.factorHigh=None
        self.factorLow=None

```

Лістинг Ц.12 – Datalog_inference.py

```

#-*- coding: utf-8 -*-
"""inference. Логічне виведення за допомогою Datalog"""

```

Лістинг Ц.13 – Dependence_dependence1.py

```

#-*- coding: utf-8 -*-
"""dependence1 ШН19 ГОСТ 13877-96"""
Xpar="довжина зарізьбової канавки"
Ypar="логарифм циклічної довговічності"
X=[15.0, 25.0, 35.0, 45.0]
Y=[4.3, 5.4, 6.1, 7.0]
source="Копей В.Б. Технологічний аудит та резерви виробництва. - № 6/2 (8).
- 2012. - С.7-8."

```

Лістинг Ц.14 – Dependence_dependence2.py

```
#-*- coding: utf-8 -*-
"""dependence2 ШН19 ГОСТ 13877-96"""
Храp="радіус скруглень зарізьбової канавки"
Ураp="коефіцієнт запасу втомної міцності"
Х=[0.5, 1.5, 2.5, 3.5]
#У=[None, None, None, None]
У=[-41.5474014244, -18.9878897227, -13.7151432304, -11.3376607349]
source="ШН19ГОСТ13877-96"
```

Лістинг Ц.15 – Dependence_dependence3.py

```
#-*- coding: utf-8 -*-
"""dependence3 ШН19 ГОСТ 13877-96"""
Храp="зовнішній радіус різьби ніпеля"
Ураp="коефіцієнт запасу втомної міцності"
Х=[12.7, 12.84, 12.98, 13.12, 13.26]
#У=[None, None, None, None, None]
У=[-12.6285730685, -12.1345249073, -12.0526604823, -11.982904361, -
11.9369383163]
source="ШН19ГОСТ13877-96"
```

Лістинг Ц.16 – DependenceMulti_dependence4.py

```
#-*- coding: utf-8 -*-
"""dependence4 ШН19 ГОСТ 13877-96"""
Храps=["радіус скруглень зарізьбової канавки",
"зовнішній радіус різьби ніпеля"]
Х=[[00.5, 01.5, 02.5, 03.5, 00.50, 01.50, 02.50, 03.50],
[12.7, 12.7, 12.7, 12.7, 13.26, 13.26, 13.26, 13.26]]
Ураps=["коефіцієнт запасу втомної міцності"]
#У=[[None, None, None, None, None, None, None, None]]
У=[[-36.0, -18.1, -14.2, -12.1, -41.5, -19.0, -13.7, -11.3]]
source="ШН19ГОСТ13877-96"
```

Лістинг Ц.17 – Fact_конц_напр_спричинює зменшення цикл_міцності.py

```
#-*- coding: utf-8 -*-
"""Fact.конц_напр_спричинює зменшення цикл_міцності"""
subject="концентрація напружень"
predicate="isCause"
objeсt="зменшення циклічної міцності"
```

Лістинг Ц.18 – Factor_відсутність корозійного пошкодження.py

```
#-*- coding: utf-8 -*-
"""відсутність корозійного пошкодження"""
isContraryOf={"корозійне пошкодження"}
```

Лістинг Ц.19 – Factor_збільшення довжини зарізьбової канавки.py

```
#-*- coding: utf-8 -*-
"""збільшення довжини зарізьбової канавки"""
```

Лістинг Ц.20 – Factor_збільшення радіуса скруглень зарізьбової канавки.py

```
#-*- coding: utf-8 -*-
"""збільшення радіуса скруглень зарізьбової канавки"""
```

Лістинг Ц.21 – Factor_збільшення циклічної міцності.py

```
#-*- coding: utf-8 -*-
"""збільшення циклічної міцності"""
isContraryOf={'зменшення циклічної міцності'}
```

Лістинг Ц.22 – Factor_зменшення циклічної міцності.py

```
#-*- coding: utf-8 -*-
"""зменшення циклічної міцності"""
isEffect={'концентрація напружень'}
```

Лістинг Ц.23 – Factor_концентрація напружень.py

```
#-*- coding: utf-8 -*-
"""концентрація напружень"""
```

Лістинг Ц.24 – Factor_корозійна язва.py

```
#-*- coding: utf-8 -*-
"""корозійна язва"""
subclassOf={"корозійне пошкодження"}
```

Лістинг Ц.25 – Factor_корозійне пошкодження.py

```
#-*- coding: utf-8 -*-
"""корозійне пошкодження"""
isCause={"концентрація напружень"}
```

Лістинг Ц.26 – Model_ШН19ГОСТ13877-96.py

```
#-*- coding: utf-8 -*-
"""Model ШН19 ГОСТ 13877-96"""
```

Лістинг Ц.27 – Model_ШН19ГОСТ13877-96_r3n=2.5.py

```
#-*- coding: utf-8 -*-
"""Model_ШН19ГОСТ13877-96_r3n=2.5"""
paramsIn["радіус скруглень зарізьбової канавки"]=2.5
#paramsIn['d_n']=13.26
```


Лістинг Ц.28 – Parameter_довжина зарізьбової канавки.py

```
#-*- coding: utf-8 -*-
"""довжина зарізьбової канавки ШН19 ГОСТ 13877-96"""
factorHigh="збільшення довжини зарізьбової канавки"
factorLow="зменшення довжини зарізьбової канавки"
```

Лістинг Ц.29 – Parameter_коефіцієнт запасу втомної міцності.py

```
#-*- coding: utf-8 -*-
"""коефіцієнт запасу втомної міцності"""
factorHigh="збільшення циклічної міцності"
factorLow="зменшення циклічної міцності"
```

Лістинг Ц.30 – Parameter_логарифм циклічної довговічності.py

```
#-*- coding: utf-8 -*-
"""логарифм циклічної довговічності"""
factorHigh="збільшення циклічної міцності"
factorLow="зменшення циклічної міцності"
```

Лістинг Ц.31 – Parameter_радіус скруглень зарізьбової канавки.py

```
#-*- coding: utf-8 -*-
"""радіус скруглень зарізьбової канавки ШН19 ГОСТ 13877-96"""
factorHigh="збільшення радіуса скруглень зарізьбової канавки"
factorLow="зменшення радіуса скруглень зарізьбової канавки"
```

Лістинг Ц.32 – Property_factorHigh.py

```
#-*- coding: utf-8 -*-
"""Property. Властивість 'factorHigh'"""
inverseOf="factorLow"
domain={"Parameter"}
range={"Factor"}
FunctionalProperty=True
```

Лістинг Ц.33 – Property_factorLow.py

```
#-*- coding: utf-8 -*-
"""Property. Властивість 'factorLow'"""
inverseOf="factorHigh"
domain={"Parameter"}
range={"Factor"}
FunctionalProperty=True
```

Лістинг Ц.34 – Property_isCause.py

```

#-*- coding: utf-8 -*-
"""isCause. Властивість 'є причиною'"""
inverseOf="isEffect"
domain={"Factor"}
range={"Factor"}

datalogRules_=KB['isCause'].datalogRules
def datalogRules():
    r={'isCause(X,Y) <= isCause(B,Y) & subClassOf(X,B)',
      'isEffect(X,Y) <= isEffect(B,Y) & subClassOf(X,B)'}
    return r|datalogRules_()

```

Лістинг Ц.35 – Property_isContraryOf.py

```

#-*- coding: utf-8 -*-
"""isContraryOf. Властивість 'є протилежністю'"""
domain={"Factor"}
range={"Factor"}
def datalogRules():
    r={'isCause(X, Y) <= isContraryOf(X, NX) & isCause(NX, NY) &
isContraryOf(Y, NY)',
      'isEffect(X, Y) <= isContraryOf(X, NX) & isEffect(NX, NY) &
isContraryOf(Y, NY)'}
    return r

```

Лістинг Ц.36 – Property_isEffect.py

```

#-*- coding: utf-8 -*-
"""isEffect. Властивість 'є наслідком'"""
inverseOf="isCause"
domain={"Factor"}
range={"Factor"}

```

Лістинг Ц.37 – Property_object.py

```

#-*- coding: utf-8 -*-
"""Property. Властивість 'об'єкт'"""
domain={"Fact"}
range={"Factor"}
FunctionalProperty=True

```

Лістинг Ц.38 – Property_source.py

```

#-*- coding: utf-8 -*-
"""Property. Властивість 'source'"""
domain={"Dependence"}
range={"Model", "str"}
FunctionalProperty=True

```

ЛІСТИНГ Ц.39 – Property_subClassOf.py

```

#-*- coding: utf-8 -*-
"""subClassOf. Властивість 'є підкласом'"""
domain={"Factor"}
range={"Factor"}

# def datalogRules():
#     r={'isCause(X,Y) <= isCause(B,Y), subClassOf(X,B)'}
#     return r

```

ЛІСТИНГ Ц.40 – Property_subject.py

```

#-*- coding: utf-8 -*-
"""Property. Властивість 'subject'"""
domain={"Fact"}
range={"Factor"}
FunctionalProperty=True

```

ЛІСТИНГ Ц.41 – Property_Xpar.py

```

#-*- coding: utf-8 -*-
"""Property. Властивість 'Xpar'"""
domain={"Dependence"}
range={"Parameter"}
FunctionalProperty=True

```

ЛІСТИНГ Ц.42 – Property_Ypar.py

```

#-*- coding: utf-8 -*-
"""Property. Властивість 'Ypar'"""
domain={"Dependence"}
range={"Parameter"}
FunctionalProperty=True

```

ДОДАТОК Ч
Акти впровадження результатів роботи

ЗАТВЕРДЖУЮ

Головний інженер

НГВУ "Долинанатогаз"

Яремко І.Я.

12.07 2016 р.



АКТ

про впровадження інформаційної системи підтримки життєвого циклу свердловинних штангових насосних установок (СШНУ)

Комісія у складі:

Голова - головний інженер Яремко І.Я.;

Члени комісії – головний механік НГВУ "Долинанатогаз" Петрів М.В., провідний інженер ЦІТС, к.т.н. Кузьмін О.О., доцент кафедри КМВ ІФНТУНГ, к.т.н. Копей В.Б., завідувач кафедри НГО ІФНТУНГ, д.т.н. Копей Б.В. цим Актом засвідчує, що інформаційна система підтримки життєвого циклу СШНУ, яка розроблена Копеем В.Б. та Копеем Б.В., впроваджена в НГВУ "Долинанатогаз" з метою підвищення якості експлуатації СШНУ та забезпечення її надійності. Нижче перелічені впроваджені компоненти інформаційної системи та основні поточні результати їх впровадження:

1. Головний модуль інформаційної системи та експертна система з проблем надійності і довговічності різьбових з'єднань. Використані для підбору оптимальних методів і засобів забезпечення надійності різьбових з'єднань насосних штанг.

2. Динамічна модель СШНУ. Виявлено відповідність результатів розрахунку моделі та експериментальної динамограми СШНУ. Модель використано для уточнення компонування штангової колони.

3. Параметричні скінченно-елементні моделі різьбових з'єднань нафтового обладнання (насосних штанг, насосно-компресорних труб, замкові

з'єднання) та програмні засоби їх побудови. Шляхом моделювання отримані рекомендації щодо вибору оптимальних моментів згвинчування насосних штанг в залежності від умов їх експлуатації.

4. Параметричні скінченно-елементні моделі з'єднань склопластикового тіла насосної штанги зі сталевим ніпелем. Шляхом моделювання отримані допустимі навантаження на склопластикові насосні штанги з врахуванням особливостей конструкції з'єднання.

5. Параметричні скінченно-елементні моделі протекторів для насосних штанг. Шляхом розрахунку гідродинамічних моделей отримані сили гідродинамічного опору протекторів різної конструкції та подані рекомендації щодо області їх раціонального використання.

Члени комісії

головний механік

НГВУ "Долинанафтогаз"

провідний інженер ЦІТС к.т.н.


доцент кафедри КМВ ІФНТУНГ, к.т.н.

завідувач кафедри НГО ІФНТУНГ, д.т.н

 Петрів М.В.

 Кузьмін О.О.

 Копей В.Б.

 Копей В.В.

« 12 » 07 2016 р.

ЗАТВЕРДЖУЮПроректор з наукової роботи
проф. Чудик І. І.
"05" березня 2020 р.**АКТ**

Впровадження у навчальний процес результатів дисертаційної роботи Копея Володимира Богдановича на тему "Науково-методологічні основи автоматизованого проектування обладнання штангової свердловинної насосної установки", яка подається на здобуття наукового ступеня доктора технічних наук

Наукові та прикладні результати дисертації доцента кафедри комп'ютеризованого машинобудування Копея В. Б. використовуються для підготовки фахівців освітньо-кваліфікаційного рівня магістр спеціальності 131 "Прикладна механіка" освітньо-професійної програми "Комп'ютеризовані і роботизовані технології машинобудування" під час вивчення дисциплін "Інформаційне забезпечення САПР", "Методи моделювання та симуляції кінематики та динаміки машин" і під час виконання магістерських робіт.

Директор інституту інженерної механіки
канд. техн. наук, професор

Л. І. Романишин

Завідувач кафедри
комп'ютеризованого машинобудування,
д-р техн. наук, професор

В. Г. Панчук