ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА
(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій
(повне найменування факультету)

Кафедра механіки, автоматизації та інформаційних технологій
(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«Моделювання коливань механічної системи з одним ступенем свободи із використанням технології WPF»

«Modeling of oscilllations of a mechanical system with one degree of freedom using WPF technology»

Виконав(ла): здобувач(ка) денної (очної) форми навчання спеціальності 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Даниленко Анастасія Олексіївна
(прізвище, ім'я, по-батькові здобувача)

Керівник викладач Недєва О.А _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент старший викладач Косирева Л.А.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
механіки, автоматизації та інформаційних технологій
№ ___ від ____.____. 20___ р.

Завідувачка кафедри
_____ Алла РАЧИНСЬКА
(підпис)

Захищено на засіданні ЕК № _____
протокол № __від ____.____.20___ р.

Оцінка_____/____/_____
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК
_____ Микола МАЛАКСІАНО
(підпис)

Одеса 2024

АНОТАЦІЯ

В рамках даної кваліфікаційної роботи розглядається важливість та ефективність чисельного моделювання в сучасному дослідженні фізичних явищ і технологічних процесів. Особлива увага приділяється чисельному моделюванню коливального руху механічних систем з одним ступенем свободи, оскільки для забезпечення кількісно правильних передбачень важливо, щоб моделі відображали не лише окремі процеси у системі, але й їх взаємодію відповідно до реальних умов.

Робота висвітлює ключові аспекти процесу моделювання, від постановки задачі до розробки та тестування програмного забезпечення для відтворення експериментів. Звертаючи особливу увагу на адекватне відображення взаємодії складних фізичних процесів у системі, ця робота обґрунтовує не лише значущість чисельного моделювання, а й ефективність використання програмної візуалізації для подання результатів. Шляхом поєднання чисельного моделювання та програмної візуалізації, досягається не лише процес отримання даних, а й їх інтерпретація та аналіз, що робить цей підхід надзвичайно корисним для наукових та інженерних досліджень.

ANNOTATION

This qualification work considers the importance and effectiveness of numerical modeling in the modern study of physical phenomena and technological processes. Particular attention is paid to the numerical modeling of the oscillatory motion of mechanical systems with one degree of freedom, since in order to provide quantitatively correct predictions, it is important that the models reflect not only the individual processes in the system, but also their interaction in accordance with real conditions.

The paper covers the key aspects of the modeling process, from problem formulation to the development and testing of software to reproduce experiments. Paying special attention to the adequate representation of the interaction of complex physical processes in the system, this paper substantiates not only the importance of numerical modeling, but also the effectiveness of using software visualization to present the results. By combining numerical modeling and software visualization, not only the process of data acquisition is achieved, but also its interpretation and analysis, which makes this approach extremely useful for scientific and engineering research.

ЗМІСТ

УМОВНІ ПОЗНАЧЕННЯ І ПОЯСНЕННЯ

Схема механічної системи у стані спокою можна побачити на рис. 1.

Як узагальнену координату ξ використано вертикальне становище вантажу **1**.

Нижче подано фізичні характеристики цієї механічної системи, які вважаються заданими або можуть бути встановлені під час чисельного експерименту. До них залежать маси тіл системи – m_1, m_2 и m_3, а також c – коефіцієнт відновлення для пружини.

R – зовнішній радіус тіла 2;

r – внутрішній радіус тіла 2, де $r < R$;

L – довжина тіла 3 (однорідний стрижень), де $R \le L$;

L_c – дистанція від точки підвісу тіла 3 до точки закріплення пружини;

Δ_0 – довжина недеформованої пружини, розміщеної горизонтально;

Δ_{cm} – деформація пружини у стані спокою системи;

H – довжина зв'язку (нерозтяжний стрижень)

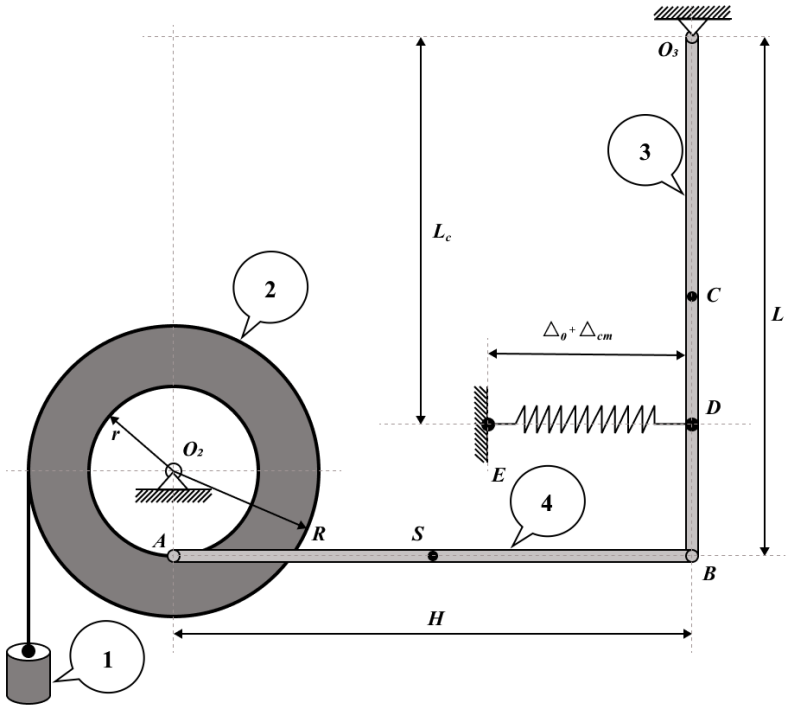


Рисунок 1 – Система у стані спокою

ξ – узагальнена координата;

$\dot{\xi}$ – узагальнена швидкість;

T – кінетична енергія механічної системи;

$П$ – її потенційна енергія;

Q_ε – узагальнена сила (за наявності непотенційних сил).

1 ВСТУП

1.1 Формулювання проблематики дослідження

У сучасному дослідженні фізичних явищ і технологічних процесів чисельне моделювання є незамінним та ефективним методом. Сьогодні цей підхід інтенсивно розвивається по всьому світу: створюються вдосконалені моделі, розробляються нові чисельні алгоритми, проводяться численні обчислювальні експерименти, проектуються кластерні системи. Стрімкий розвиток комп'ютерної техніки, доступ до високопродуктивних обчислювальних ресурсів, а також прогрес в операційних системах, що підтримують паралельні обчислювальні процеси, ще більше підсилюють інтерес до проектування та реалізації чисельних експериментів у галузях фізики та механіки.

Чисельне моделювання дозволяє передбачити і вивчити поведінку складних фізичних та механічних систем. Для отримання кількісно точних передбачень моделювання повинно адекватно описувати як окремі процеси, що відбуваються в системі, так і їх взаємодію. Після створення та тестування моделі її можна використовувати для інтерпретації вимірювань і спостережень, розширення теоретичних моделей на нові параметричні області, а також для оцінки нових ідей та рішень. Найважливішим застосуванням чисельних моделей є перевірка повноти нашого розуміння фізико-механічних систем шляхом порівняння результатів розрахунків з експериментальними даними або спрощеними аналітичними рішеннями.

Навіть найпростіші натурні експерименти часто є складними та ресурсоємними. Модельні рівняння, що описують такі експерименти, зазвичай не мають аналітичних розв'язків і потребують чисельного підходу для свого розв'язання.

У цій кваліфікаційній роботі розглядається важливість та ефективність чисельного моделювання в сучасному дослідженні фізичних явищ і технологічних процесів. Особлива увага приділяється чисельному

моделюванню коливального руху механічних систем з одним ступенем свободи, оскільки для забезпечення кількісно правильних передбачень важливо, щоб моделі відображали не лише окремі процеси у системі, але й їх взаємодію відповідно до реальних умов.

Дана кваліфікаційна робота присвячена дослідженню чисельного моделювання коливального руху механічної системи з одним ступенем свободи та її програмній візуалізації. Приділяючи особливу увагу точному відображенню взаємодії складних фізичних процесів у системі, ця робота не тільки демонструє важливість чисельного моделювання, але й показує ефективність використання програмної візуалізації для представлення результатів.

1.2 Актуальність теми

Є низка актуальних проблем, пов'язаних із коливаннями механічних систем, які можна вирішити за допомогою комп'ютерного моделювання. Коливання механічних систем є складною задачею, що стикається з різними викликами і вимагає вдосконалення методів аналізу та проектування.

Однією з головних проблем є визначення динамічної стійкості механічних систем. При високих швидкостях або в умовах змінних навантажень системи можуть втратити стійкість, що може призвести до аварій. Наприклад, маятник, закріплений на підвісі, може втратити стійкість і вийти з рівноваги під впливом періодичних зовнішніх сил. Іншим прикладом є пружинний механізм, який під дією змінних навантажень може почати коливатись з небезпечною амплітудою, що призведе до його пошкодження. Комп'ютерне моделювання дозволяє досліджувати вплив різних факторів на стійкість системи, виявляти критичні точки і розробляти методи їх запобігання.

Для підвищення ефективності та зниження ваги механічних систем необхідна оптимізація їх конструкції. Комп'ютерне моделювання дозволяє проводити чисельні експерименти з різними варіантами конструкцій і

матеріалів, оцінювати їх вплив на коливання системи і знаходити оптимальні рішення.

Коливання механічних систем можуть бути викликані як зовнішніми навантаженнями (вібрацією або ударами), так і внутрішніми факторами (неоднорідністю матеріалу або недосконалістю з'єднань). Наприклад, маятник може бути виведений з рівноваги шляхом зсуву вантажу, що призводить до його коливань. Комп'ютерне моделювання дозволяє вивчати вплив цих факторів на коливання системи, ідентифікувати критичні точки та розробляти заходи для зниження їхнього впливу.

З розвитком технологій виникають усе складніші механічні системи, такі як роботи, автоматизовані виробничі лінії та транспортні системи. Аналіз і проектування таких систем потребують нових підходів і методів. Комп'ютерне моделювання дає змогу досліджувати взаємодію складних компонентів системи, оцінювати їх динаміку та ефективність, виявляти проблемні зони та знаходити способи їх усунення.

Усі ці виклики сучасного часу підкреслюють необхідність постійного розвитку нових стратегій для аналізу та проектування механічних систем. Використання комп'ютерного моделювання є ключовим інструментом у розв'язанні цих завдань, оскільки воно дозволяє створювати докладні та реалістичні моделі, проводити чисельні експерименти і знаходити оптимальні рішення.

Таким чином, застосування комп'ютерного моделювання вирішує широкий спектр актуальних проблем, пов'язаних з коливаннями механічних систем, і сприяє постійному вдосконаленню процесів аналізу та проектування в контексті інформаційних систем. Розвиток нових підходів до комп'ютерного моделювання сприятиме впровадженню більш точних, швидких та ефективних методів розрахунків та оптимізації механічних систем.

1.3 Мета та завдання

Метою даної роботи є розробка та реалізація програмної моделі коливального руху механічної системи з одним ступенем свободи з використанням технології WPF (Windows Presentation Foundation).

Задана механічна система з одним ступенем свободи може здійснювати коливання відносно положення рівноваги.

У початковий момент часу ($\boldsymbol{t} = 0$) систему виводять із положення рівноваги шляхом вертикального зсуву вантажу **1**, що не порушує цілісності самої системи, а швидкості всіх точок системи під час $\boldsymbol{t} = 0$ дорівнюють нулю (система у стані спокою). У наступний момент часу система починає рух (коливання), перебуваючи під дією лише консервативних сил.

Для досягнення цієї мети були поставлені наступні завдання:

а) Визначення диференціальних рівнянь руху та взаємодії механічних елементів.

б) Аналіз динамічної поведінки механічної системи.

в) Огляд технології WPF та її можливостей для візуалізації механічних процесів.

г) Розробка програмної реалізації математичної моделі за допомогою WPF.

д) Проектування та реалізація графічного інтерфейсу для візуалізації результатів моделювання.

е) Тестування програмної моделі.

У результаті виконання цих завдань буде створено ефективний інструмент для моделювання та візуалізації коливань механічних систем, що може бути застосований у різних інженерних галузях.

2 ПОБУДОВА МАТЕМАТИЧНОЇ МОДЕЛІ ДЛЯ МЕХАНІЧНОЇ СИСТЕМИ

2.1 Опис та структурний аналіз механічної системи з одним ступенем свободи та її компонентів

Розглянута механічна система має один ступінь свободи (див. рис. 2.1). Три фізичні тіла системи з'єднані між собою за допомогою геометричних зв'язків, які визначають допустимі траєкторії їхнього руху та обмежують їхні положення відносно одне одного. Пружина забезпечує повернення системи до рівноважного стану після збурень, створюючи еластичну силу, що протидіє деформаціям. Геометричні зв'язки відіграють ключову роль у визначенні розподілу сил і моментів у системі, впливаючи на її динамічні характеристики.

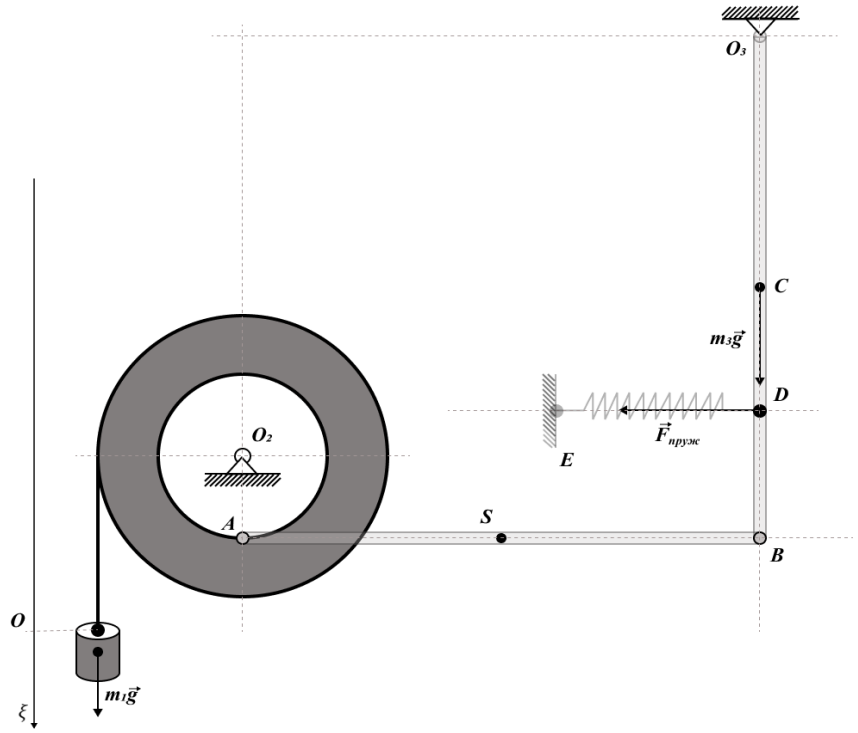


Рисунок 2.1 – Сукупність зовнішніх сил, що підтримують стан спокою

2.2 Визначення диференціальних рівнянь руху та взаємодії частин механічних елементів

Щоб дослідити динаміку руху заданої системи будемо використовувати рівняння Лагранжа 2-го роду:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\xi}}\right) - \frac{\partial T}{\partial \xi} = -\frac{\partial \Pi}{\partial \xi} + Q_\xi$$

В якості узагальненої координати ξ обираємо вертикальне відхилення вантажу **1** від положення рівноваги в напрямку дії сили тяжіння $m_1\vec{g}$. Відповідно узагальнену швидкість вантажу **1** позначимо $\dot{\xi}$. Кінетична енергія цього тіла під час його вертикального поступального руху буде: $T_1 = \frac{1}{2}m_1\dot{\xi}^2$.

Можемо виразити кінематичні характеристики інших тіл через узагальнену швидкість $\dot{\xi}$ вантажу **1**, оскільки вказана система має один ступінь свободи. Тіло **2** здійснює обертальний рух навколо горизонтальної осі, що проходить через його геометричний центр O_2.

Положення тіла 2 визначатиметься кутом повороту φ навколо даної осі. Таким чином кутова швидкість φ тіла 2 може бути розрахована за формулою Ейлера: $\varphi = \dot{\xi}/R$.

За допомогою формули $T_2 = \frac{1}{2}J_2\dot{\varphi}^2$, де J_2 – момент інерції тіла 2 щодо горизонтальної осі обертання, що проходить через центр O_2, можемо обчислити кінетичну енергію тіла 2.

Величина $V_A = r * \dot{\varphi} = \left(\frac{r}{R}\right) * \dot{\xi}$ визначає швидкість точки А, яка лежить на внутрішньому радіусі тіла 2.

Тіло 3, яке представляє однорідний стрижень довжиною L, також буде здійснювати обертальний рух. Горизонтальна вісь проходить через точку підвісу O_3. За ψ приймемо кут відхилення тіла 3 від вертикалі. А швидкість точки B стрижня 3 обчислюється за формулою $V_B = L * \dot{\psi}$, де $\dot{\psi}$ - кутова швидкість цього тіла.

Кінетична енергія тіла 3 можна обчислити за допомогою формули $T_3 = \frac{1}{2}J_3\dot{\psi}^2$, де $J_3 = m_3L^2/3$ є моментом інерції відносно горизонтальної осі обертання, яка проходить через точку O_3.

Розглядається механічна система з одним ступенем свободи. Отож всі кутові швидкості тіл мають виражатися через узагальнену швидкість.

На рис. 3 можна побачити, що тіла 2 та 3 з'єднуються за допомогою стрижня AB довжини H. Стрижень AB вважається невагомим, а в точках A та B він має сферичні шарніри. Звідки витікає, що цей стрижень є бездоганним недеформованим зв'язком.

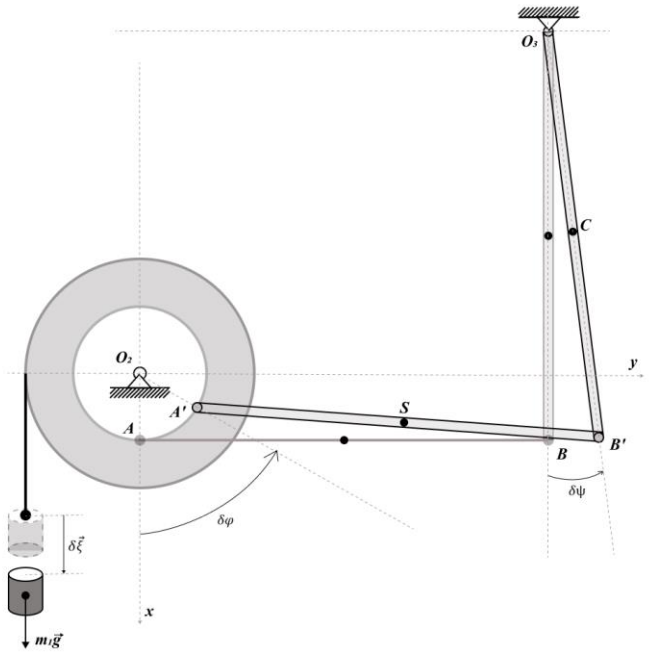


Рисунок 2.2 – Віртуальні рухи тіл механічної системи

На рис. 2.2 видно, що віртуальне зміщення у напрямку дії сили тяжіння призводить до переміщення тіл 2 і 3. Конкретно, це виявляється у вигляді поворотів на кути $\delta\varphi$ і $\delta\psi$. А точки A і B тіл 2 і 3 відповідно займуть нові положення A' і B'.

Відстань між точками A' і B' залишається незмінною, через жорсткість стрижня AB і це означає, що $|AB| = |A'B'| = H$.

Центр O_3 в осях системи координат O_2xy матиме координати: $x_{O_3} = r - L$ і $y_{O_3} = H$, (що можна побачити з рис. 1).

Координати точки A та B будуть:

$x_A = r * \cos\varphi$ і $y_A = r * \sin\varphi$.

$x_B = r - L + L * \cos\psi$ і $y_B = H + L * \sin\psi$

А співвідношення $(x_B - x_A)^2 + (y_B - y_A)^2 = H^2$ виражає під час руху системи незмінність довжини H стрижня AB. Цьому співвідношенню можна надати форму рівняння зв'язку:

$$f(\varphi,\psi) = (x_B - x_A)^2 + (y_B - y_A)^2 - H^2 = 0, \text{ або}$$

$$f(\varphi,\psi) = \big(r * (1 - \cos\varphi) - L * (1 - \cos\psi)\big)^2 + (H + L * \sin\psi - r * \sin\varphi)^2 - H^2 = 0, \quad (2.1)$$

Тут параметри H, L і r є постійними величинами, заданими значеннями

Фактично через рівняння (2.1) отримуємо неявний зв'язок змінних ξ та ψ з урахуванням того, що $\dot{\varphi} = \dot{\xi}/R$:

$$f(\varphi,\psi) = \left(r * \left(1 - \cos\left(\frac{\xi}{R}\right)\right) - L * (1 - \cos\psi)\right)^2 + \left(H + L * \sin\psi - r * \sin\left(\frac{\xi}{R}\right)\right)^2 - \\ -H^2 = 0, \quad (2.2)$$

За наявності зв'язку, що описується рівнянням (2.1), варіації $\delta\varphi$ і $\delta\psi$ кутових координат φ і ψ повинні задовольняти співвідношення

$$\frac{\partial f}{\partial \varphi}\delta\varphi + \frac{\partial f}{\partial \psi}\delta\psi = 0, \quad (2.3)$$

Знайдемо часткові похідні, що входять до останнього співвідношення:

$$\frac{\partial f}{\partial \varphi} = 2\big(r * (1 - \cos\varphi) - L * (1 - \cos\psi)\big) * r * \sin\varphi - \\ 2(H + L * \sin\psi - r * \sin\varphi) * r * \cos\varphi, \quad (2.4)$$

$$\frac{\partial f}{\partial \psi} = -2\big(r * (1 - \cos\varphi) - L * (1 - \cos\psi)\big) * L * \sin\psi + \\ 2(H + L * \sin\psi - r * \sin\varphi) * L * \cos\psi$$

Переписуємо співвідношення (2.3) для варіацій $\delta\varphi$ і $\delta\psi$ у такому вигляді:

$$\boldsymbol{X} * \delta\varphi - \boldsymbol{Z} * \delta\psi = 0 \text{ або}$$

$$\delta\varphi = \frac{\boldsymbol{X}}{\boldsymbol{Z}} * \delta\psi, \quad (2.5)$$

де

$$\boldsymbol{X}(\varphi,\psi) = 2r\begin{bmatrix}\big(r * (1 - \cos\varphi) - L * (1 - \cos\psi)\big) * \sin\varphi - \\ (H + L * \sin\psi - r * \sin\varphi) * \cos\varphi\end{bmatrix},$$

$$\boldsymbol{Z}(\varphi,\psi) = 2r\begin{bmatrix}\big(r * (1 - \cos\varphi) - L * (1 - \cos\psi)\big) * \sin\psi - \\ (H + L * \sin\psi - r * \sin\varphi) * \cos\psi\end{bmatrix}, \quad (2.6)$$

Співвідношення (2.6) буде так само справедливим і для кутових швидкостей тіл 2 і 3, що означає

$$\dot{\psi} = \frac{d\psi}{dt} = \frac{\boldsymbol{X}}{\boldsymbol{Z}} * \frac{d\varphi}{dt} = \frac{\boldsymbol{X}}{\boldsymbol{Z}} * \dot{\varphi}, \tag{2.7}$$

Таким чином кутові швидкості $\dot{\varphi}$ і $\dot{\psi}$ тіл 2 і 3 виражаються через узагальнену швидкість $\dot{\xi}$:

$$\dot{\varphi} = \frac{1}{R} * \dot{\xi}, \dot{\psi} = \frac{1}{R}\frac{\boldsymbol{X}}{\boldsymbol{Z}} * \dot{\xi}, \tag{2.8}$$

Повна кінетична енергія механічної системи, яку ми розглядаємо, визначається наступним чином:

$$T = T_1 + T_2 + T_3 = \frac{1}{2}(m_1\dot{\xi}^2 + J_2\dot{\varphi}^2 + J_3\dot{\psi}^2).$$

Після підстановки (8) в останню формулу отримуємо:

$$T = \frac{1}{2R^2}\left(m_1R^2 + J_2 + J_3 * \Phi(\varphi, \psi)\right) * \dot{\xi}^2, \tag{2.9}$$

де

$$\Phi = \Phi(\varphi, \psi) = \left[\frac{\boldsymbol{X}}{\boldsymbol{Z}}\right]^2, \tag{2.10}$$

Моменти інерції, які входять у формулу (2.9), записуються наступним чином:

$$J_1 = m_1R^2, J_2 = \frac{m_2r^4 + R^4}{2r^2 + R^2} \text{ і } J_3 = \frac{m_3L^4}{3}, \tag{2.11}$$

де m_2 і m_3 - відповідно маси тіл 2 і 3.

І тоді повна кінетична енергія системи буде:

$$T = \frac{1}{2R^2}(J_1 + J_2 + J_3 * \Phi) * \dot{\xi}^2, \tag{2.12}$$

Розглянемо випадок малих коливань, коли відрізок *AB* здійснює рух, близький до поступального (тобто $V_A = V_B$), основні розрахункові формули істотно спрощуються:

$$\delta\psi = \frac{1}{R}\frac{r}{L} * \delta\xi, \dot{\psi} = \frac{1}{R}\frac{r}{L} * \dot{\xi}, \ \frac{\boldsymbol{X}}{\boldsymbol{Z}} = \frac{r}{L}, \ ,$$

$$\Phi(\varphi, \psi) = \left(\frac{r}{L}\right)^2 \text{ і } T_* = \frac{1}{2R^2}\left(J_1 + J_2 + J_3\frac{r^2}{L^2}\right) * \dot{\xi}^2, \tag{2.13}$$

До рівняння Лагранжа 2-го роду

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\xi}}\right) - \frac{\partial T}{\partial \xi} = -\frac{\partial \Pi}{\partial \xi} + Q_\xi, \tag{2.14}$$

входять часткові похідні кінетичної енергії та потенційної енергії за узагальненою координатою ξ, приватна похідна кінетичної енергії по узагальненій швидкості $\dot{\xi}$ і узагальнена сила Q_ξ (необхідна для непотенційних сил).

Також з (2.12) отримуємо

$$\frac{\partial T}{\partial \xi} = \frac{J_3}{2R^2} * \dot{\xi}^2 * \frac{\partial \Phi}{\partial \xi}, \frac{\partial T}{\partial \dot{\xi}} = \frac{1}{R^2}(J_1 + J_2 + J_3 * \Phi) * \dot{\xi}, \quad (2.15)$$

І вираз для $\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\xi}}\right)$ можна представити в наступному вигляді:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\xi}}\right) = \frac{d}{dt}\left(\frac{1}{R^2}(J_1 + J_2 + J_3 * \Phi) * \dot{\xi}\right) =$$

$$\frac{J_1 + J_2 + J_3 * \Phi}{R^2} * \ddot{\xi} + \frac{J_3}{R^2}\frac{d\Phi}{dt} * \dot{\xi}, \quad (2.16)$$

Для обчислення другого члену в (2.16) скористаємося представленням $\frac{d\Phi}{dt} = \frac{\partial \Phi}{\partial \xi}\dot{\xi} + \frac{\partial \Phi}{\partial t}$. Якщо $\frac{\partial \Phi}{\partial t} - 0$, то отримуємо $\frac{d\Phi}{dt} = \frac{\partial \Phi}{\partial \xi} * \dot{\xi}$ та формула (2.16) буде представлена у вигляді:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\xi}}\right) = \frac{J_1+J_2+J_3*\Phi}{R^2} * \ddot{\xi} + \frac{J_3}{R^2}\frac{\partial \Phi}{\partial \xi} * \dot{\xi}^2.$$

Отримуємо наступний вигляд для лівої частини рівняння Лагранжа:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\xi}}\right) - \frac{\partial T}{\partial \xi} = \frac{J_1+J_2+J_3*\Phi}{R^2} * \ddot{\xi} + \frac{J_3}{R^2}\frac{\partial \Phi}{\partial \xi} * \dot{\xi}^2 - \frac{J_3}{2R^2}\frac{\partial \Phi}{\partial \xi} * \dot{\xi}^2 = \frac{J_1+J_2+J_3*\Phi}{R^2} * \ddot{\xi} +$$

$$\frac{J_3}{2R^2}\frac{\partial \Phi}{\partial \xi} * \dot{\xi}^2.$$

Очевидно (дивитися (2.10)), що $\frac{\partial \Phi}{\partial \xi} = 2\left[\frac{\boldsymbol{X}}{\boldsymbol{Z}}\right] * \frac{\partial\left[\frac{\boldsymbol{X}}{\boldsymbol{Z}}\right]}{\partial \xi} = 2\frac{\boldsymbol{X}}{\boldsymbol{Z^3}} * \left(\frac{\partial \boldsymbol{X}}{\partial \xi} * \boldsymbol{Z} - \boldsymbol{X} * \frac{\partial \boldsymbol{Z}}{\partial \xi}\right)$.

Далі

$$\frac{\partial \boldsymbol{X}}{\partial \xi} = \frac{\partial \boldsymbol{X}}{\partial \varphi} * \frac{\partial \varphi}{\partial \xi} + \frac{\partial \boldsymbol{X}}{\partial \psi} * \frac{\partial \psi}{\partial \xi} \text{ і } \frac{\partial \boldsymbol{Z}}{\partial \xi} = \frac{\partial \boldsymbol{Z}}{\partial \varphi} * \frac{\partial \varphi}{\partial \xi} + \frac{\partial \boldsymbol{Z}}{\partial \psi} * \frac{\partial \psi}{\partial \xi}, \quad (2.17)$$

де відповідно з (2.8):

$$\frac{\partial \varphi}{\partial \xi} = \frac{1}{R} \text{ і } \frac{\partial \psi}{\partial \xi} = \frac{1}{R} * \frac{\boldsymbol{X}}{\boldsymbol{Z}}, \quad (2.18)$$

і

$$\begin{cases} \dfrac{\partial \boldsymbol{X}}{\partial \varphi} = 2r[(r-L)*\cos\varphi + H*\sin\varphi + L*\cos(\varphi-\psi)], \\ \dfrac{\partial \boldsymbol{X}}{\partial \psi} = 2r[-L*\sin\psi\sin\varphi - L*\cos\psi\cos\varphi] = -2rL*\cos(\varphi-\psi), \\ \dfrac{\partial \boldsymbol{Z}}{\partial \varphi} = 2L[r*\sin\psi\sin\varphi + r*\cos\psi\cos\varphi] = 2rL*\cos(\varphi-\psi), \\ \dfrac{\partial \boldsymbol{Z}}{\partial \psi} = 2L[(r-L)*\cos\psi + H*\sin\varphi\psi - r*\cos(\varphi-\psi)]. \end{cases} \tag{2.19}$$

Використовуючи отримані вирази (2.17) - (2.19), отримуємо формула для обчислення $\frac{\partial \Phi}{\partial \xi}$ у наступному вигляді:

$$\frac{\partial \Phi}{\partial \xi} = 2\left[\frac{\boldsymbol{X}}{\boldsymbol{Z}}\right] * \frac{\partial\left[\frac{\boldsymbol{X}}{\boldsymbol{Z}}\right]}{\partial \xi} = \frac{2}{r}\frac{\boldsymbol{X}}{\boldsymbol{Z^3}} * \left[\left(\frac{\partial \boldsymbol{X}}{\partial \varphi} + \frac{\boldsymbol{X}}{\boldsymbol{Z}} * \frac{\partial \boldsymbol{X}}{\partial \psi}\right) * \boldsymbol{Z} - \boldsymbol{X} * \left(\frac{\partial \boldsymbol{Z}}{\partial \varphi} + \frac{\boldsymbol{X}}{\boldsymbol{Z}} * \frac{\partial \boldsymbol{Z}}{\partial \psi}\right)\right], \tag{2.20}$$

Зрештою маємо:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\xi}}\right) - \frac{\partial T}{\partial \xi} = \frac{J_1 + J_2 + J_3 * \left[\frac{\boldsymbol{X}}{\boldsymbol{Z}}\right]^2}{R^2} * \ddot{\xi} +$$

$$\frac{J_3}{R^2}\frac{\boldsymbol{X}}{\boldsymbol{Z^3}} * \left[\left(\frac{\partial \boldsymbol{X}}{\partial \varphi} + \frac{\boldsymbol{X}}{\boldsymbol{Z}} * \frac{\partial \boldsymbol{X}}{\partial \psi}\right) * \boldsymbol{Z} - \boldsymbol{X} * \left(\frac{\partial \boldsymbol{Z}}{\partial \varphi} + \frac{\boldsymbol{X}}{\boldsymbol{Z}} * \frac{\partial \boldsymbol{Z}}{\partial \psi}\right)\right] * \dot{\xi}^2, \tag{2.21}$$

Перейдемо до аналізу потенційної енергії заданої механічної системи.

На фізичні тіла досліджуваної системи діють лише сили тяжіння та пружна сила пружини.

Фізичні тіла в досліджуваній системі відчувають лише сили тяжіння та пружинні сили. Зв'язки в системі вважаються ідеальними, і відсутність сил опору враховується відповідно до початкових умов задачі. Це дозволяє нам стверджувати, що механічна система є консервативною, і її повна механічна енергія зберігається протягом руху.

Отже, якщо на початку часу механічна система буде відхилена від положення рівноваги і відпущена без початкової швидкості, вона буде коливатися біля цього положення рівноваги.

Потенційна енергія системи виникає внаслідок дії зовнішніх сил тяжіння та сил пружності.

Вантаж 1 здійснює вертикальний поступальний рух.

Потенційна енергія вантажу 1 дорівнюватиме роботі сили тяжіння на переміщенні тіла 1 з заданого положення у положення його рівноваги:

$$\Pi_1 = -m_1 g * \xi, \frac{\partial \Pi}{\partial \xi} = -mg. \quad (2.22)$$

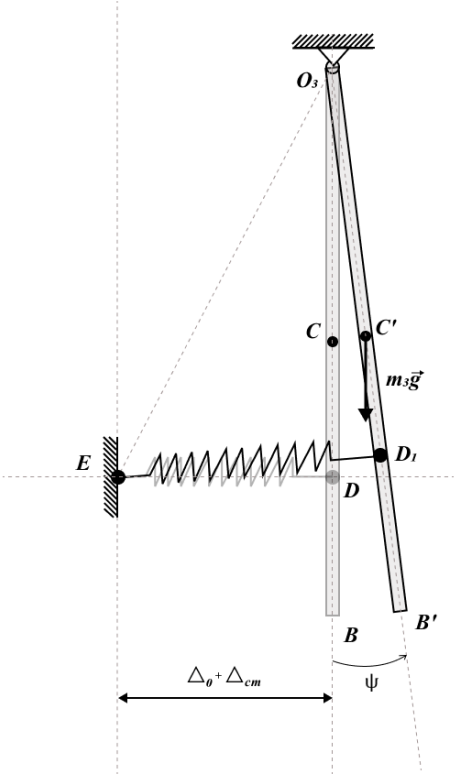


Рисунок 2.3 – Обчислення потенційної сили

Потенційна енергія, що відповідає силі тяжіння $m_3\vec{g}$, прикладеної до тіла 3, дорівнюватиме роботі цієї сили при переміщенні його центру тяжіння з положення *C'* в положення *C*, див. рис. 2.3:

$$\Pi_3 = \frac{L}{2}(1 - \cos\psi) * m_3 g, \frac{\partial \Pi_3}{\partial \xi} = \frac{L}{2} m_3 g * \sin\psi * \frac{\partial \psi}{\partial \xi}, \quad (2.23)$$

Знайдемо вираз для узагальненої сили $Q_{\text{пруж}}$, відповідній силі пружності пружини.

За визначенням $\delta A = -M_{\text{пруж}} * \delta\psi = Q_{\text{пруж}} * \delta\xi$, де $M_{\text{пруж}}$ – момент пружної сили відносно центру O_3.

Розглянемо рис. 2.4, на якому зображена схема роботи сили пружності:

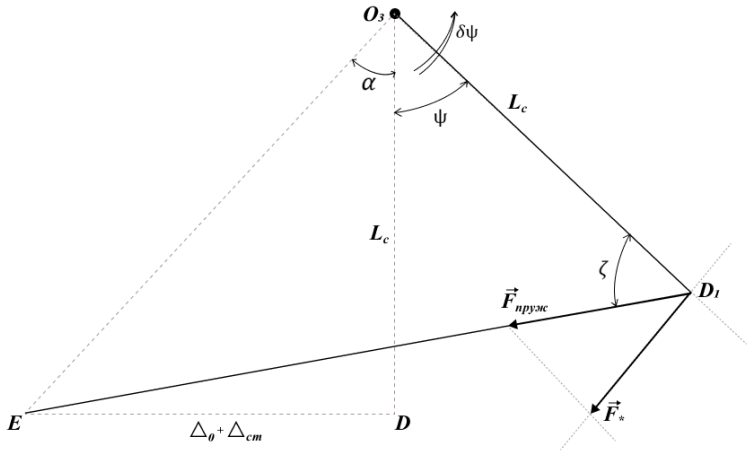


Рисунок 2.4 – Робота пружної сили

Очевидно, що $M_{\text{пруж}} = F_* * L_c = F_{\text{пруж}} * L_c * \sin\zeta$.

Використовуючи теорему «синусів» для трикутника EO_3D_1 обчислюємо величину $\sin\zeta$:

$$\frac{ED_1}{\sin(\alpha+\psi)} = \frac{EO_3}{\sin\zeta}, \tag{2.24}$$

звідки:

$$\sin\zeta = \frac{EO_3}{ED_1}\sin(\alpha+\psi) = \frac{EO_3}{ED_1} * (\sin\alpha\cos\psi + \cos\alpha\sin\psi).$$

Недеформована пружина має задану початкову довжину Δ_0. Коефіцієнт відновлення пружини заданий і дорівнює c.

Статичне подовження Δ_{cm} цієї пружини забезпечує стан рівноваги тіла 3, яке зображене на рис. 2.1. Його можна обчислити, використовуючи умову рівності нулю суми моментів сил, прикладених до цього тіла щодо центру O_3:

$$c\Delta_{cm} * L_0 = m_1 g\frac{R}{r} * L, \text{звідки } \Delta_{cm} = \frac{m_1 g}{c} * \frac{RL}{rL_c}, \tag{2.25}$$

де коефіцієнт $n_3 = L_c/L$ задається в межах: $0.6 \le n_3 \le 0.9$.

Отже, величина сили пружності, зображеної на рис. 2.4, дорівнюватиме $F_{\text{пруж}} = c * (ED_1 - \Delta_0)$.

Далі, щоб отримати вирази для довжин відрізків, що входять у наведені вище формули, треба розглянути трикутники EDO_3, та ED_1O_3.

Маємо:

$$ED = \Delta_0 + \Delta_{cm},\ O_3D = O_3D_1 = L_c,\ (O_3E)^2 = (ED)^2 + (O_3D)^2,$$
$$\cos\alpha = \frac{O_3D}{O_3E} = \frac{L_c}{O_3E},\ \sin\alpha = \frac{ED}{O_3E} = \frac{(\Delta_0 + \Delta_{cm})}{O_3E}, \tag{2.26}$$

$$(ED_1)^2 = (O_3E)^2 + (O_3D_1)^2 - 2 * O_3E * O_3D_1$$
$$* \cos(\alpha + \psi)$$
$$= (O_3E)^2 + (O_3D_1)^2 - 2 * O_3E * O_3D_1 * (\cos\alpha\cos\psi - \sin\alpha\sin\psi)$$
$$= (O_3E)^2 + (O_3D_1)^2 - 2 * O_3D_1 * (O_3D * \cos\psi - ED * \sin\psi)$$

Після підстановки вираз набуває вигляд:

$$ED_1 = \sqrt{(\Delta_0 + \Delta_{cm})^2 + 2L_c^2} - 2L_c(L_c\cos\psi - (\Delta_0 + \Delta_{cm})\sin\psi), \tag{2.27}$$

$$\sin\zeta = \frac{EO_3}{ED_1} * (\sin\alpha\cos\psi + \cos\alpha\sin\psi) = \frac{(\Delta_0+\Delta_{cm})*\cos\psi + L_c*\sin\psi}{ED_1}.$$

У результаті момент сили пружності дорівнюватиме:

$$M_{\text{пруж}} = F_{\text{пруж}} * L_c * \sin\zeta = \text{c} * \frac{(ED_1 - \Delta_0)}{ED_1} * L_c * [(\Delta_0 + \Delta_{cm}) * \cos\psi + L_c * \sin\psi].$$

А шукана узагальнена сума дорівнює

$$Q_{\text{пруж}} = -M_{\text{пруж}} * \frac{1}{r}\frac{\boldsymbol{X}}{\boldsymbol{Z}},\ \text{так як}\ \delta A = Q_{\text{пруж}} * \delta\xi.$$

І як результат маємо

$$Q_{\text{пруж}} = -M_{\text{пруж}} * \frac{1}{r}\frac{\boldsymbol{X}}{\boldsymbol{Z}} = -c * \frac{(ED_1 - \Delta_0)}{ED_1} * \frac{L_c}{R} * [(\Delta_0 + \Delta_{cm}) * \cos\psi + L_c * \sin\psi] * \frac{\boldsymbol{X}}{\boldsymbol{Z}}.$$

З іншого боку, ми знаємо, що сила пружності є потенційною, тобто

$$Q_{\text{пруж}} = -\frac{\partial\Pi_{\text{пруж}}}{\partial\xi},$$

де $\Pi_{\text{пруж}} = \frac{c}{2}[(ED_1 - \Delta_0)^2 - \Delta_{cm}^2]$ - потенційна енергія пружини.

Очевидно, що

$$-\frac{\partial\Pi_{\text{пруж}}}{\partial\xi} = -\text{c} * (ED_1 - \Delta_0)\frac{\partial ED_1}{\partial\xi}.$$

Скористаємося виразом (2.27):

$$\frac{\partial ED_1}{\partial\xi} = \frac{L_c(L_c\sin\psi + (\Delta_0+\Delta_{cm})\cos\psi)}{\sqrt{(\Delta_0+\Delta_{cm})^2 + 2L_c^2 - 2L_c(L_c\cos\psi - (\Delta_0+\Delta_{cm})\sin\psi)}} * \frac{\partial\psi}{\partial\xi},$$

та з урахуванням (2.16) отримаємо

$$\frac{\partial ED_1}{\partial\xi} = \frac{1}{ED_1}(L_c\sin\psi + (\Delta_0 + \Delta_{cm})\cos\psi) * \frac{L_c}{r} * \frac{\boldsymbol{X}}{\boldsymbol{Z}}.$$

У результаті ми отримали той самий результат для узагальненої сили $Q_{\text{пруж}}$. Що підтверджує правильність виконаних математичних розрахунків.

Тепер є можливість повністю записати вираз для правої частини рівняння Лагранжа $-\frac{\partial(\Pi_1+\Pi_3)}{\partial\xi}+Q_{\text{пруж}}$:

$$-\frac{\partial\Pi_{\text{пруж}}}{\partial\xi}=m_1-\frac{L}{2R}m_3g*\sin\psi*\frac{\boldsymbol{X}}{\boldsymbol{Z}}-c*\frac{ED_1-\Delta_0}{ED_1}*$$
$$\left((\Delta_0+\Delta_{cm})*\cos\psi-L_c*\sin\psi\right)*\frac{L_c}{R}*\frac{\boldsymbol{X}}{\boldsymbol{Z}}, \qquad (2.30)$$

У випадку малих коливань вираз (2.30) значно спрощується.

Горизонтальне зміщення DD_1, точки D в положення D_1 будемо вважати рівним $L_c\sin\psi$. Тоді потенційна енергія пружини дорівнюватиме $\Pi_{\text{пруж}}=\frac{c}{2}\left[(DD_1+\Delta_{cm})^2-{\Delta_{cm}}^2\right]$, і враховуючи співвідношення $\psi=\frac{1}{R}\frac{r}{L}\xi$ отримаємо $DD_1\approx L_c\psi=\frac{rL_c}{RL}\xi=\sigma*\xi$, і тоді

$$\Pi_{\text{пруж}}=\frac{c}{2}\left[(DD_1)^2+2DD_1*\Delta_{cm}\right]=\frac{c}{2}\left[(\sigma\xi)^2+2\sigma\xi*\Delta_{cm}\right],\ \ \frac{\partial\Pi_{\text{пруж}}}{\partial\xi}=$$
$$c[\sigma^2*\xi+\sigma*\Delta_{cm}].$$

Остаточно маємо такий вираз.:

$$-\frac{\partial\Pi_*}{\partial\xi}-\frac{\partial}{\partial\xi}\left[\Pi_1+\Pi_3+\Pi_{\text{пруж}}\right]=m_1g-\frac{m_3gL}{2R}\xi-$$
$$c*\sigma^2\xi-\sigma\Delta_{cm}=-\left(c\sigma^2+\frac{m_3gL}{2R}\right)\xi, \qquad (2.31)$$

Вираз (2.31) отримано з урахуванням потенційних енергій сил тяжіння для тіл 1 та 3, а також використовуючи умову рівноваги $m_1g=c*\sigma\Delta_{cm}$. Ця умова випливає із співвідношення (2.25).

В результаті отримаємо рівняння лінійних коливань наступного вигляду:

$$\frac{1}{R^2}\left(J_1+J_2+J_3\frac{r^2}{L^2}\right)\ddot{\xi}+\left(c\sigma^2+\frac{m_3gL}{2R}\right)*\xi$$
$$\text{або }\ddot{\xi}+\omega_*^2*\xi=0,\omega_*^2=\frac{\left(c\sigma^2+\frac{m_3gL}{2R}\right)R^2}{J_1+J_2+\frac{r^2}{L^2}J_3}, \qquad (2.32)$$

де ω_* - кругова частота малих коливань розглянутої системи.

Закон малих коливань напишемо у такій формі:

$$\xi_*(t) = \xi_0 * \cos(\omega_* t)\,, \tag{2.33}$$

З початковими умовами руху $\xi(0) = \xi_0$ і $\dot{\xi}(0) = 0.$ (2.34)

Таким чином для реалізації чисельного методу Рунге-Кутта було отримано рівняння Лагранжа 2-го роду, яке описує нелінійні коливання системи наступного вигляду:

$$\frac{J_1 + J_2 + J_3 * \left[\frac{\boldsymbol{X}}{\boldsymbol{Z}}\right]^2}{R^2} * \ddot{\xi} = \frac{J_3}{R^3}\frac{\boldsymbol{X}}{\boldsymbol{Z^3}} *$$
$$\left[\boldsymbol{X} * \left(\frac{\partial \boldsymbol{Z}}{\partial \varphi} + \frac{\boldsymbol{X}}{\boldsymbol{Z}} * \frac{\partial \boldsymbol{Z}}{\partial \psi}\right) - \left(\frac{\partial \boldsymbol{X}}{\partial \varphi} + \frac{\boldsymbol{X}}{\boldsymbol{Z}} * \frac{\partial \boldsymbol{X}}{\partial \psi}\right) * \boldsymbol{Z}\right] * \ddot{\xi}^2 + m_1 g - \frac{m_3 g L}{2R} * \sin\psi * \frac{\boldsymbol{X}}{\boldsymbol{Z}} -$$
$$c * \frac{ED_1 - \Delta_0}{ED_1} * \left((\Delta_0 + \Delta_{cm}) * \cos\psi + L_c * \sin\psi\right) * \frac{L_c}{R} * \frac{\boldsymbol{X}}{\boldsymbol{Z}}, \tag{2.35}$$

Особливість такого диференціального рівняння (2.35) полягає насамперед в тому, що воно містить вирази, де узагальнена координата ξ входить через неявні залежності, які базуються на представленнях виду (2.2), (2.6) та (2.17) - (2.19).

Така ситуація потребує особливого підходу для реалізації обчислень за класичною схемою чисельного методу Рунге-Кутта, яка наведена нижче:

$$k_1 = h * f\left(t_j; \xi_j; \dot{\xi}_j\right),$$
$$k_2 = h * f\left(t_j + \frac{h}{2}; \xi_j + \frac{h}{2}\dot{\xi}_j + \frac{h}{8}k_1; \dot{\xi}_j + \frac{h}{2}k_1\right),$$
$$k_3 = h * f\left(t_j + \frac{h}{2}; \xi_j + \frac{h}{2}\dot{\xi}_j + \frac{h}{8}k_1; \dot{\xi}_j + \frac{h}{2}k_2\right), \tag{2.36}$$
$$k_4 = h * f\left(t_j + h; \xi_j + h\dot{\xi}_j + \frac{h}{2}k_3; \dot{\xi}_j + k_3\right),$$
$$\dot{\xi}_{j+1} = \dot{\xi}_j + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}, \xi_{j+1} = \xi_j + h * \left(\dot{\xi}_j + \frac{k_1 + k_2 + k_3}{6}\right),$$

де $h = t_{j+1} - t_j$ - крок інтегрування незалежної тимчасової змінної t.

Тут $f\left(t; \xi; \dot{\xi}\right)$ – функція, яка обчислює значення правої частини інтегрованого диференціального рівняння (2.35), що записане у вигляді $\ddot{\xi} = f\left(t; \xi; \dot{\xi}\right)$. Початковими умовами для рівняння (2.35) тут виступають умови (2.34).

2.3 Встановлення допустимих діапазонів характеристик та початкових умов для механічної системи

Задана механічна система знаходиться під дією тільки консервативних сил і у положенні стійкої рівноваги, відповідному мінімальному значенню потенційної енергії. Тому початкові умови (2.34) свідчать про те, що узагальнена координата ξ буде обмеженою функцією часу і $-\xi_0 \leq \xi(t) \leq \xi_0$.

Очевидно при такій постановці задачі, значення ξ_0 не може бути довільним. Це так, тому що передбачуваний рух механічної системи має бути коливанням біля її положення рівноваги.

Для проведення чисельного експерименту слід встановити діапазони допустимих значень для основних характеристик механічної системи, що розглядається. В роботі задіяна Таблиця 2.1:

Таблиця 2.1

Маси тіл (*кг*)	Розміри тіл (*м*)	Параметри пружини
$2.00 \leq m_1 \leq 7.00$	$0.25 \leq r \leq 0.75$	$100.0 \leq c \leq 500.0$ (*Н/м*)
$1.00 \leq m_2 \leq 3.00$	$1.00 \leq R \leq 2.00$	$1.00 \leq \Delta_0 \leq 2.00$ (*м*)
$3.00 \leq m_3 \leq 9.00$	$2.00 \leq L \leq 5.00$	$0.60 \leq (L_0/L) \leq 0.95$, n3 = L_0/L
	$3.00 \leq H \leq 9.00$	

Ще одне обмеження, яке ми накладаємо, є статична деформація Δ_{cm} пружини, яка забезпечує становище спокою системи в заданій конфігурації

Під час проведення чисельного експерименту будемо вимагати, щоб статична деформація не перевищувала 15% недеформованої довжини Δ_0, самої пружини.

Після усіх розрахунків отримана формула для статичної деформації має такий вигляд: $\Delta_{cm} = \frac{m_1 g}{c} * \frac{R}{r} * \frac{L_0}{L}$ і для задоволення умови

$$\Delta_{cm} \leq \frac{15}{100}\Delta_0, \tag{2.37}$$

ми можемо змінювати значення різних параметрів за власним бажанням.

Вважати якусь із них переважною без обґрунтування не є належним. Наведене вище говорить про необхідність проведення попередніх обчислень і надання користувачеві інформації щодо виконання зазначених умов під час процесу вибору початкових значень параметрів системи.

2.4 Аналіз динамічної поведінки механічної системи

Отримане рівняння руху є ключовим результатом аналізу даної механічної системи. Воно дозволяє математично описати динаміку системи з урахуванням всіх фізичних параметрів та взаємодій між її складовими частинами.

Основні висновки з отриманого рівняння руху:

а) Включення потенційної енергії та пружних сил в рівняння руху дозволяє оцінювати, як зміщення однієї частини системи впливає на інші, що важливо для розуміння динамічної відповіді системи.

б) Присутність пружних сил у рівнянні вказує на можливість коливань у системі. Аналізуючи узагальнені сили, можна визначити стійкість системи — чи схильна вона до стабільного руху, або схильна до коливань та реакцій на малі збурення.

в) Рівняння руху залежить від параметрів системи, таких як маси, довжини і радіуси складових частин, а також константи пружності. Це дозволяє аналізувати, як зміна цих параметрів впливає на рух та стан системи в цілому.

Аналіз динамічної поведінки механічної системи базується на математичному моделюванні через рівняння руху, яке враховує всі фізичні взаємодії та параметри системи, що дозволяє зрозуміти її реакцію на різноманітні умови та впливи.

Отже дана механічна система, яка перебуває під дією консервативних сил і знаходиться у стані рівноваги з мінімальною потенційною енергією, виявляє дві граничні конфігурації.

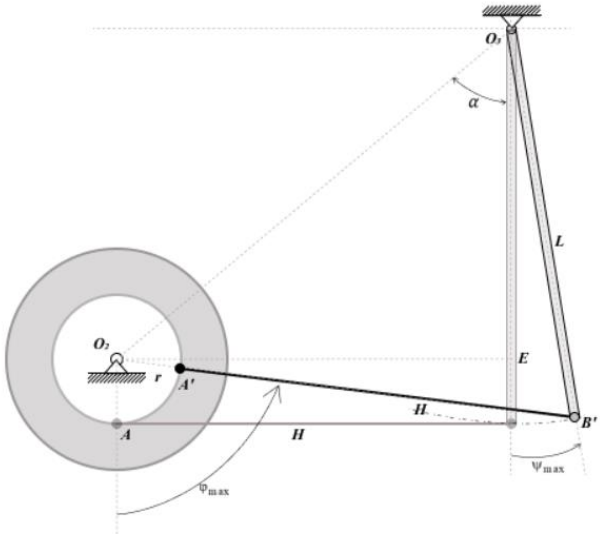


Рисунок 2.5 – перша гранична конфігурація системи

Перша конфігурація відповідає максимальному відхиленню стрижня 3 від початкового положення на кут Ψ_{max} (див. рис. 2.5), при якому точка B досягає максимального віддалення по горизонталі від положення рівноваги. За теоремою косинусів для трикутника, визначається максимально допустимий кут Ψ_{max} (2.38).

$$\psi_0 < \psi_{max} = \arccos\left[\frac{L^2 - R * (L + H)}{L * \sqrt{H^2 + (L - R)^2}}\right] - \arccos\left[\frac{L - R}{\sqrt{H^2 + (L - R)^2}}\right], \qquad (2.38)$$

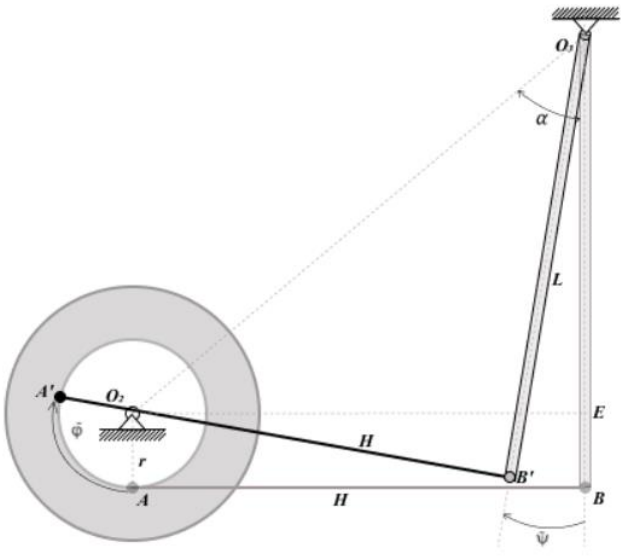


Рисунок 2.6 – Друга гранична конфігурація системи

Друга гранична конфігурація показує мінімально можливий кут $\tilde{\psi} < 0$, при якому тіло 3 знаходиться у положенні рівноваги. Візуальний аналіз підтверджує (див. рис. 2.6), що мінімально можливий кут φ у другій конфігурації є більшим за максимально допустимий кут ψ з першої конфігурації. Таким чином, для визначення початкових умов системи (2.34), обранням конкретних значень та умови $|\psi_0| < \psi_{max}$ ураховуються обмеження на кути φ та ψ для тіл 2 і 3 відповідно.

Аналіз діапазонів кутів φ та ψ показує взаємозв'язок між ними в межах допустимих значень, як показано на рис. 2.7.

при $0 < \varphi < \varphi_*$ буде $0 < \psi < \psi_{max}$; при $\varphi_* < \varphi < 2\pi$ буде $\tilde{\psi} < \psi < 0$.

Для чисельного експерименту важливо встановити допустимі діапазони основних характеристик механічної системи, щоб забезпечити коректність розрахунків та відповідність заданим умовам.

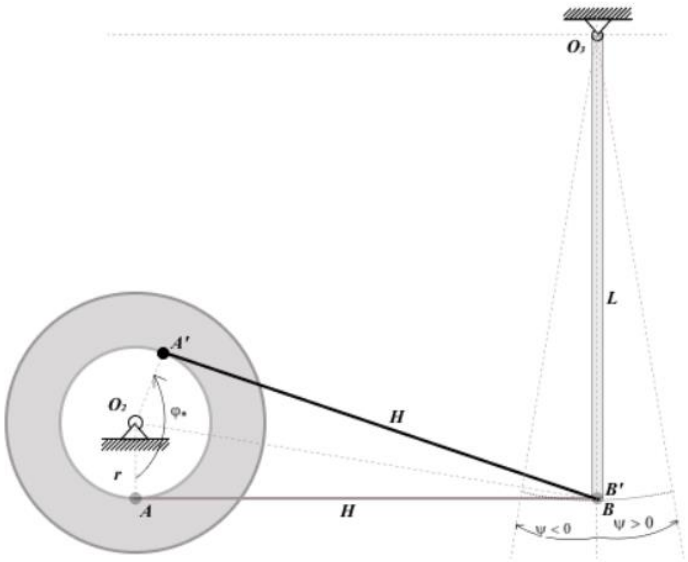


Рисунок 2.7 – Проходження тіла 3 через вертикаль

3 ОГЛЯД ТЕХНОЛОГІЇ WPF (WINDOWS PRESENTETION FOUNDATION)

Windows Presentation Foundation (WPF) - це спеціальна технологія, що надає можливість створення користувацьких інтерфейсів для застосунків Windows, а також є складовою частиною платформи .NET. Ця технологія була розроблена як сучасна альтернатива вже існуючій технології Windows Forms.

WPF є високорівневою об'єктно-орієнтованою платформою, що дозволяє створювати як двовимірні, так і тривимірні інтерфейси користувача.

Для роботи зручна у використанні Visual Studio, яка підтримує як програмування, так і ручне редагування XAML. Розширювана мова розмітки застосунків (eXtensible Application Markup Language, XAML) являється XML-подібною мовою і використовується для представлення дерев об'єктів .NET. Дані XAML перетворюються на дерево об'єктів за допомогою аналізатора XAML. Основним призначенням XAML є опис інтерфейсів користувача в додатках WPF.

Для створення програми на WPF була використана мова програмування C#, яка сумісна з платформою .NET. C# є однією з основних мов програмування, яку використовують для розробки застосунків на платформі WPF. Вона є мовою програмування .NET, яка надає зручний та потужний інструментарій для створення інтерфейсів користувача, включаючи графічні та анімаційні ефекти, взаємодію з даними та реалізацію бізнес-логіки застосунків. Крім того, C# має широку спільноту розробників та багато ресурсів для самостійного вивчення, що робить її популярним вибором для розробників, що працюють з WPF.

3.1 Основи та можливості технології WPF

Основою WPF є векторна система візуалізації, яка не залежить від роздільної здатності пристрою виведення. Вона була розроблена з урахуванням можливостей сучасного графічного обладнання. WPF надає інструменти для створення візуального інтерфейсу, вмістивши в собі мову

XAML, елементи управління, прив'язку даних, двовимірну та тривимірну графіку, стилі, шаблони, анімацію, макети, документи, текст, мультимедіа та оформлення.

Технологія WPF пропонує широкий спектр можливостей для створення графічних інтерфейсів механічних систем, в яких можна використовувати різні елементи управління, розробляти привабливий зовнішній вигляд, створювати анімації, а також реагувати на користувацькі події.

WPF надає розширений і гнучкий набір графічних властивостей. Основною одиницею вимірювання є апаратно-незалежні пікселі, які дозволяють створювати графіку, яка не залежить від роздільної здатності дисплея та пристрою. WPF має вбудовану підтримку графіки та анімації для спрощення програмування та автоматичного управління анімацією. Крім того, графічна система WPF використовує апаратне прискорення, яке знижує навантаження на ЦП за рахунок використання функціональності графічного обладнання.

Також WPF забезпечує гнучку систему макета, яка динамічно розміщує елементи керування на основі їх безпосереднього вмісту, а не фіксованих координат. Це дозволяє створювати інтерфейс, який адаптується до відображення динамічного вмісту та підтримки різних мов програмування.

WPF пропонує широкий спектр можливостей для створення інтерактивних застосунків. Прив'язка даних є гнучким механізмом, що може дозволити пов'язувати різноманітні дані, починаючи від значень властивостей елементів управління, закінчуючи загальнодоступними властивостями, що реалізують поля бази даних через Entity Framework, через розширення розмітки XAML.

Технологія WPF розділена на два рівні: керований API (managed API) і некерований API (unmanaged API). Керований API містить у собі код, який виконується під контролем середовища виконання .NET- Common Language Runtime. Цей API описує основні функціональні можливості платформи WPF.

Схематично архітектуру WPF показано на рис. 3.1.

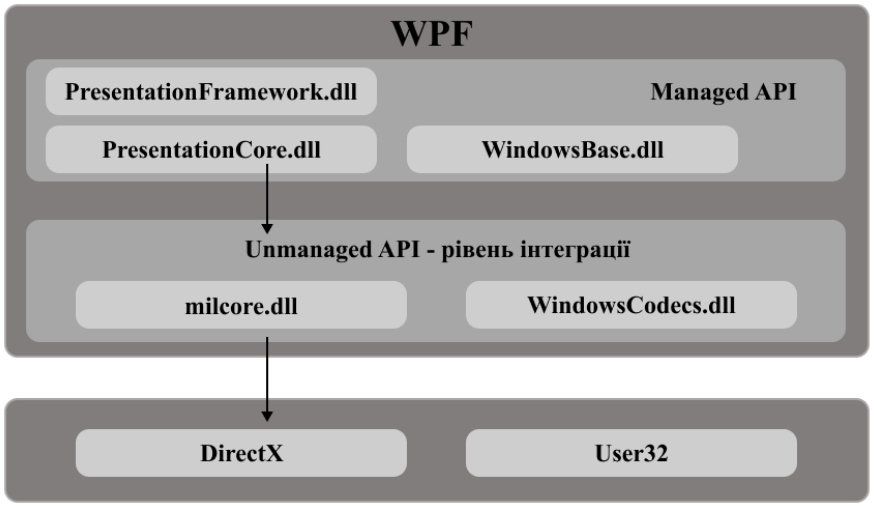


Рис. 3.1 – Схема архітектури WPF

Загалом, технологія WPF відкриває безліч можливостей для розробників, спрощуючи процес створення графічних інтерфейсів та забезпечуючи високу продуктивність та зручність в роботі. Перша версія WPF 3.0 вийшла разом із .NET 3.0 і ОС Windows Vista в 2006 році і з тих пір успішно розвивається, залишаючись однією з популярних платформ для розробки застосунків, завдяки своїм потужним можливостям та надійності.

3.2 Переваги використання WPF для візуалізації механічних процесів

Візуалізація механичних процесів є важливим аспектом в наукових дослідженнях, інженерії, освіті та промисловості. Використання технології Windows Presentation Foundation для цієї мети має численні переваги. Якщо порівнювати із Windows Forms, то основною перевагою WPF є використання графічної технології DirectX, на відміну від GDI/GDI+. За допомогою використання апаратного прискорення через DirectX, WPF досягає набагато вищої продуктивності у порівнянні з GDI+. При створенні додатку в WPF, він не буде залежати від розширення екрану, через те що автоматично використовуються компоненти DirectX. Роздільна здатність у WPF завжди однакова для будь-якого розширення екрану.

Також не менш важливим аспектом є гнучкість і адаптивність інтерфейсу. За допомогою WPF можна легко налаштувати візуальні елементи

для відображення різних механічних процесів. Це дозволяє створювати інтерфейси, які відповідають конкретним вимогам та вимогам проекту.

WPF пропонує безліч графічних функцій, включаючи векторну графіку, 3D-моделювання, анімацію та трансформацію. Це дозволяє візуалізувати складні механічні процеси у вигляді візуального моделювання з високою точністю і реалістичністю.

Технологія WPF дозволяє створювати інтерактивні інтерфейси, які дозволяють користувачеві взаємодіяти з візуалізованими механічними процесами. Одним з прикладів є те, що користувач може змінювати параметри моделювання в режимі реального часу та спостерігати зміни в результатах.

Плюсом технології WPF є легка інтеграція з іншими технологіями Microsoft, такими як .NET Framework та Visual Studio, що значно спрощує розробку та супровід моделей.

WPF підтримує гнучку потокову передачу, розміщуючи усі елементи керування на основі їх контексту, а не блокуючи їх у певних координатах. В результаті виходить користувальницький інтерфейс, який можна адаптувати задля відображення дуже динамічного вмісту.

Використовуючи технологію WPF для візуалізації механичних процесів, можна створювати потужні та ефективні інструменти для досліджень, навчання та промислового застосування. Її гнучкість, графічні можливості та інтерактивність роблять її ідеальним вибором для розробки візуалізацій складних фізичних процесів.

3.3 Інструменти та компоненти WPF для моделювання коливань

Windows Presentation Foundation надає розробникам безліч інструментів і компонентів для створення ефективних візуальних моделей коливальних процесів.

Одним з основних інструментів WPF для моделювання коливань є XAML (Extensible Application Markup Language). XAML використовується для

декларативного опису інтерфейсу користувача. Розробники можуть створювати різноманітні макети, стилі та шаблони рендерингу для моделі, а також створювати динамічні візуалізації коливальних процесів.

Code-behind файли реалізують саму логіку моделювання коливань, а саме розрахунки параметрів коливань, взаємодії з користувачем і оновлення графіки в режимі реального часу. У Code-behind файлах також реалізуються обробники подій для інтерактивності, такі як використання елементів керування для зміни параметрів моделі.

Одним із ключових і важливих компонентів WPF для моделювання коливань є контейнер для вільного розміщення графічних елементів Canvas. Він ідеально підходить для відтворення візуалізації коливань та траєкторій руху об'єктів, дозволяючи легко і швидко створювати та переміщувати елементи, які імітують коливання. Path і Geometry використовуються для створення складних графічних форм і траєкторій. Вони можуть використовуватись для візуалізації траєкторій коливань, а Path можна анімувати для створення динамічних візуалізацій коливальних процесів.

Розкадровка Storyboard використовується для створення анімацій у WPF та керування ними. Це дозволяє анімувати зміни властивостей об'єктів, таких як положення, розмір та колір, які важливі для моделювання динаміки коливань. Storyboard дозволяє створювати складні послідовності анімацій у різні періоди часу, що дозволяє моделювати складні коливальні процеси.

Ще одним не менш важливим інструментом є LiveCharts. Це бібліотека, що допомагає у створенні графіків та діаграм, яка підтримує інтерактивність та анімації. Вона дозволяє легко створювати графіки зміни параметрів коливань у реальному часі.

Важливим аспектом також є можливість інтеграції WPF з деякими іншими технологіями для більш розширеного функціоналу моделювання коливань. Одним із прикладів у WPF є взаємодія з бібліотеками для наукових обчислень, такими як Math.NET, для виконання складних математичних розрахунків.

Загалом, інструменти та компоненти WPF забезпечують потужні можливості для моделювання та візуалізації коливальних процесів, надаючи розробникові гнучкі та ефективні засоби для створення інтерактивних та наочних симуляцій.

4 РОЗРОБКА ПРОГРАМНОЇ МОДЕЛІ КОЛИВАНЬ МЕХАНІЧНОЇ СИСТЕМИ

4.1 Числове моделювання та візуалізація коливань із використанням бібліотеки LiveCharts

LiveCharts.Wpf — це бібліотека для побудови графіків і діаграм у середовищі WPF (Windows Presentation Foundation). Ця бібліотека дозволяє легко створювати різноманітні графіки та інтерактивні візуалізації даних:

а) Лінійні графіки (Line Chart) - Графік, де дані відображаються лініями, які з'єднують точки, відповідно до значень по осі X та Y.

б) Стовпчикові діаграми (Column Chart) - Діаграма, де дані представлені у вигляді стовпчиків, що відображають величини на осі Y, з розділенням по осі X.

в) Кругові діаграми (Pie Chart) - Діаграма, що показує частки або відсотки від загальної суми. Кожна категорія представлена сектором, пропорційним її значенню.

г) Площинні графіки (Area Chart) - Графік, де область під кривою заповнюється кольором, що дозволяє візуально виділити область між кривими та осію X або Y.

д) Гістограми (Histogram) - Графік, де вісі асоційовані з інтервалами значень, а стовпці демонструють частоту входження в кожен інтервал.

е) Графіки з точками (Scatter Plot) - Графік, де точки розташовані у просторі відповідно до значень двох змінних, що дозволяє візуалізувати залежність між ними.

Ці типи графіків можна легко інтегрувати в WPF додатки за допомогою LiveCharts.Wpf, що забезпечує зручний і потужний інструментарій для візуалізації даних.

2. Бібліотека LiveCharts.Wpf підтримує анімації, що дозволяє зробити візуалізацію графіків більш динамічною та привабливою для користувачів. Основні можливості анімацій включають:

а) **Анімація зміни даних**: При зміні даних на графіку (наприклад, додавання нових точок або зміна значень), LiveCharts.Wpf здатна плавно оновлювати відображення, використовуючи анімаційні ефекти.

б) **Анімація переходів**: При зміні типу графіка або параметрів візуалізації (наприклад, зміна типу діаграми з лінійної на стовпчикову), бібліотека підтримує анімаційні ефекти для плавних переходів.

в) **Анімація маркерів та підписів**: LiveCharts.Wpf дозволяє анімувати маркери точок на графіку або підписи до вісей, що поліпшує візуальне сприйняття та взаємодію з графіком.

г) **Анімація інтерактивних елементів**: Деякі типи графіків підтримують анімаційні ефекти під час взаємодії користувача з графіком (наприклад, при наведенні курсора на точки на графіку або при виборі діапазону).

Ці анімації додають естетичну привабливість та покращують користувацький досвід при роботі з візуалізаціями даних у WPF додатках, роблячи їх більш динамічними та зрозумілими.

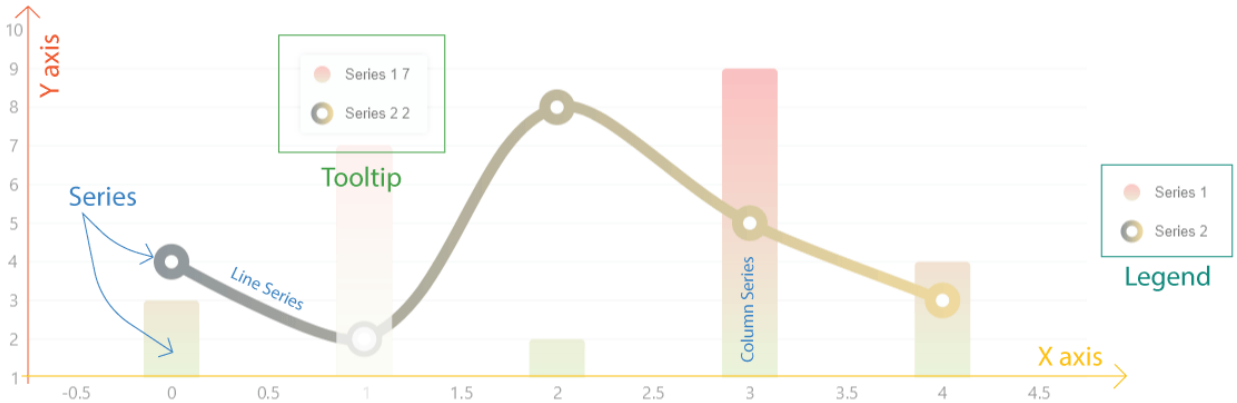
3. Користувачі можуть взаємодіяти з графіками, наприклад, за допомогою підсвічування певних областей, масштабування або перетягування. Ці функції роблять графіки більш інтерактивними та зручними для використання, що сприяє зручнішому аналізу даних та взаємодії з ними у додатках WPF.

4. LiveCharts.Wpf надає гнучкі можливості для настройки вигляду графіків, що дозволяє створювати індивідуальні візуалізації відповідно потреб. Можна налаштувати кольори ліній, точок, стовпчиків та інших елементів графіку. Також є можливість змінювати стилі ліній (товщину, тип штриху) і маркерів (форма, розмір, колір). Це дозволяє надати графікам більш різноманітний і професійний вигляд. Крім основних параметрів, ви також можете налаштовувати такі елементи, як товщина рамки графіку, розмір вікна для візуалізації, наявність легенди та її вигляд.

З метою візуалізації числового моделювання була вивчена основна документація бібліотеки LiveCharts.Wpf та успішно налаштована для роботи і використання у застосунку. Для роботи було обрано елемент керування CartesianChart.

Елемент керування CartesianChart – це "готовий до використання" елемент керування для створення графіків з використанням декартової системи координат. Щоб розпочати роботу необхідно присвоїти властивості Series колекцію ICartesianSeries.

Основними компонентами цього контролу є Series, Axes, Tooltip, Legend



CartesianChart має 2 осі, властивості YAxis і XAxis, а вісь IAxis, окрім візуального налаштування, такого як формат міток і кольори розділювачів, також керує багатьма важливими функціями, такими як масштабування, панорамування, шкала (журнал) і перегортання сторінок.

4.2 Програмна реалізація математичної моделі за допомогою WPF

WPF був обраний для реалізації цього проекту через його потужні можливості для створення інтерфейсів користувача та візуалізації даних. Основними інструментами, які використовувались під час розробки, були:

а) Visual Studio – інтегроване середовище розробки (IDE), яке забезпечує підтримку для WPF та інших технологій .NET.

б) C# – основна мова програмування для реалізації бізнес-логіки та взаємодії з користувачем.

в) XAML – мова розмітки, яка використовується для створення користувацького інтерфейсу в WPF.

Для реалізації математичної моделі за допомогою WPF було встановлено бібліотеку LiveCharts.Wpf через NuGet Package Manager у Visual Studio. Сама математична модель була створена у вигляді сукупності методів, які обчислюють результати, що потрібно візуалізувати. Ці данні описують динаміку заданої системи тіл.

Візуалізація є ключовим компонентом програми. Було реалізовано 2 способи візуалізації: графіки та анімація.

Відповідно до задачі було створено Wpf-застосунок, який виконує певні розрахунки і таким чином визначає кінематичні характеристики руху тіл заданої механічної системи.

Програмна реалізація математичної моделі дозволила ефективно поєднати теоретичні аспекти механіки з практичними інструментами чисельного аналізу. Розроблений застосунок надає користувачам можливість проводити чисельні експерименти та візуалізувати результати, що значно полегшує розуміння поведінки механічної системи.

5 РЕАЛІЗАЦІЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ ДЛЯ ВІЗУАЛІЗАЦІЇ КОЛИВАНЬ

5.1 Проектування користувацького інтерфейсу

Під час розробки користувацького застосунку основний функціонал було розташовано на головному вікні. Інтерфейс головного вікна складається з області, яка відображає роботу механічної системи та демонструє графіки коливань, зліва вікна розташована панель параметрів механічної системи. На початковому етапі роботи додатку тут відображаються початкові задані параметри системи. Використовуючи відповідну кнопку, користувач має можливість змінювати ці початкові умови. Серед параметрів, які можна налаштувати, знаходяться розміри тіл 1, 2 та 3, їхні маси, а також характеристики пружини, довжина недеформованої пружини Δ_0, деформація пружини у стані спокою системи Δ_{cm}, жорсткість пружини c та коефіцієнт нелінійності пружини n_3. Ця панель забезпечує гнучкість та точність у налаштуванні початкових параметрів, що є важливим для проведення подальших розрахунків та моделювання механічних процесів. На етапі проектування головне вікно має такий вигляд (див. рис. 5.1):

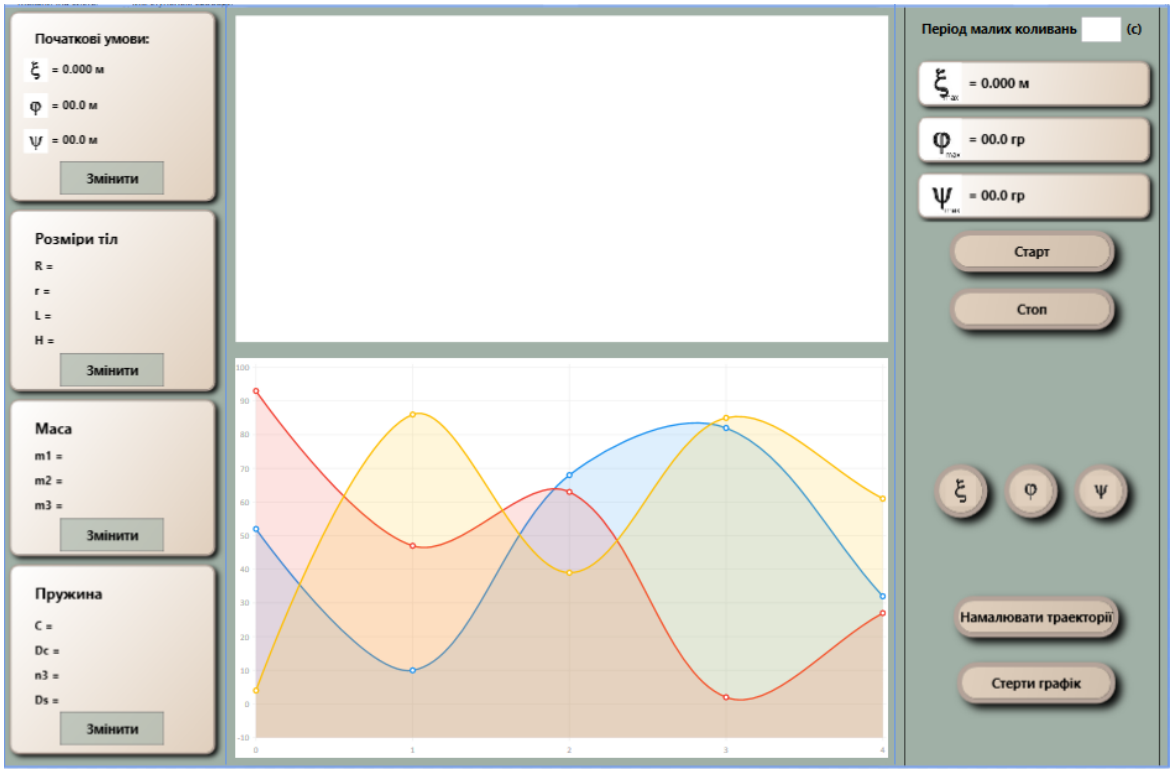


Рисунок 5.1 – Головне вікно застосунку

В лівій частині розміщені параметри заданої механічної системи з можливістю змінювати їх значення. На кожній з панелей розміщено кнопку «Змінити». Натискання по цій кнопці призводить до ініціалізації додаткових вікон, призначених для редагування значень відповідних параметрів. (див. Рисунок. 5.2).

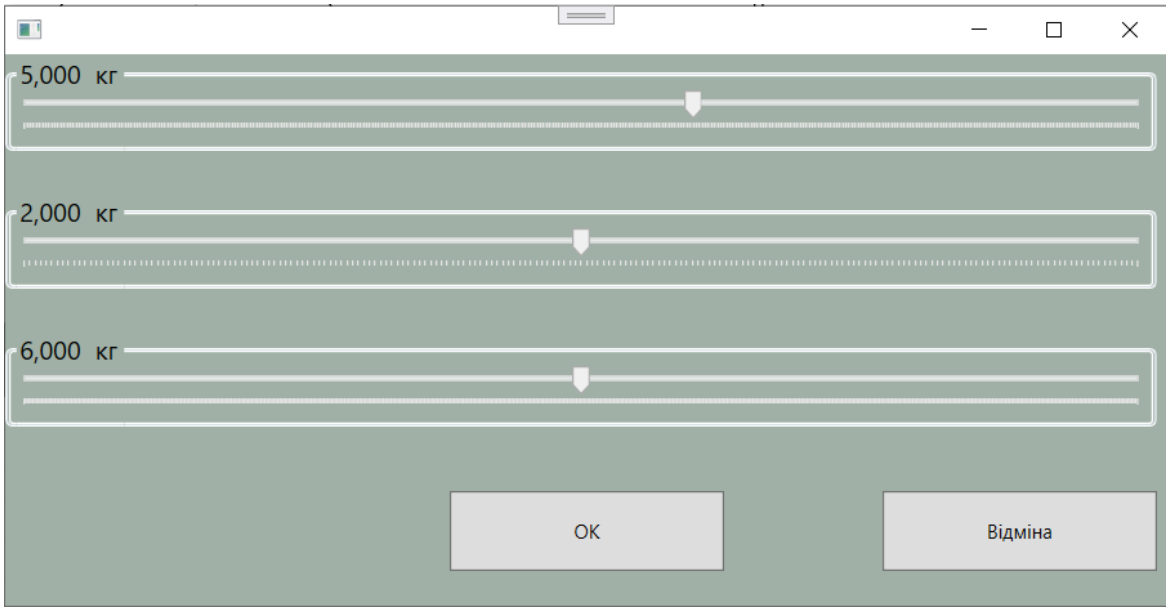


Рисунок 5.2 – Вікно редагування маси тіл

У правій частині передбачено функціонал, який демонструє основну роботу застосунку. А саме реалізацію чисельного експерименту. Натиснувши на кнопку «Намалювати траєкторії» виконується розрахунок за схемою Рунге-Кутта. Після цього можна переглянути графіки коливань, обираючи кнопки з позначенням узагальненої координати або кутів обертання тіл 2 та 3.

На рис. 5.3 можна побачити результат послідовного виконання цих дій. Для візуалізації графіків було створено спеціальний метод Graph(). Цей метод Graph() дозволяє побудувати три різних лінійних графіки на одному елементі CartesianChart, використовуючи бібліотеку LiveCharts.Wpf.

Внесені зміни автоматично відображаються на StackPanel, що дозволяє користувачу візуально контролювати та коригувати параметри в режимі реального часу. Такий інтерфейс є інтуїтивно зрозумілим і зручним для користувача, забезпечуючи точне та швидке налаштування системи.

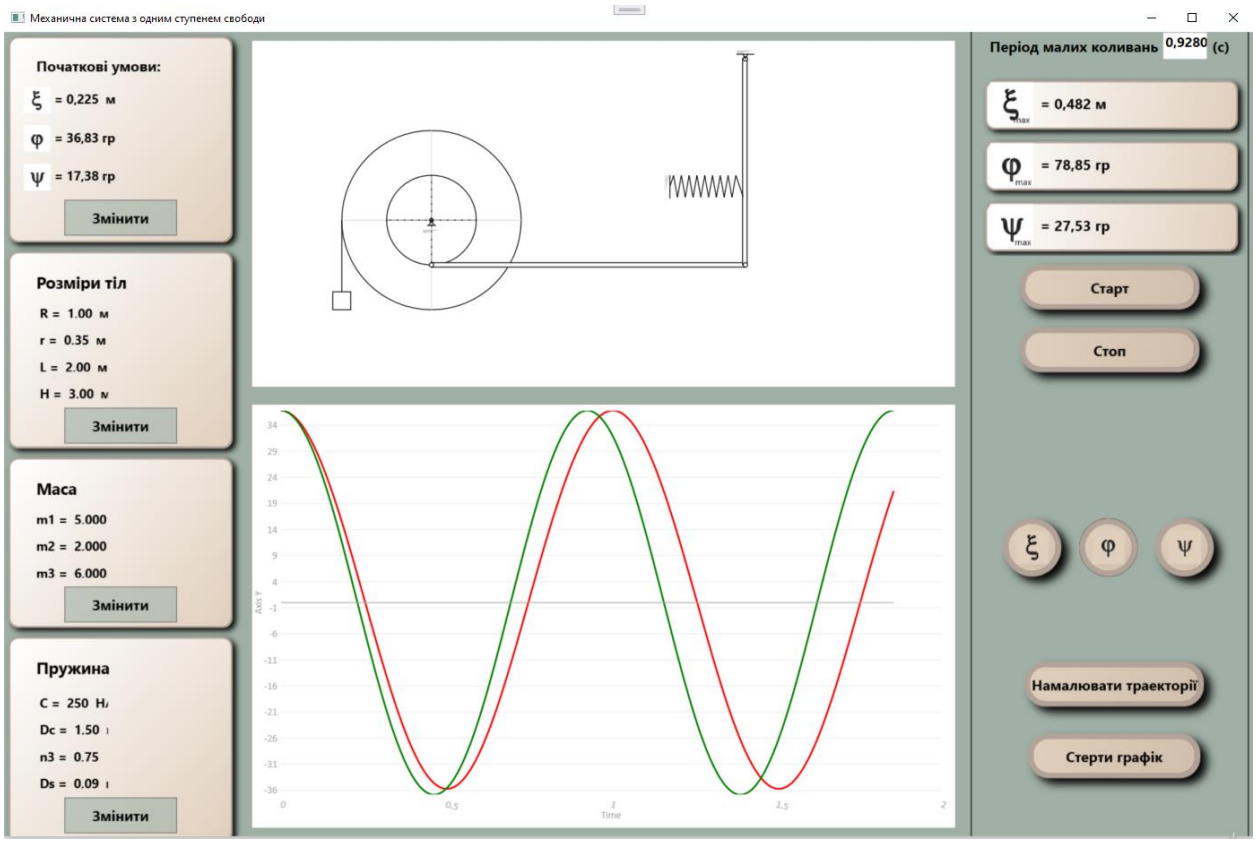


Рисунок 5.3 – Візуалізація графіків

По центру зверху розташовано Canvas, у якому за допомогою анімації відображатиметься зображення механічної системи. Canvas слугує основним візуальним елементом для демонстрації динаміки системи, забезпечуючи наочне представлення змін параметрів та їх впливу на поведінку механічних тіл. А знизу, у другій частині вікна, розташована графіка закону руху. Ця область призначена для відображення графічної інтерпретації закону руху механічної системи.

На самому верху праворуч розташована інформаційна панель, яка відображає результати обчислень після прорисовки траєкторії. На цій панелі відображаються такі параметри, як максимальні кути ξ, φ, ψ та період малих коливань. Ця інформація є ключовою для аналізу динамічних властивостей системи і дозволяє користувачу оцінити ефективність та коректність заданих параметрів, а також зробити висновки щодо подальших налаштувань та оптимізації системи.

Під цією панеллю розташовані кнопки «Старт» і «Стоп». Ці кнопки забезпечують управління запуском та зупинкою анімації механічної системи. Кнопка «Старт» активує анімацію, дозволяючи спостерігати за динамікою системи в реальному часі, тоді як кнопка «Стоп» зупиняє анімацію, що дає змогу аналізувати поточний стан системи та вносити необхідні коригування.

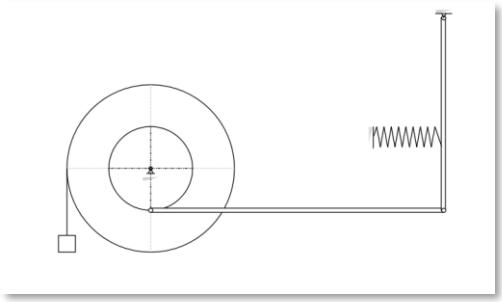


Рисунок 5.4 – Анімація механічної системи

Окремо розташовані кнопки ToggleButton дозволяють користувачеві вибирати між узагальненою координатою, кутом φ, кутом ψ. Після натискання на будь-яку з кнопок ToggleButton в CartesianChart відображається графік закону руху відповідно до обраної залежності.

Рисунок 5.5 – Кнопки для прорисовки графіків

При натисканні кнопки «Намалювати траєкторії» на графіку відображаються результати, що стосуються коливань системи. А також, додатково наявна кнопка «Стерти графік», яка дозволяє очистити графік від попередніх результатів. Це дає користувачеві можливість легко очистити графічну область і підготувати її для відображення нових результатів без необхідності закривати вікно чи перезапускати програму.

5.2 Оптимізація інтерфейсу за допомогою стилів та шаблонів у WPF

Найвідомішою особливістю WPF є можливість надавати будь-якому елементу користувальницького інтерфейсу радикально нового зовнішнього вигляду, не змінюючи його поведінку.

Стилі в WPF представляють собою простий механізм відділення значень властивостей від елементів користувацького інтерфейсу, подібно до того, як CSS відокремлює стилі від HTML. Це дозволяє зручно та ефективно управляти зовнішнім виглядом інтерфейсу, спрощуючи процес його налаштування. Можна не турбуватися про те, що стиль буде застосовано до елемента, у якого немає якихось властивостей залежності, визначених у стилі; вони будуть просто проігноровані.

Шаблони у WPF – це потужні об'єкти, які надають розробникам можливість гнучко змінювати зовнішній вигляд елементів інтерфейсу. Вони є основним інструментом для модифікації вигляду компонентів у WPF, забезпечуючи високий рівень кастомізації та відповідності дизайну вимогам користувача. Шаблони — це не лише механізм для втілення сторонніх розширень; стандартний зовнішній вигляд всіх елементів управління WPF визначається в шаблонах (і налаштовано для кожної теми Windows). Вихідний код елементів управління повністю відокремлений від подання візуального дерева (або "візуального вихідного коду").

Наявність шаблонів і бажання відокремити зовнішній вигляд від логіки — це причини того, що елементи управління WPF містять не так багато простих властивостей для налаштування їхнього вигляду.

У рамках цієї роботи було розроблено декілька стилів та шаблонів для покращення зовнішнього вигляду та взаємодії елементів інтерфейсу (див. додаток А).

5.3 Візуалізація результатів моделювання

Візуалізація є важливим аспектом будь-якої науково-дослідної роботи, оскільки вона дозволяє графічно представити результати моделювання і

аналізу, роблячи їх більш зрозумілими та доступними для сприйняття. Візуалізація математичної моделі механічної системи була розроблена з використанням технології Windows Presentation Foundation та бібліотеки LiveCharts.

Механічні системи з одним ступенем свободи, які моделюються в роботі, відображають складну динаміку, яка може бути складною для візуального аналізу без відповідних інструментів. З метою вивчення руху системи, виявлення характерних особливостей і порівняння різних варіантів вхідних параметрів було створено графічні представлення результатів моделювання механічної системи.

Підхід базується на використанні класу VisualHost, який надає можливість додавання власних візуальних елементів у WPF застосунки. Цей клас відокремлює візуальну частину системи від її логічного функціоналу, що дозволяє ефективно інтегрувати графічні представлення безпосередньо з результатами моделювання. Додатково, для реалізації інтерактивних і динамічних графіків використано бібліотеку LiveCharts.

Клас VisualHost, що наслідується від FrameworkElement, призначений для створення контейнера для візуальних елементів в WPF. Основна ідея полягає в тому, щоб мати можливість додавати власні візуальні елементи до цього контейнера і мати можливість керувати ними у візуальній структурі.

Для створення графічного зображення у WPF використовується DrawingContext. За допомогою DrawingContext можна малювати різні елементи, такі як кола, лінії, прямокутники та пружину, на візуальній поверхні (DrawingVisual)

Основні етапи роботи:

а) Створення DrawingVisual: Створюється новий об'єкт DrawingVisual, який служить контейнером для графічних елементів.

б) Малювання елементів за допомогою DrawingContext: використовуючи DrawingContext, визначаються різні графічні елементи (круги, лінії, прямокутники тощо) з заданими параметрами, такими як кольори, товщина ліній, стилі (штрихова та штрих-крапка-крапка).

в) Додавання DrawingVisual до VisualHost і Canvas: створюється екземпляр VisualHost, до якого додається створений DrawingVisual, а потім VisualHost додається до Canvas для відображення результатів на екрані.

Цей підхід дозволяє програмно створювати складні графічні представлення без прив'язки до стандартних елементів управління WPF, що відкриває широкі можливості для налаштування вигляду та поведінки графічних елементів у вашому додатку.

6 АНАЛІЗ МЕХАНІЧНОЇ СИСТЕМИ З ОДНИМ СТУПЕНЕМ СВОБОДИ

Дослідження вільних коливань механічної системи було спрямоване на виявлення відмінностей між розв'язками, отриманими для лінеаризованої та нелінійної версій рівняння руху.

Під час чисельних експериментів очевидно спостерігалася тенденція до того, що при малих початкових відхиленнях від положення рівноваги нелінійне рівняння Лагранжа надавало розв'язок (позначений червоним на графіку), який був близький до розв'язку лінеаризованого рівняння (позначений зеленим на графіку).

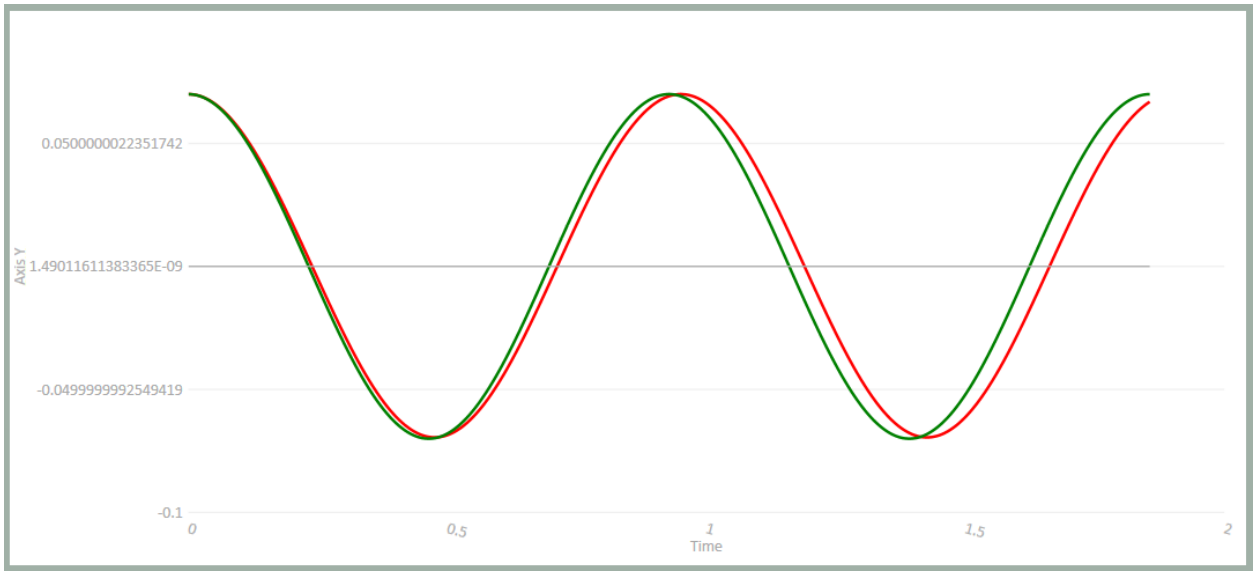


Рисунок 6.1 – Залежність $\xi = \xi(t)$ при малому початковому значенні $\xi_0 = 0.07$ м

Рисунок 6.2 – Залежність $\xi = \xi(t)$ при початковому значенні $\xi_0 = 0.35$ м

На рис. 6.1 можна спостерігати практично однакові значення як амплітуд, так і періодів розв'язків, що підтверджує характер нелінійних коливань, близьких до гармонійних. Однак при значних початкових відхиленнях системи від положення рівноваги, які можна вважати "скінченними", розв'язки відповідних рівнянь значно розходяться. Крім того, амплітуди нелінійних коливань для позитивної та негативної фаз руху значно різняться. Відрізняються і часи тривалості цих фаз (див. рис. 6.2 – 6.3).

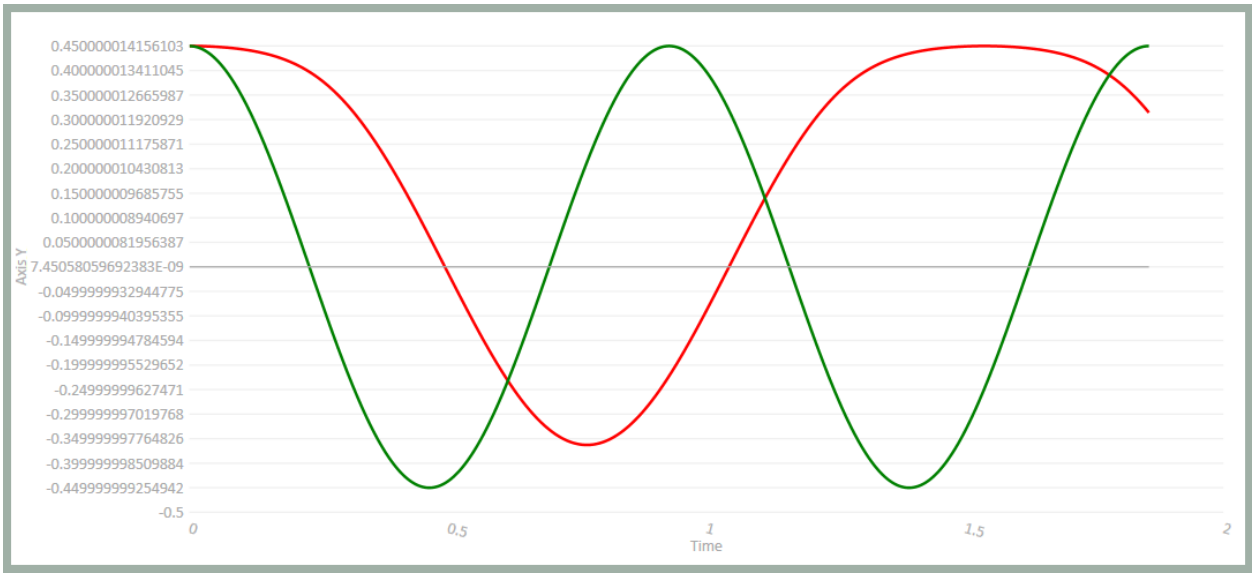


Рисунок 6.3 – Залежність $\xi = \xi(t)$ при початковому значенні $\xi_0 = 0.45$ м

На рис. 6.3 чітко видно, що нелінійні коливання мають період, майже на 50% більший, ніж період лінійних коливань. Чисельні експерименти проводилися з різними комбінаціями значень вхідних параметрів і підтвердили зроблені вище висновки.

Таким чином, мету дипломної роботи можна вважати досягнутою, а завдання з чисельного моделювання - вирішеним у вигляді Windows-застосунку.

ВИСНОВКИ

Отримане рівняння Лагранжа 2 порядку описує нелінійні коливання механічної системи з одним ступенем свободи. Під час виведення цього рівняння двома способами для потенційної енергії сили пружності проводився контроль правильності. Також було знайдено розв'язок лінеаризованої задачі (2.32), що описує малі коливання системи навколо положення рівноваги.

Для чисельного розв'язання рівняння Лагранжа використовувалася схема Рунге-Кутта четвертого порядку. Для розв'язання нелінійних алгебраїчних рівнянь типу (2.2) застосовувалися гібридні методи, зокрема метод пошуку мінімуму для невід'ємної функції та метод дихотомії.

Був проведен кінематичний аналіз можливих рухів заданої механічної системи, під час якого були виявлені граничні значення відхилень тіл системи, при яких відбуваються різні типи подальших рухів цих тіл.

Створено Windows-застосунок для чисельного експерименту, який дає можливість користувачеві змінювати вхідні параметри та проводити чисельний аналіз для нових наборів даних. Результати експерименту зберігаються у текстовому файлі для подальшого використання у роботі.

Створений застосунок також надає можливість візуального спостереження за кінематикою руху механічної системи в умовах рівномірного обертання тіла 2.

Таким чином, WPF застосунок інтегрується в проект для виведення рівнянь, виконання чисельного аналізу та візуалізації роботи системи, що спрощує аналіз результатів досліджень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Львов М.С., Співаковський О.В. Вступ до об'єктно-орієнтованого програмування. – Херсон: ХГПУ, 2000. – 238 с.

2. Воробйов В.В., Воробйова Л.Д., Киба С.П. Основи прикладної теорії коливань. – К.: Техніка, 2007. – 348 с.

3. Коноваленко І.В., Марущак П.О. Платформа .NET та мова програмування C# 8.0: Навчальний посібник. – Тернопіль: Навчальна книга - Богдан, 2020. – 256 с.

4. Офіційна документація Microsoft з Windows Presentation Foundation. – [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/desktop/wpf

5. Nathan A. WPF 4.5 Unleashed. – [Електронний ресурс]. – Режим доступу: https://books.google.com.ua/books?id=ubgRAAAAQBAJ&printsec=frontcover&redir_esc=y#v=onepage&q&f=false

6. MacDonald M. Pro WPF in C# 2010: Windows Presentation Foundation in .NET 4. – [Електронний ресурс]. – Режим доступу: https://books.google.com.ua/books?id=nYl7J7z3KssC&printsec=frontcover&redir_esc=y#v=onepage&q&f=false

7. Петренко І.В., Іваненко О.П. Аналіз коливань механічних систем з однією ступеню свободи // Вісник Національного технічного університету України «КПІ ім. І. Сікорського». – 2019. – №3. – С. 45-52.

8. Ковальчук С.В. Теоретична механіка: Навчальний посібник. – К.: Техніка, 2017. – 320 с.

9. Марущак П.О. Коливання механічних систем: курс лекцій. – Львів: Львівська політехніка, 2018. – 250 с.

10. Устенко О.В., Візняк Р.І., Ловська А.О. та ін. Основи теорії коливань та стійкості рухомого складу: Навч. посібник. – Харків: УкрДУЗТ, 2021. – 129 с.

ДОДАТОК А
Головне вікно MainWindow.xaml

```
<Window x:Class="DiplomDA.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:local="clr-namespace:DiplomDA"
        xmlns:lvc="clr-
namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"
        mc:Ignorable="d"
        Title="Механична система з одним ступенем свободи"
        Height="930" Width="1360" Background="#a0b0a6"
        MaxWidth="{Binding Source={x:Static
SystemParameters.PrimaryScreenWidth}}"
        MaxHeight="{Binding Source={x:Static
SystemParameters.PrimaryScreenHeight}}"
        FontSize="16" FontWeight="Bold" Loaded="Window_Loaded">

    <Window.Resources>
        <Style x:Key="BorderWithStyle" TargetType="Border">
            <Setter Property="BorderBrush" Value="#b4a399"/>
            <Setter Property="BorderThickness" Value="2"/>
            <Setter Property="CornerRadius" Value="10"/>
            <Setter Property="Height" Value="55"/>
            <Setter Property="Background">
                <Setter.Value>
                    <LinearGradientBrush StartPoint="0,0"
EndPoint="1,1">
                        <GradientStop Color="#ffffff"
Offset="0.0"/>
                        <GradientStop Color="#e0cfbd"
Offset="1.0"/>
                    </LinearGradientBrush>
                </Setter.Value>
            </Setter>
            <Setter Property="Effect">
                <Setter.Value>
                    <DropShadowEffect Color="Black"
Direction="320" ShadowDepth="5" BlurRadius="10"/>
                </Setter.Value>
```

```
            </Setter>
        </Style>

        <Style x:Key="ButtonStyle" TargetType="ButtonBase">
            <Setter Property="Margin" Value="10"/>
            <Setter Property="Height" Value="50"/>
            <Setter Property="Width" Value="200"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="FontSize" Value="16"/>
            <Setter Property="Foreground" Value="Black"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="ButtonBase">
                        <Border CornerRadius="20"
BorderThickness="6" BorderBrush="#b4a399">
                            <Border.Background>
                                <LinearGradientBrush
StartPoint="0,0" EndPoint="1,1">
                                    <GradientStop
Color="#e0cfbd" Offset="0.0"/>
                                    <GradientStop
Color="#d0bfad" Offset="1.0"/>
                                </LinearGradientBrush>
                            </Border.Background>
                            <Border.Effect>
                                <DropShadowEffect Color="Black"
Direction="320"

ShadowDepth="10" BlurRadius="15"/>
                            </Border.Effect>
                            <ContentPresenter
HorizontalAlignment="Center"

VerticalAlignment="Center"/>
                        </Border>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>

        <Style x:Key="ToggleButtonStyle"
TargetType="ToggleButton">
            <Setter Property="Margin" Value="10,20,10,10"/>
            <Setter Property="Height" Value="65"/>
            <Setter Property="Width" Value="65"/>
            <Setter Property="FontWeight" Value="Bold"/>
```

```
            <Setter Property="FontSize" Value="16"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="ToggleButton">
                        <Border x:Name="Border"
CornerRadius="50" BorderThickness="6" BorderBrush="#b4a399">
                            <Border.Background>
                                <LinearGradientBrush
StartPoint="0,0" EndPoint="1,1">
                                    <GradientStop
Color="#e0cfbd" Offset="0.0"/>
                                    <GradientStop
Color="#d0bfad" Offset="1.0"/>
                                </LinearGradientBrush>
                            </Border.Background>
                            <Border.Effect>
                                <DropShadowEffect
x:Name="ShadowEffect" Color="Black" Direction="320"
ShadowDepth="10" BlurRadius="15"/>
                            </Border.Effect>
                            <ContentPresenter
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                        </Border>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsChecked"
Value="True">
                                <Setter TargetName="Border"
Property="Effect">
                                    <Setter.Value>
                                        <DropShadowEffect
Color="Black" Direction="320" ShadowDepth="0" BlurRadius="0"/>
                                    </Setter.Value>
                                </Setter>
                            </Trigger>
                            <Trigger Property="IsChecked"
Value="False">
                                <Setter TargetName="Border"
Property="Effect">
                                    <Setter.Value>
                                        <DropShadowEffect
Color="Black" Direction="320" ShadowDepth="10" BlurRadius="15"/>
                                    </Setter.Value>
                                </Setter>
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
```

```
                </Setter.Value>
            </Setter>
        </Style>
    </Window.Resources>

    <Grid Margin="0">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="300*"/>
            <ColumnDefinition Width="904*"/>
            <ColumnDefinition Width="366*"/>
        </Grid.ColumnDefinitions>
        <Grid Grid.Column="1">
            <Grid.RowDefinitions>
                <RowDefinition Height="450*"/>
                <RowDefinition Height="545*"/>
            </Grid.RowDefinitions>
            <Border Grid.Row ="0" BorderBrush="#a0b0a6"
BorderThickness="10" CornerRadius="20">
                <Canvas x:Name="canvas_System"
Background="#ffffff"/>
            </Border>

            <Border Grid.Row="1" BorderBrush="#a0b0a6"
BorderThickness="10" CornerRadius="20" Margin="-10,10,10,101" >
                <lvc:CartesianChart x:Name="chart"
                                    Background="White"
Margin="10,16,-10,-92"/>
            </Border>

        </Grid>
        <Grid Grid.Column="0">
            <Grid.RowDefinitions>
                <RowDefinition Height ="Auto"/>
                <RowDefinition Height ="Auto"/>
                <RowDefinition Height ="Auto"/>
                <RowDefinition Height ="Auto"/>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="300"/>
            </Grid.ColumnDefinitions>
            <Viewbox VerticalAlignment="Center"
                     HorizontalAlignment="Center"
Stretch="Uniform" Height="200" Margin="0,0,48,0">
                <Border Style="{StaticResource BorderWithStyle}"
                    Height="230"
                    VerticalAlignment="Top"
```

```
                    HorizontalAlignment="Left"
                    Margin="5"
                    Width="250">
                    <StackPanel>
                        <Label Content="Початкові умови:"
FontSize="16" HorizontalAlignment="Left" Height="31"
Margin="24,15,0,0" Width="198"/>
                        <Label Margin="10,3,0,0" FontSize="14">
                            <Label.Content>
                                <WrapPanel Width="278">
                                    <Image Source="/Ksi.png"
Height="30" Width="30"/>
                                    <TextBlock
x:Name="TBlock_Ksi_0" Text=" = 0.000 м" Height="22" Width="84"/>
                                </WrapPanel>
                            </Label.Content>
                        </Label>
                        <Label Margin="10,3,0,0" FontSize="14">
                            <Label.Content>
                                <WrapPanel Width="278">
                                    <Image Source="/Fi.png"
Height="30" Width="30"/>
                                    <TextBlock
x:Name="TBlock_Fi_0" Text=" = 00.0 м" Height="22" Width="84"/>
                                </WrapPanel>
                            </Label.Content>
                        </Label>
                        <Label Margin="10,3,0,0" FontSize="14">
                            <Label.Content>
                                <WrapPanel Width="278">
                                    <Image Source="/Psi.png"
Height="30" Width="30"/>
                                    <TextBlock
x:Name="TBlock_Psi_0" Text=" = 00.0 м" Height="22" Width="84"/>
                                </WrapPanel>
                            </Label.Content>
                        </Label>
                        <Button Background="#99a0b0a6"
Grid.Column="1"  Grid.Row="0" Content="Змінити"
                            Margin="0,5,0,0"
HorizontalAlignment="Center"
                            FontSize="16" FontWeight="Bold"
Height="40" Width="126" Click="Button_Click"/>
                    </StackPanel>
                </Border>
            </Viewbox>
```

```
            <Viewbox  VerticalAlignment="Center"
                      HorizontalAlignment="Center" Grid.Row="1"
Stretch="Uniform" Height="200" Margin="0,0,48,0">
                <Border Style="{StaticResource BorderWithStyle}"
                    Grid.Row="1"
                    Height="229"
                    VerticalAlignment="Top"
                    HorizontalAlignment="Left"
                    Width="239">
                    <StackPanel>
                        <Label Content="Розміри тіл"
HorizontalAlignment="Left" Height="32" Margin="24,15,0,0"
Width="139" FontWeight="Bold" FontSize="18"/>
                        <Label x:Name="Lbl_sizeR" Content="R = "
HorizontalAlignment="Left" Height="25" Margin="24,5,0,0"
VerticalAlignment="Top" Width="90" FontWeight="Bold"
FontSize="14"/>
                        <Label x:Name="Lbl_sizer" Content="r = "
HorizontalAlignment="Left" Height="25" Margin="24,5,0,0"
VerticalAlignment="Top" Width="90" FontWeight="Bold"
FontSize="14"/>
                        <Label x:Name="Lbl_sizeL" Content="L = "
HorizontalAlignment="Left" Height="25" Margin="24,5,0,0"
VerticalAlignment="Top" Width="90" FontWeight="Bold"
FontSize="14"/>
                        <Label x:Name="Lbl_sizeH" Content="H = "
HorizontalAlignment="Left" Height="25" Margin="24,5,0,0"
VerticalAlignment="Top" Width="90" FontWeight="Bold"
FontSize="14"/>
                        <Button Background="#99a0b0a6"
Grid.Column="1"  Grid.Row="1"
                                Content="Змінити"  Margin="0,5,0,0"
HorizontalAlignment="Center"
                                FontSize="16" FontWeight="Bold"
Height="40" Width="126" Click="Button_Click_1"/>
                    </StackPanel>
                </Border>
            </Viewbox>
            <Viewbox  VerticalAlignment="Center"
                      HorizontalAlignment="Center" Grid.Row="2"
Stretch="Uniform" Height="200" Margin="0,0,48,0">
                <Border Style="{StaticResource BorderWithStyle}"
                    Grid.Row="2"
                    Height="199"
                    VerticalAlignment="Top"
                    HorizontalAlignment="Left"
```

```
                    Width="209">
                    <StackPanel>
                        <Label Content="Маса"
HorizontalAlignment="Left" Height="32" Margin="24,15,0,0"
Width="90" FontWeight="Bold" FontSize="18"/>
                        <Label x:Name="Lbl_weight1" Content="m1
= " HorizontalAlignment="Left" Height="25" Margin="24,5,0,0"
VerticalAlignment="Top" Width="90" FontWeight="Bold"
FontSize="14"/>
                        <Label x:Name="Lbl_weight2" Content="m2
= " HorizontalAlignment="Left" Height="25" Margin="24,5,0,0"
VerticalAlignment="Top" Width="90" FontWeight="Bold"
FontSize="14"/>
                        <Label x:Name="Lbl_weight3" Content="m3
= " HorizontalAlignment="Left" Height="25" Margin="24,5,0,0"
VerticalAlignment="Top" Width="90" FontWeight="Bold"
FontSize="14"/>
                        <Button Background="#99a0b0a6"
Grid.Column="1"  Grid.Row="2" Content="Змінити"
Margin="0,5,0,0" HorizontalAlignment="Center"
                                FontSize="16" FontWeight="Bold"
Height="40" Width="126" Click="Button_Click_2"/>
                    </StackPanel>
                </Border>
            </Viewbox>
            <Viewbox  VerticalAlignment="Center"
                      HorizontalAlignment="Center"
                      Grid.Row="3"  Stretch="Uniform"
Height="200" Margin="0,0,48,0">
                <Border Style="{StaticResource BorderWithStyle}"
                    Grid.Row="3"
                    Height="230"
                    VerticalAlignment="Center"
                    HorizontalAlignment="Center"
                    Margin="5"
                    Width="250">
                    <StackPanel>
                        <Label Content="Пружина"
HorizontalAlignment="Left" Height="37" Margin="24,15,0,0"
Width="109" FontWeight="Bold" FontSize="18"/>
                        <Label x:Name="Lbl_springC" Content="C =
" HorizontalAlignment="Left" Height="25" Margin="24,5,0,0"
VerticalAlignment="Top" Width="100" FontWeight="Bold"
FontSize="14"/>
                        <Label x:Name="Lbl_springDc" Content="Dc
= " HorizontalAlignment="Left" Height="25" Margin="24,5,0,0"
```

```
VerticalAlignment="Top" Width="100" FontWeight="Bold"
FontSize="14"/>
                        <Label x:Name="Lbl_springn3" Content="n3
= " HorizontalAlignment="Left" Height="25" Margin="24,5,0,0"
VerticalAlignment="Top" Width="100" FontWeight="Bold"
FontSize="14"/>
                        <Label x:Name="Lbl_springDs" Content="Ds
= " HorizontalAlignment="Left" Height="25" Margin="24,5,0,0"
VerticalAlignment="Top" Width="100" FontWeight="Bold"
FontSize="14"/>
                        <Button Background="#99a0b0a6"
Grid.Column="1"  Grid.Row="3"
                             Content="Змінити"  Margin="0,5,0,0"
HorizontalAlignment="Center"
                            FontSize="16" FontWeight="Bold"
Height="40" Width="126" Click="Button_Click_3"/>
                    </StackPanel>
                </Border>
            </Viewbox>
        </Grid>
        <Border BorderBrush="Black"
                BorderThickness="1" Grid.Column="2"
                HorizontalAlignment="Left" Height="830"
VerticalAlignment="Top"
                Width="310" Margin="5,10,0,0">
            <Grid Grid.Column="1">
                <Grid.RowDefinitions>
                    <RowDefinition/>
                    <RowDefinition Height="410"/>
                </Grid.RowDefinitions>

                <StackPanel Grid.Row="0" Height="450"
VerticalAlignment="Top" Grid.RowSpan="2">
                    <Border Height="265">
                        <StackPanel>
                            <Label Margin="10" FontSize="16">
                                <Label.Content>
                                    <WrapPanel Width="278">
                                        <TextBlock Text="Період
малих коливань" FontWeight="Bold" Margin="5"></TextBlock>
                                        <TextBox x:Name="Period"
Text="" Width="50" FontSize="15"></TextBox>
                                        <TextBlock Text="(c)"
Margin="5"></TextBlock>
                                    </WrapPanel>
                                </Label.Content>
```

```
                            </Label>
                            <Label Margin="10,3,0,0"
FontSize="16">
                                <Label.Content>
                                    <Border
Style="{StaticResource BorderWithStyle}">
                                        <WrapPanel Width="278">
                                            <Image
Source="/Ksimax.png" Height="48" Width="55"/>
                                            <TextBlock
x:Name="TBlock_Ksi" Text=" = 0.000 м" Height="22" Width="84"

VerticalAlignment="Center" FontWeight="Bold"/>
                                        </WrapPanel>
                                    </Border>
                                </Label.Content>
                            </Label>
                            <Label Margin="10,3,0,0"
FontSize="16">
                                <Label.Content>
                                    <Border
Style="{StaticResource BorderWithStyle}">
                                        <WrapPanel Width="278">
                                            <Image
Source="/Fimax.png" Height="48" Width="55"/>
                                            <TextBlock
x:Name="TBlock_Fi" Text=" = 00.0 гр" Height="22" Width="84"

FontWeight="Bold"/>
                                        </WrapPanel>
                                    </Border>
                                </Label.Content>
                            </Label>
                            <Label Margin="10,3,0,0"
FontSize="16">
                                <Label.Content>
                                    <Border
Style="{StaticResource BorderWithStyle}">
                                        <WrapPanel Width="278">
                                            <Image
Source="/Psimax.png" Height="48" Width="55"/>
                                            <TextBlock
x:Name="TBlock_Psi" Text=" = 00.0 гр" Height="22" Width="84"

FontWeight="Bold"/>
                                        </WrapPanel>
```

```
                                    </Border>
                                </Label.Content>
                            </Label>
                        </StackPanel>
                    </Border>
                    <Button x:Name="Start" Content="Старт"
Style="{StaticResource ButtonStyle}" Click="Start_Click"/>
                    <Button x:Name="Stop" Content="Стоп"
Style="{StaticResource ButtonStyle}" Click="Stop_Click"/>


                </StackPanel>

                <StackPanel Grid.Row="1" Height="395"
VerticalAlignment="Top">
                    <Border Height="150">
                        <WrapPanel Margin="0,10,0,0">
                            <ToggleButton x:Name="toggleKsi"
Margin="35,20,10,10" Style="{StaticResource ToggleButtonStyle}"

Checked="toggleKsi_Checked" Unchecked="toggleKsi_Unchecked">
                                <Image Width="37" Height="37"
Source="/ksi_color.png"/>
                            </ToggleButton>

                            <ToggleButton x:Name="toggleFi"
Style="{StaticResource ToggleButtonStyle}"

Checked="toggleFi_Checked" Unchecked="toggleFi_Unchecked">
                                <Image Width="37" Height="37"
Source="/fi_color.png"/>
                            </ToggleButton>
                            <ToggleButton x:Name="togglePsi"
Style="{StaticResource ToggleButtonStyle}"

Checked="togglePsi_Checked" Unchecked="togglePsi_Unchecked">
                                <Image Width="37" Height="37"
Source="/psi_color.png">
                                </Image>
                            </ToggleButton>
                        </WrapPanel>
                    </Border>
                    <Label x:Name="Raschet" FontSize="16"
FontWeight="Bold">

                    </Label>
```

```
                    <Button x:Name="Traek" Content="Намалювати
траєкторії" Margin="10,30,0,20"
                            Style="{StaticResource ButtonStyle}"
Click="Traek_Click" >

                    </Button>
                    <Button x:Name="Clear" Content="Стерти
графік" Margin="10,10,0,20" Height="50" Width="192"
                             Style="{StaticResource ButtonStyle}"
Click="Clear_Click">
                    </Button>
                </StackPanel>
            </Grid>

        </Border>
    </Grid>
</Window>
```

ДОДАТОК Б.

Лістинг програмної реалізації класу MainWindow

```
public partial class MainWindow : Window
 {
     public static double m1, m2, m3;
     public static double r, R, L, H, n3, Lc, Dc, C, Ds, DcDs;
     public static double omega, omega2, sigma, J1, J2, J3, r2,
R2, L2;
     public SeriesCollection SeriesCollection { get; set; }
     public delegate double F_t_x_v(double t, double x, double v);

     public static double M1, M2, t_init, t_full, t_step, Psi_Ini;
     public static double Ksi_Max, Ksi_Ini, Phi_Max, Phi_Ini,
Psi_Max;
     public static double period_L, period_N;
     public static bool clean, start;

     public const double grad = 180.0 / Math.PI, g = 9.81, eps =
1.0E-8;
     public const float fgrad = (float)grad;
     public const double nC = 0.15;
     private DispatcherTimer timer;
     private void Start_Click(object sender, RoutedEventArgs e)
     {
         timer.Start();
     }
     private void Stop_Click(object sender, RoutedEventArgs e)
     {
         timer.Stop();
     }
     private void Timer_Tick(object sender, EventArgs e)
     {
         movePicture();
         dt = dt + 0.1f;
     }
     private massa f_massa;
     private size f_length;
     private zigzag f_coil;
     private first f_init;
     private void Button_Click(object sender, RoutedEventArgs e)
     {
         f_init     =     new     first();     f_init.Activate();
f_init.ShowDialog();
         Parameters(); chart.Series.Clear();
```

```
    }
    private void Button_Click_1(object sender, RoutedEventArgs e)
    {
        f_length = new size(); f_length.Activate();
f_length.ShowDialog();
        Parameters(); chart.Series.Clear();
    }
    private void Button_Click_2(object sender, RoutedEventArgs e)
    {
        f_massa = new massa(); f_massa.Activate();
f_massa.ShowDialog();
        Parameters(); chart.Series.Clear();
    }

    private void Button_Click_3(object sender, RoutedEventArgs e)
    {
        f_coil = new zigzag(); f_coil.Activate();
f_coil.ShowDialog();
        Parameters(); chart.Series.Clear();
    }
    public const int Nt = 500, Mt = Nt + 1;
    public static float[] Ksi_Num, dKsi_Num, Ksi_Anl, dKsi_Anl,
Time;
    public static float[] Phi_Num, Phi_Anl, Psi_Anl, Psi_Num;

    public Point Center = new Point(200, 200);
    public int R1 = 50;
    public int Rr2 = 100;
    public double Omega = 0.1;
    public double t = 0;
    public bool cl = true;
    public bool timerEnd = false;
    private float dt = 0.1f;
    public MainWindow()
    {
        InitializeComponent();
        timer = new DispatcherTimer();
        timer.Interval = TimeSpan.FromSeconds((float)dt);
        timer.Tick += Timer_Tick;

        #region налаштування вісі X та У chart
        chart.AxisX.Add(new Axis
        {
            Title = "Time",
            Position = AxisPosition.LeftBottom, // Це стандартне
положення осі X
```

```
            LabelsRotation = 15,
            MinValue = 0, // Мінімальне значення осі X
            MaxValue = 2, // Максимальне значення осі X
            Separator = new LiveCharts.Wpf.Separator
            {
                Step = 0.5,
                IsEnabled = false
            }
        });

        chart.AxisY.Add(new Axis
        {
            Title = "Values",
            Position = AxisPosition.LeftBottom, // Переміщуємо
ось Y вверх
            MinValue = -0.4, // Мінімальне значення осі Y
            MaxValue = 0.4, // Максимальне значення осі Y
            IsMerged = false,
            Separator = new LiveCharts.Wpf.Separator
            {
                Step = 5,
                IsEnabled = true,
            }
        });
        #endregion

        #region Початкові умови
        clean = false; start = true;
        m1 = 5.0;
        m2 = 2.0;
        m3 = 6.0;

        r = 0.35;
        R = 1.00;
        L = 2.00;
        H = 3.00;

        C = 250.0;
        Dc = 1.5;
        n3 = 0.75;
        #endregion

        Parameters();
        start = false;

        PictureSystem(cl);
```

```
    }
    private void movePicture()
    {
        // Створення DrawingVisual для малювання
        DrawingVisual drawingVisual = new DrawingVisual();
        using (DrawingContext drawingContext =
drawingVisual.RenderOpen())
        {
            Pen pen = new Pen(Brushes.Black, 1);
            Brush brush = Brushes.White;


            Rect rect = new Rect(Center.X - Rr2, Center.Y - Rr2,
2 * Rr2, 2 * Rr2);
            drawingContext.DrawEllipse(brush, pen, Center, Rr2,
Rr2);
            pen = new Pen(cl ? Brushes.LightGray : Brushes.White,
1);
            pen.DashStyle = DashStyles.Dash;
            double fi = Omega * t;
            Point a1 = new Point(Center.X + Rr2 * Math.Sin(fi),
Center.Y - Rr2 * Math.Cos(fi));
            Point a2 = new Point(Center.X - Rr2 * Math.Sin(fi),
Center.Y + Rr2 * Math.Cos(fi));
            drawingContext.DrawLine(pen, a1, a2);
            a1 = new Point(Center.X + Rr2 * Math.Cos(fi),
Center.Y + Rr2 * Math.Sin(fi));
            a2 = new Point(Center.X - Rr2 * Math.Cos(fi),
Center.Y - Rr2 * Math.Sin(fi));
            drawingContext.DrawLine(pen, a1, a2);
            pen = new Pen(Brushes.Black, 1);
            rect = new Rect(Center.X - R1, Center.Y - R1, 2 * R1,
2 * R1);
            drawingContext.DrawEllipse(brush, pen, Center, R1,
R1);

            pen = new Pen(cl ? Brushes.Black : Brushes.White, 1);
            pen.DashStyle = DashStyles.DashDotDot;
            Point b1 = new Point(Center.X + R1 * Math.Sin(fi),
Center.Y - R1 * Math.Cos(fi));
            Point b2 = new Point(Center.X - R1 * Math.Sin(fi),
Center.Y + R1 * Math.Cos(fi));
            drawingContext.DrawLine(pen, b1, b2);
            b1 = new Point(Center.X + R1 * Math.Cos(fi), Center.Y
+ R1 * Math.Sin(fi));
```

```
            b2 = new Point(Center.X - R1 * Math.Cos(fi), Center.Y
- R1 * Math.Sin(fi));
            drawingContext.DrawLine(pen, b1, b2);
            DrawingBrush hatchBrush = CreateHatchBrush();
            pen = new Pen(Brushes.Black, 1);
            Point tochka = new Point(Center.X - 3, Center.Y + 6);
            drawingContext.DrawLine(pen, tochka, Center);
            tochka = new Point(Center.X + 3, Center.Y + 6);
            drawingContext.DrawLine(pen, Center, tochka);

            pen = new Pen(Brushes.White, 1);
            rect = new Rect(Center.X - 10, Center.Y + 10, 20, 5);
            drawingContext.DrawRectangle(hatchBrush, pen, rect);

            pen = new Pen(cl ? Brushes.Black : Brushes.White, 1);
            drawingContext.DrawLine(pen, new Point(Center.X + 5,
Center.Y + 6), new Point(Center.X - 5, Center.Y + 6));

            rect = new Rect(Center.X - 3, Center.Y - 3, 4, 4);
            drawingContext.DrawEllipse(hatchBrush, pen, Center,
2, 2);
            drawingContext.DrawLine(pen, new Point(Center.X -
Rr2, Center.Y), new Point(Center.X - Rr2, Center.Y - Omega * Rr2
* t + 80));
            rect = new Rect(Center.X - Rr2 - 10, Center.Y - Omega
* Rr2 * t + 80, 20, 20);
            drawingContext.DrawRectangle(brush, pen, rect);

            pen = new Pen(Brushes.Black, 1);
            // стержень S
            rect = new Rect(Center.X + R1*Math.Cos(-0.5f * dt -
(float)Math.PI * 1.5f), Center.Y + (R1 * Math.Sin(-0.5f * dt -
(float)Math.PI * 1.5f)) - 2.5 , 350, 5);
            drawingContext.DrawRectangle(brush, pen, rect);

            float pivotX = (float)Center.X + 347;
            float pivotY = (float)Center.Y - 180;

            float   rotatedEndX   =   pivotX   +   (180)   *
(float)Math.Cos(-0.1f * dt);
            float   rotatedEndY   =   pivotX   +   (180)   *
(float)Math.Sin(-0.1f * dt);
            double lineThickness = 5.0;
            brush = Brushes.Black;
            Pen linePen = new Pen(brush, lineThickness);
            brush = Brushes.White;
```

```
            rect = new Rect(Center.X + 347, Center.Y - 180 , 5,
180 + R1);
            drawingContext.PushTransform(new   RotateTransform(-
0.1f * dt * 180 / Math.PI, Center.X + 347, Center.Y - 180));
            drawingContext.DrawRectangle(brush, pen, rect);
            drawingContext.Pop();

            pen = new Pen(Brushes.Black, 1);

            Point point1 = new Point(Center.X, Center.Y + R1);
            drawingContext.DrawEllipse(brush, pen, point1, 2.5,
2.5);

            point1 = new Point(Center.X + 350, Center.Y + R1);
            drawingContext.DrawEllipse(brush, pen, point1, 2.5,
2.5);

            point1 = new Point(Center.X + 350, Center.Y - 180);
            drawingContext.DrawEllipse(brush, pen, point1, 2.5,
2.5);

            Point point2 = new Point(Center.X + 350 - 3, Center.Y
- 180 - 6);
            Point center = new Point(Center.X + 350, Center.Y -
180);
            drawingContext.DrawLine(pen, point2, center);
            point2 = new Point(Center.X + 350 + 3, Center.Y - 180
- 6);
            drawingContext.DrawLine(pen, center, point2);
            pen = new Pen(Brushes.White, 1);
            rect = new Rect(Center.X + 350 - 10, Center.Y - 180
- 10, 20, 5);
            drawingContext.DrawRectangle(hatchBrush, pen, rect);
            pen = new Pen(Brushes.Black, 1);
            point2 = new Point(Center.X + 350 - 10, Center.Y -
180 - 5);
            point1 = new Point(Center.X + 350 + 10, Center.Y -
180 - 5);
            drawingContext.DrawLine(pen, point2, point1);
            double coilLength = 400;
            double zigzagAmplitude = 25;
            double angle = Math.PI * 80 / 180;

            Point p0 = new Point(Center.X + 266, Center.Y - R1 +
12);
```

```
            Point startPoint = new Point(Center.X + 270, Center.Y
- R1);
            pen = new Pen(Brushes.White, 1);
            rect = new Rect(Center.X + 260, Center.Y - R1, 5,
20);
            drawingContext.DrawRectangle(hatchBrush, pen, rect);
            pen = new Pen(Brushes.Black, 1);
            drawingContext.DrawLine(pen, p0, startPoint);

            p0 = new Point(Center.X + 266, Center.Y - R1 + 26);
            Point p00 = new Point(Center.X + 266, Center.Y - R1);
            drawingContext.DrawLine(pen, p0, p00);
            double totalLength = 0;
            bool isUpward = true;
            Point p1, p2 = new Point();
            while (totalLength < coilLength)
            {
                double dx1 = Math.Cos(angle) * zigzagAmplitude;
                double dy1 = Math.Sin(angle) * zigzagAmplitude;

                p1 = new Point(startPoint.X + dx1, startPoint.Y
+ dy1);
                drawingContext.DrawLine(new   Pen(Brushes.Black,
1), startPoint, p1);

                startPoint = p1;

                double dx2 = Math.Cos(angle) * zigzagAmplitude;
                double dy2 = -Math.Sin(angle) * zigzagAmplitude;

                p2 = new Point(startPoint.X + dx2, startPoint.Y
+ dy2);
                drawingContext.DrawLine(new   Pen(Brushes.Black,
1), startPoint, p2);

                startPoint = p2;

                totalLength += 2 * zigzagAmplitude;
            }
            p00 = new Point(Center.X + 350 - 2.5, Center.Y - R1
+ 24);
            drawingContext.DrawLine(pen, p2, p00);
        }
        VisualHost visualHost = new VisualHost();
        visualHost.AddVisual(drawingVisual);
        canvas_System.Children.Add(visualHost);
```

```
        }
        private void PictureSystem(bool cl)
        {
            DrawingVisual drawingVisual = new DrawingVisual();
            using (DrawingContext drawingContext =
drawingVisual.RenderOpen())
            {
                Pen pen = new Pen(Brushes.Black, 1);
                Brush brush = Brushes.White;

                Rect rect = new Rect(Center.X - Rr2, Center.Y - Rr2,
2 * Rr2, 2 * Rr2);
                drawingContext.DrawEllipse(brush, pen, Center, Rr2,
Rr2);
                pen = new Pen(cl ? Brushes.LightGray : Brushes.White,
1);
                pen.DashStyle = DashStyles.Dash;
                double fi = Omega * t;
                Point a1 = new Point(Center.X + Rr2 * Math.Sin(fi),
Center.Y - Rr2 * Math.Cos(fi));
                Point a2 = new Point(Center.X - Rr2 * Math.Sin(fi),
Center.Y + Rr2 * Math.Cos(fi));
                drawingContext.DrawLine(pen, a1, a2);
                a1 = new Point(Center.X + Rr2 * Math.Cos(fi),
Center.Y + Rr2 * Math.Sin(fi));
                a2 = new Point(Center.X - Rr2 * Math.Cos(fi),
Center.Y - Rr2 * Math.Sin(fi));
                drawingContext.DrawLine(pen, a1, a2);
                pen = new Pen(Brushes.Black, 1);
                rect = new Rect(Center.X - R1, Center.Y - R1, 2 * R1,
2 * R1);
                drawingContext.DrawEllipse(brush, pen, Center, R1,
R1);
                pen = new Pen(cl ? Brushes.Black : Brushes.White, 1);
                pen.DashStyle = DashStyles.DashDotDot;
                Point b1 = new Point(Center.X + R1 * Math.Sin(fi),
Center.Y - R1 * Math.Cos(fi));
                Point b2 = new Point(Center.X - R1 * Math.Sin(fi),
Center.Y + R1 * Math.Cos(fi));
                drawingContext.DrawLine(pen, b1, b2);
                b1 = new Point(Center.X + R1 * Math.Cos(fi), Center.Y
+ R1 * Math.Sin(fi));
                b2 = new Point(Center.X - R1 * Math.Cos(fi), Center.Y
- R1 * Math.Sin(fi));
                drawingContext.DrawLine(pen, b1, b2);
                DrawingBrush hatchBrush = CreateHatchBrush();
```

```
            pen = new Pen(Brushes.Black, 1);
            Point tochka = new Point(Center.X - 3, Center.Y + 6);
            drawingContext.DrawLine(pen, tochka, Center);
            tochka = new Point(Center.X + 3, Center.Y + 6);
            drawingContext.DrawLine(pen, Center, tochka);

            pen = new Pen(Brushes.White, 1);
            rect = new Rect(Center.X - 10, Center.Y + 10, 20, 5);
            drawingContext.DrawRectangle(hatchBrush, pen, rect);

            pen = new Pen(cl ? Brushes.Black : Brushes.White, 1);
            drawingContext.DrawLine(pen, new Point(Center.X + 5,
Center.Y + 6), new Point(Center.X - 5, Center.Y + 6));

            rect = new Rect(Center.X - 3, Center.Y - 3, 4, 4);
            drawingContext.DrawEllipse(hatchBrush, pen, Center,
2, 2);
            drawingContext.DrawLine(pen, new Point(Center.X -
Rr2, Center.Y), new Point(Center.X - Rr2, Center.Y - Omega * Rr2
* t + 80));
            rect = new Rect(Center.X - Rr2 - 10, Center.Y - Omega
* Rr2 * t + 80, 20, 20);
            drawingContext.DrawRectangle(brush, pen, rect);
            rect = new Rect(Center.X, Center.Y + R1-2.5, 350, 5);
            drawingContext.DrawRectangle(brush, pen, rect);
            rect = new Rect(Center.X+347, Center.Y-180, 5,
180+R1);
            drawingContext.DrawRectangle(brush, pen, rect);

            pen = new Pen(Brushes.Black, 1);

            Point point1 = new Point(Center.X, Center.Y + R1);
            drawingContext.DrawEllipse(brush, pen, point1, 2.5,
2.5);

            point1 = new Point(Center.X + 350, Center.Y+R1);
            drawingContext.DrawEllipse(brush, pen, point1, 2.5,
2.5);

            point1 = new Point(Center.X + 350, Center.Y -180);
            drawingContext.DrawEllipse(brush, pen, point1, 2.5,
2.5);

            Point point2 = new Point(Center.X +350 - 3, Center.Y-
180-6);
```

```
            Point center = new Point(Center.X + 350, Center.Y -
180);
            drawingContext.DrawLine(pen, point2, center);
            point2 = new Point(Center.X + 350 +3, Center.Y - 180
- 6);
            drawingContext.DrawLine(pen, center, point2);
            pen = new Pen(Brushes.White, 1);
            rect = new Rect(Center.X + 350- 10, Center.Y-180 -
10, 20, 5);
            drawingContext.DrawRectangle(hatchBrush, pen, rect);
            pen = new Pen(Brushes.Black, 1);
            point2 = new Point(Center.X + 350 - 10, Center.Y -
180 -5);
            point1 = new Point(Center.X + 350 + 10, Center.Y -
180 -5);
            drawingContext.DrawLine(pen, point2, point1);

            double coilLength = 400; // Длина пружины
            double zigzagAmplitude = 25; // Амплитуда зигзага
            double angle = Math.PI * 80 / 180; // Угол наклона
зигзагов

            Point p0 = new Point(Center.X + 266, Center.Y - R1 +
12);
            Point startPoint = new Point(Center.X + 270, Center.Y
- R1);
            pen = new Pen(Brushes.White, 1);
            rect = new Rect(Center.X + 260, Center.Y - R1, 5,
20);
            drawingContext.DrawRectangle(hatchBrush, pen, rect);
            pen = new Pen(Brushes.Black, 1);
            drawingContext.DrawLine(pen, p0, startPoint);

            p0 = new Point(Center.X + 266, Center.Y - R1 + 26);
            Point p00 = new Point(Center.X + 266, Center.Y - R1);
            drawingContext.DrawLine(pen, p0, p00);
            double totalLength = 0;
            bool isUpward = true; // Флаг для определения
направления текущего зигзага
            Point p1, p2 = new Point();
            while (totalLength < coilLength)
            {
                double dx1 = Math.Cos(angle) * zigzagAmplitude;
                double dy1 = Math.Sin(angle) * zigzagAmplitude;
```

```
                p1 = new Point(startPoint.X + dx1, startPoint.Y
+ dy1);
                drawingContext.DrawLine(new   Pen(Brushes.Black,
1), startPoint, p1);

                startPoint = p1;

                double dx2 = Math.Cos(angle) * zigzagAmplitude;
                double dy2 = -Math.Sin(angle) * zigzagAmplitude;

                p2 = new Point(startPoint.X + dx2, startPoint.Y
+ dy2);
                drawingContext.DrawLine(new   Pen(Brushes.Black,
1), startPoint, p2);

                startPoint = p2;

                totalLength += 2 * zigzagAmplitude;
            }
            p00 = new Point(Center.X + 350-2.5, Center.Y - R1 +
24);
            drawingContext.DrawLine(pen, p2, p00);
        }

        // Добавление DrawingVisual в Canvas
        VisualHost visualHost = new VisualHost();
        visualHost.AddVisual(drawingVisual);
        canvas_System.Children.Add(visualHost);
    }
        private DrawingBrush CreateHatchBrush()
        {
            GeometryGroup geometryGroup = new GeometryGroup();
            for (int i = 0; i < 10; i++)
            {
                geometryGroup.Children.Add(new  LineGeometry(new
Point(i * 10, 0), new Point(0, i * 10)));
            }

            GeometryDrawing      geometryDrawing      =      new
GeometryDrawing
            {
                Brush = Brushes.Black,
                Pen = new Pen(Brushes.Black, 1),
                Geometry = geometryGroup
            };
            return new DrawingBrush(geometryDrawing);
```

```
        }
    private void Traek_Click(object sender, RoutedEventArgs e)
    {
        double x0, v0, x1, v1, xL, vL, t0, t1, xx; string txt =
"";
        Parameters();

        #region <calc>

        t1 = 0; x1 = Ksi_Ini; v1 = -0.000; xL = x1; vL = v1;
        txt = string.Format("{0:F4}", t1).PadLeft(7) +
              string.Format("{0:F6}", x1).PadLeft(14) +
              string.Format("{0:F6}", v1).PadLeft(11) +
              string.Format("{0:F6}", xL).PadLeft(14) +
              string.Format("{0:F6}", vL).PadLeft(11) +
              string.Format("{0:F6}",   Psi_Ini).PadLeft(14)   +
"\r\n";

        Ksi_Anl = new float[Mt]; dKsi_Anl = new float[Mt];
        Ksi_Num = new float[Mt]; dKsi_Num = new float[Mt];
        Time = new float[Mt];

        Phi_Anl = new float[Mt]; Phi_Num = new float[Mt];
        Psi_Anl = new float[Mt]; Psi_Num = new float[Mt];

        Ksi_Anl[0] = (float)Ksi_Ini; dKsi_Anl[0] = 0;
        Ksi_Num[0] = (float)Ksi_Ini; dKsi_Num[0] = 0; Time[0] =
0;
        Phi_Anl[0] = fgrad * Ksi_Anl[0] / (float)r;
        Phi_Num[0] = fgrad * Ksi_Num[0] / (float)r;
        Psi_Anl[0] = fgrad * (float)PSI(Ksi_Ini);
        Psi_Num[0] = fgrad * (float)PSI(Ksi_Ini);

        chart.Series.Clear();
        toggleKsi.IsChecked = true;

        for (int i = 1; i <= Nt; i++)
        {
            t0 = t1; x0 = x1; v0 = v1; t1 = t0 + t_step;

            if (toggleKsi.IsChecked == true) RK2(t0, x0, v0, t1,
ref x1, ref v1, eps, fLinear);
            RK2(t0, x0, v0, t1, ref x1, ref v1, eps, nonLinear);

            xL = Ksi_Ini * Math.Cos(omega * t1);
            vL = -omega * Ksi_Ini * Math.Sin(omega * t1);
```

```
            xx = PSI(x1) * grad;

            txt += string.Format("{0:F4}", t1).PadLeft(7) +
                   string.Format("{0:F6}", x1).PadLeft(14) +
                   string.Format("{0:F6}", v1).PadLeft(11) +
                   string.Format("{0:F6}", xL).PadLeft(14) +
                   string.Format("{0:F6}", vL).PadLeft(11) +
                   string.Format("{0:F6}",   xx).PadLeft(14)   +
"\r\n";

            Ksi_Anl[i] = (float)xL; dKsi_Anl[i] = (float)vL;
            Ksi_Num[i] = (float)x1; dKsi_Num[i] = (float)v1;

            Phi_Anl[i]  =  fgrad  *  Ksi_Anl[i]  /  (float)r;
Phi_Num[i] = fgrad * Ksi_Num[i] / (float)r;
            Psi_Anl[i] = fgrad * (float)PSI(xL); Psi_Num[i] =
fgrad * (float)PSI(x1);

            Time[i] = (float)t1;
     }
         #endregion <calc>

         #region <Mechanics.txt>
         StreamWriter SW = new StreamWriter("Mechanics.txt");

            txt = string.Format(" Маса тіла 1  {0:F3} кг", m1);
SW.WriteLine(txt);
            txt = string.Format(" Маса тіла 2  {0:F3} кг", m2);
SW.WriteLine(txt);
            txt = string.Format(" Маса тіла 3  {0:F3} кг", m3);
SW.WriteLine(txt);
            SW.WriteLine("");
            txt = string.Format(" Радіус r тіла 2  {0:F2} м", r);
SW.WriteLine(txt);
            txt = string.Format(" Радіус R тіла 2  {0:F2} м", R);
SW.WriteLine(txt);
            txt = string.Format(" Довжина L тіла 3 {0:F2} м", L);
SW.WriteLine(txt);
            txt = string.Format(" Довжина AB       {0:F2} м", H);
SW.WriteLine(txt);
            SW.WriteLine("");
            txt = string.Format("  Жорсткість пружини  {0:F0} ",
C); SW.WriteLine(txt);
            txt = string.Format("    Довжина пружини  {0:F2} ",
Dc); SW.WriteLine(txt);
```

```
            txt = string.Format(" Деформація пружини  {0:F2} ",
Ds); SW.WriteLine(txt);
            txt = string.Format("    Відношення Lo/L  {0:F2} ",
n3); SW.WriteLine(txt);
            SW.WriteLine("");
            txt  =  string.Format("  Початкова  умова  {0:F3}  ",
Ksi_Ini); SW.WriteLine(txt);
            SW.WriteLine("");
            SW.WriteLine("   t          ksi      dksi         phi
psi");
            for (int i = 0; i <= Nt; i++)
            {
                txt          =          string.Format("{0:F4}",
Time[i]).PadLeft(7) +
                           string.Format("{0:F6}",
Ksi_Num[i]).PadLeft(14) +
                           string.Format("{0:F6}",
dKsi_Num[i]).PadLeft(11) +
                           string.Format("{0:F6}",
Phi_Num[i]).PadLeft(14) +
                           string.Format("{0:F6}",
Psi_Num[i]).PadLeft(11);
                SW.WriteLine(txt);
            }
            SW.Close();

        #endregion <Mechanics.txt>

        #region <graph>
        Graph(Ksi_Num, Ksi_Anl, 0.05f);
        #endregion <graph>

    }
    private void RK2(double t0, double x0, double v0, double t1,
ref double x1,
    ref double v1, double eps, F_t_x_v f)
    {
        double k1, k2, k3, k4, ht, error, t, x, v, tj, xj, vj;
        int j, M, it = 0;
        M = 2 * (int)Math.Ceiling(t1 - t0);
        x1 = 1.0E10; v1 = 1.0E10; x = 0.0; v = 0.0;
        do
        {
            ht = (t1 - t0) / M; it++;
            t = t0; x = x0; v = v0;
            for (j = 1; j <= M; j++)
```

```
            {
                tj = t;
                xj = x;
                vj = v;
                k1 = ht * f(t, x, v);

                t = tj + ht / 2;
                x = xj + ht * (vj + k1 / 4) / 2;
                v = vj + k1 / 2;
                k2 = ht * f(t, x, v);

                v = vj + k2 / 2;
                k3 = ht * f(t, x, v);

                t = tj + ht;
                x = xj + ht * (vj + k3 / 2);
                v = vj + k3;
                k4 = ht * f(t, x, v);

                x = xj + ht * (vj + (k1 + k2 + k3) / 6);
                v = vj + (k1 + 2 * (k2 + k3) + k4) / 6;
            }
            error = Math.Sqrt((x1 - x) * (x1 - x) + (v1 - v) *
(v1 - v));
            x1 = x; v1 = v; M = M * 2;
            if (it > 5)
            {
                Raschet.Content = string.Format("it = {0}   M =
{1}", it, M);
                Raschet.Visibility = Visibility.Visible;
            }
        }
        while ((error > eps) && (it < 14));
        return;
    }

    private void Clear_Click(object sender, RoutedEventArgs e)
    {
        toggleKsi.IsChecked = false;
        toggleFi.IsChecked = false;
        togglePsi.IsChecked = false;
        chart.Series.Clear();

    }
    private void UpdateYAxis(float[] data1, float[] data2, float
step)
```

```
    {
        float minY, maxY;
        minY = Math.Min(data1.Min(), data2.Min());
        maxY = Math.Max(data1.Max(), data2.Max());
        if (Math.Min(data1.Min(), data2.Min()) < 0 ||
Math.Max(data1.Max(), data2.Max()) < 0)
        {
            minY = -0.5f;
            maxY = 0.5f;
        }

        Axis axisY1 = new Axis
        {
            Title = "Axis Y",
            MinValue = Math.Round(minY,1),
            MaxValue = Math.Round(maxY,1),
            Separator = new LiveCharts.Wpf.Separator { Step =
step, IsEnabled = true }
        };
        chart.AxisY.Clear();
        chart.AxisY.Add(axisY1);
        chart.Update();
    }

    public void Graph(float[] data1, float[] data2, float step)
    {

        // Перевіряємо, що масиви ініціалізовані та мають
однакову довжину
        if (Time == null || data1 == null || data2 == null ||
            Time.Length != data1.Length || Time.Length !=
data2.Length)
        {
            throw new InvalidOperationException("Данні масиву
неініціалізовані або їх довжини не співпадають");
        }
        UpdateYAxis(data1, data2, step);
        // Створення та налаштування серії для sxN
        var sxNSeries = new LineSeries
        {
            Title = null,
            Values = new ChartValues<ObservablePoint>(),
            Stroke = new SolidColorBrush(Colors.Red),
            Fill = Brushes.Transparent,
            StrokeThickness = 2,
            PointGeometry = null,
```

```
            LineSmoothness = 1 // 1 - повністю згладжена лінія
(Spline)
        };

        // sxN
        for (int i = 0; i <= Nt; i++)
        {
            sxNSeries.Values.Add(new    ObservablePoint(Time[i],
data1[i]));
        }
        chart.Series.Add(sxNSeries);

        // sxA
        var sxASeries = new LineSeries
        {
            Title = null,
            Values = new ChartValues<ObservablePoint>(),
            Stroke = new SolidColorBrush(Colors.Green),
            Fill = Brushes.Transparent,
            StrokeThickness = 2,
            PointGeometry = null,
            LineSmoothness = 1
        };

        for (int i = 0; i <= Nt; i++)
        {
            sxASeries.Values.Add(new    ObservablePoint(Time[i],
data2[i]));
        }
        chart.Series.Add(sxASeries);

        // Создание и настройка серии для s0
        var s0Series = new LineSeries
        {
            Title = null,
            Values = new ChartValues<ObservablePoint>
        {
            new ObservablePoint(0.0, 0.0),
            new ObservablePoint(Time[Nt], 0.0)
        },
            Stroke = new SolidColorBrush(Colors.DarkGray),
            Fill = Brushes.Transparent,
            StrokeThickness = 1,
            PointGeometry = null,
            LineSmoothness = 0.0
        };
```

```
            chart.Series.Add(s0Series);
        }

        private      void      toggleKsi_Checked(object      sender,
RoutedEventArgs e)
        {
            toggleFi.IsChecked = false;
            togglePsi.IsChecked = false;
            Graph(Ksi_Num, Ksi_Anl, 0.05f);
        }
        private     void     toggleKsi_Unchecked(object     sender,
RoutedEventArgs e)
        {
            chart.Series.Clear();
        }

        private void toggleFi_Checked(object sender, RoutedEventArgs
e)
        {
            toggleKsi.IsChecked = false;
            togglePsi.IsChecked = false;
            Graph(Phi_Num, Phi_Anl, 5.0f);
        }

        private      void      toggleFi_Unchecked(object      sender,
RoutedEventArgs e)
        {
            chart.Series.Clear();
        }

        private      void      togglePsi_Checked(object      sender,
RoutedEventArgs e)
        {
            toggleKsi.IsChecked = false;
            toggleFi.IsChecked = false;
            Graph(Psi_Num, Psi_Anl,3.0f);
        }

        private     void     togglePsi_Unchecked(object     sender,
RoutedEventArgs e)
        {
            chart.Series.Clear();
        }
        public void Parameters()
           {
               double alfa, O2O3;
```

```
            Lbl_weight1.Content = "m1 =  " + string.Format("{0:F3}
кг", m1);
            Lbl_weight2.Content = "m2 =  " + string.Format("{0:F3}
кг", m2);
            Lbl_weight3.Content = "m3 =  " + string.Format("{0:F3}
кг", m3);

            Lbl_sizer.Content = " r =  " + string.Format("{0:F2}
м", r);
            Lbl_sizeR.Content = " R =  " + string.Format("{0:F2}
м", R);
            Lbl_sizeL.Content = " L =  " + string.Format("{0:F2}
м", L);
            Lbl_sizeH.Content = " H =  " + string.Format("{0:F2}
м", H);

            r2 = r * r; R2 = R * R; L2 = L * L; Lc = n3 * L;

            Ds = (m1 * g / C) * (r / R) * (L / Lc);     // формула
(25)
            DcDs = Dc + Ds;

            Lbl_springC.Content = " C =  " + string.Format("{0:F0}
Н/м", C);
            Lbl_springDc.Content    =    "    Dc    =        "    +
string.Format("{0:F2}  м", Dc);
            Lbl_springn3.Content    =    "    n3    =        "    +
string.Format("{0:F2}   ", n3);
            Lbl_springDs.Content    =    "    Ds    =        "    +
string.Format("{0:F2}  м", Ds);

            if ((nC * Dc) < Ds)
            {
                string err = " Статична деформація пружини  " +
                                string.Format("{0:F4}    (м)\r\n",
Ds) +
                                "перевищує допустиме значення  " +
                                string.Format("{0:F4} (м) !", Dc *
nC);
                MessageBox.Show(err,  "          Помилка  даних",
MessageBoxButton.OK, MessageBoxImage.Error);
            }

            J1 = m1 * r2;                    // формула (11)
            J2 = (m2 / 2.0) * (r2 * r2 + R2 * R2) / (r2 + R2);
```

```
            J3 = m3 * L2 / 3.0;

            M1 = J1 + J2 + J3 * R2 / L2;    // формула (32)
            sigma = (R / r) * (Lc / L);
            M2 = r2 * (C * sigma * sigma + 0.5 * m3 * g * L / r);
            omega2 = M2 / M1; omega = Math.Sqrt(omega2);

            period_L = 2.0 * Math.PI / omega;

            Period.Text = string.Format("{0:F4}", period_L);

            t_full = 2.0 * period_L; t_step = t_full / Nt;


            O2O3 = Math.Sqrt(H * H + (L - R) * (L - R));
            alfa = Math.Acos((L - R) / O2O3);
            Psi_Max = Math.Acos((L * L - R * (L + H)) / L / O2O3)
- alfa;

            TBlock_Psi.Text = string.Format(" = {0:F2} гр", grad
* Psi_Max);


            Phi_Max = Math.Acos((R * (L - H) - H * H - R * R) /
(H + R) / O2O3) - alfa;

            Ksi_Max = Phi_Max * r;

            TBlock_Fi.Text = string.Format(" = {0:F2} гр", grad *
Phi_Max);
            TBlock_Ksi.Text  =  string.Format("  =  {0:F3}  м",
Ksi_Max);



            Ksi_Max = Math.Floor(Ksi_Max * 100 - 3) / 100.0;

            if (start) Ksi_Ini = 0.5 * Ksi_Max;

            if (Ksi_Ini > Ksi_Max) Ksi_Ini = Ksi_Max;

            TBlock_Ksi_0.Text = " = " + string.Format("{0:F3}  м",
Ksi_Ini);

            Phi_Ini = (Ksi_Ini / r) * grad;
```

```
            TBlock_Fi_0.Text = " = " + string.Format("{0:F2} гр",
Phi_Ini);

            Psi_Ini = PSI(Ksi_Ini) * grad;
            TBlock_Psi_0.Text = " = " + string.Format("{0:F2} гр",
Psi_Ini);
        }
        public static double PSI(double ksi)
        {
            double a, b, c, fa, fc, fb; int k;

            if (ksi == 0) return 0;
            k = 0;
            if (ksi > 0)
            {
                a = 0; b = Psi_Max; fa = f(ksi, a);
                do
                {
                    c = (a + b) * 0.5; fc = f(ksi, c); k++;
                    if ((fa * fc) < 0) b = c; else a = c;
                    if ((b - a) < eps) break;
                }
                while (Math.Abs(fc) > eps);
            }
            else
            {
                a = -Psi_Max; b = 0; fb = f(ksi, b);
                do
                {
                    c = (a + b) * 0.5; fc = f(ksi, c); k++;
                    if ((fb * fc) < 0) a = c; else b = c;
                    if ((b - a) < eps) break;
                }
                while (Math.Abs(fc) > eps);
            }
            return c;
        }

        public static double KSI(double psi)
        {
            double d_ksi = 0.001, ksi = Ksi_Max, fa, fb = f(ksi,
psi);
            do
            {
                d_ksi = -d_ksi / 10;
                do
```

```
                {
                    fa = fb; ksi += d_ksi; fb = f(ksi, psi);
                }
                while (fa > fb);
            } while (fb > eps);
            return ksi;
        }

        public static double f(double ksi, double psi)
        {
            double phi = ksi / r;
            double x = R * (1.0 - Math.Cos(phi)) - L * (1.0 -
Math.Cos(psi));
            double y = H + L * Math.Sin(psi) - R * Math.Sin(phi);
            return x * x + y * y - H * H;
        }
        public static double fLinear(double t, double x, double v)
        {
            return -omega2 * x;
        }
        public static double nonLinear(double t, double x, double
v)
        {
            double ksi, phi, psi, X_psi, X_phi, Z_psi, Z_phi, XZ,
ED1;
            double X, Z, a, b, u1, u2, u3, u;

            ksi = x;

            // кути  phi  и  psi

            phi = ksi / r;
            psi = PSI(ksi);

            double S_phi = Math.Sin(phi), S_psi = Math.Sin(psi),
                   C_phi = Math.Cos(phi), C_psi = Math.Cos(psi),
                   C_phs = Math.Cos(phi - psi);

            // формула (6)

            a = R * (1.0 - C_phi) - L * (1.0 - C_psi);
            b = H + L * S_psi - R * S_phi;

            X = 2.0 * R * (a * S_phi - b * C_phi);
            Z = 2.0 * L * (a * S_psi - b * C_psi);
```

```
            XZ = X / Z;

            // формула (19)

            X_phi = 2.0 * R * ((R - L) * C_phi + H * S_phi + L *
C_phs);
            X_psi = -2.0 * R * L * C_phs;
            Z_phi = 2.0 * R * L * C_phs;
            Z_psi = 2.0 * L * ((R - L) * C_psi + H * S_psi - R *
C_phs);

            u1 = J3 * XZ * (X * (Z_phi + XZ * Z_psi) - Z * (X_phi
+ XZ * X_psi)) / Z / Z / r2 / r;

            u2 = m1 * g - 0.5 * m3 * g * S_psi * XZ * L / r;

            ED1 = Math.Sqrt(DcDs * DcDs + 2.0 * Lc * Lc - 2.0 * Lc
* (Lc * C_psi - DcDs * S_psi));
            u3 = C * (ED1 - Dc) / ED1 * (DcDs * C_psi + Lc * S_psi)
* Lc * XZ / r;

            u = (J1 + J2 + J3 * XZ * XZ) / r2;

            return (u1 * v * v + u2 - u3) / u;
        }
 }
```

ДОДАТОК В
Вікно для зміни мас тіл механічної системи

```
<Window x:Class="DiplomDA.massa"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:local="clr-namespace:DiplomDA"
        mc:Ignorable="d"
       Height="380" Width="725" Background="#a0b0a6">
    <Grid >
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>

        </Grid.RowDefinitions>
        <GroupBox x:Name="m1" Grid.Row="0"
          HorizontalAlignment="Left" Height="60"
          Header=" Маса вантажу 1  " FontSize="16"
          VerticalAlignment="Top" Width="715">
            <Slider x:Name="Bar_m1" Height="50"
TickPlacement="BottomRight" ValueChanged="Bar_m1_ValueChanged"
SmallChange="0.5"/>
        </GroupBox>
        <GroupBox x:Name="m2" Grid.Row="1"
   HorizontalAlignment="Left" Height="60"
   Header="Маса блоку 2" FontSize="16"
  VerticalAlignment="Top" Width="715">
            <Slider x:Name="Bar_m2" Height="50"
TickPlacement="BottomRight" ValueChanged="Bar_m2_ValueChanged"
SmallChange="0.5"/>
        </GroupBox>
        <GroupBox x:Name="m3" Grid.Row="2"
   HorizontalAlignment="Left" Height="60"
   Header="Маса стрижня 3 " FontSize="16"
    VerticalAlignment="Top" Width="715">
            <Slider x:Name="Bar_m3" Height="50"
TickPlacement="BottomRight" ValueChanged="Bar_m3_ValueChanged"
SmallChange="0.5"/>
        </GroupBox>
```

```
        <Button  Grid.Row="3" Content="OK"
HorizontalAlignment="Center" Height="50" VerticalAlignment="Top"
Width="170" Click="Button_Click" Margin="0,14,0,0"/>
        <Button Grid.Row="3" Content="Відміна"
HorizontalAlignment="Left" Height="50" VerticalAlignment="Top"
Width="170" Click="Button_Click_1" Margin="545,14,0,0"/>
    </Grid>
</Window>
```

ДОДАТОК Г
Лістинг програмної реалізації класу massa

```
public partial class massa : Window
 {
     private static double m1_min, m1_max, m11;
     private static double m2_min, m2_max, m22;
     private static double m3_min, m3_max, m33;
     private static double d_stat, d, d_max;


     private void Button_Click_1(object sender, RoutedEventArgs
e)
     {
         this.Close();
     }

     private void Button_Click(object sender, RoutedEventArgs e)
     {
         MainWindow.m1 = m11; MainWindow.m2 = m22; MainWindow.m3
= m33;
         MainWindow.clean = true; this.Close();
     }

     private void Bar_m1_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
     {
         int m = Convert.ToInt32(Bar_m1.Value);
         m11 = m1_min + m * (m1_max - m1_min) / Bar_m1.Maximum;
         m1.Header =  string.Format("{0:F3}  кг", m11);

         d_stat = m11 * d;
     }

     private void Bar_m2_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
     {
         int m = Convert.ToInt32(Bar_m2.Value);
         m22 = m2_min + m * (m2_max - m2_min) / Bar_m2.Maximum;
         m2.Header = string.Format("{0:F3}  кг", m22);
     }

     private void Bar_m3_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
     {
```

```
            int m = Convert.ToInt32(Bar_m3.Value);
            m33 = m3_min + m * (m3_max - m3_min) / Bar_m3.Maximum;
            m3.Header = string.Format("{0:F3}  кг", m33);
        }
        private static double L_min, L_max, L1;
        private static double H_min, H_max, H1;

        public massa()
        {
            InitializeComponent();

            m1_min = 2.0; m1_max = 7.0;
            m11 = MainWindow.m1;
            Bar_m1.Maximum = (int)(100 * m1_max - 100 * m1_min);
            Bar_m1.Value = (int)(Bar_m1.Maximum * (m11 - m1_min) /
(m1_max - m1_min));
            m1.Header = string.Format("{0:F3}  кг", m11);

            m2_min = 1.0; m2_max = 3.0;
            m22 = MainWindow.m2;
            Bar_m2.Maximum = (int)(100 * m2_max - 100 * m2_min);
            Bar_m2.Value = (int)(Bar_m2.Maximum * (m22 - m2_min) /
(m2_max - m2_min));
            m2.Header = string.Format("{0:F3}  кг", m22);

            m3_min = 3.0; m3_max = 9.0;
            m33 = MainWindow.m3;
            Bar_m3.Maximum = (int)(100 * m3_max - 100 * m3_min);
            Bar_m3.Value = (int)(Bar_m3.Maximum * (m33 - m3_min) /
(m3_max - m3_min));
            m3.Header = string.Format("{0:F3}  кг", m33);

            d = MainWindow  .g * MainWindow.r * MainWindow.L /
(MainWindow.C * MainWindow.R * MainWindow.Lc);
            d_stat = MainWindow.m1 * d;
            d_max = 0.15 * MainWindow.Dc;
        }
    }
```

ДОДАТОК Д
Додаткове вікно зміни початкових умов

```
<Window x:Class="DiplomDA.first"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:local="clr-namespace:DiplomDA"
        mc:Ignorable="d"
         Height="340" Width="825" Background="#a0b0a6">
    <Grid >
        <GroupBox x:Name="gruz"
                  HorizontalAlignment="Left" Height="185"
                  Header="Початкова умова вантажу 1"
FontSize="16"
                  Margin="60,59,0,0" VerticalAlignment="Top"
Width="715">
            <Slider x:Name="Bar_ksi" Height="50"
TickFrequency="10" TickPlacement="BottomRight"
ValueChanged="Bar_ksi_ValueChanged" >

            </Slider>
        </GroupBox>
        <Button Content="OK" HorizontalAlignment="Left"
Height="50" Margin="375,249,0,0" VerticalAlignment="Top"
Width="170" Click="Button_Click"/>
        <Button Content="Відміна" HorizontalAlignment="Left"
Height="50" Margin="590,249,0,0" VerticalAlignment="Top"
Width="170" Click="Button_Click_1"/>

    </Grid>
</Window>
```

ДОДАТОК Е
Лістинг програмної реалізації класу first

```
 public partial class first : Window
  {
      private static double ksi_min, ksi_max, ksi_init, phi;

      private void Button_Click_1(object sender, RoutedEventArgs
e)
      {
          this.Close();
      }

      private void Button_Click(object sender, RoutedEventArgs
e)
      {
          MainWindow.Ksi_Ini = ksi_init; this.Close();
      }

      private void Bar_ksi_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
      {
          int m = Convert.ToInt32(Bar_ksi.Value);
          ksi_init = ksi_min + m * (ksi_max - ksi_min) /
Bar_ksi.Maximum;

          phi = (ksi_init / MainWindow.r) * MainWindow.grad;
      }

      public first()
      {
          InitializeComponent();
          ksi_min = -MainWindow.Ksi_Max; ksi_max =
MainWindow.Ksi_Max; ksi_init = MainWindow.Ksi_Ini;
          Bar_ksi.Maximum = (int)(100 * ksi_max - 100 *
ksi_min);
          Bar_ksi.Value = (int)(Bar_ksi.Maximum * (ksi_init -
ksi_min) / (ksi_max - ksi_min));
          //gruz.Header =  string.Format("{0:F2}  м", ksi_init);
          phi = (ksi_init / MainWindow.r) * MainWindow.grad;

      }
  }
```

ДОДАТОК Ж
Додаткове вікно зміни параметрів пружини

```
<Window x:Class="DiplomDA.zigzag"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:local="clr-namespace:DiplomDA"
        mc:Ignorable="d"
        Height="380" Width="725" Background="#a0b0a6">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>

        </Grid.RowDefinitions>
        <GroupBox x:Name="C" Grid.Row="0"
        HorizontalAlignment="Left" Height="60"
        Header="Коефіцієнт відновлення C " FontSize="16"
        VerticalAlignment="Top" Width="715">
            <Slider x:Name="Bar_C" Height="50"
TickPlacement="BottomRight" ValueChanged="Bar_C_ValueChanged"/>
        </GroupBox>
        <GroupBox x:Name="Dc" Grid.Row="1"
 HorizontalAlignment="Left" Height="60"
 Header="Довжина недеформованої пружини " FontSize="16"
VerticalAlignment="Top" Width="715">
            <Slider x:Name="Bar_Dc" Height="50"
TickPlacement="BottomRight" ValueChanged="Bar_Dc_ValueChanged"/>
        </GroupBox>
        <GroupBox x:Name="n3" Grid.Row="2"
 HorizontalAlignment="Left" Height="60"
 Header="n3 " FontSize="16"
  VerticalAlignment="Top" Width="715">
            <Slider x:Name="Bar_n3" Height="50"
TickPlacement="BottomRight" ValueChanged="Bar_n3_ValueChanged"/>
        </GroupBox>
```

```
        <Button  Grid.Row="3" Content="OK"
HorizontalAlignment="Center" Height="50" VerticalAlignment="Top"
Width="170" Click="Button_Click" Margin="0,14,0,0"/>
        <Button Grid.Row="3" Content="Відміна"
HorizontalAlignment="Left" Height="50" VerticalAlignment="Top"
Width="170" Click="Button_Click_1" Margin="545,14,0,0"/>
    </Grid>
</Window>
```

ДОДАТОК К
Лістинг програмної реалізації класу zigzag

```
            public partial class zigzag : Window
{
    private static double C_min, C_max, Cc;
    private static double dc_min, dc_max, d_coil;
    private static double n3_min, n3_max, n33;

    private void Button_Click_1(object sender, RoutedEventArgs
e)
    {
        this.Close();
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        MainWindow.C = Cc; MainWindow.Dc = d_coil; MainWindow.n3
= n33;
        MainWindow.clean = true; this.Close();
    }

    private static double d, d_stat, d_max;

    private void Bar_Dc_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
    {
        int m = Convert.ToInt32(Bar_Dc.Value);
        d_coil = dc_min + m * (dc_max - dc_min) /
Bar_Dc.Maximum;
        Dc.Header = string.Format("{0:F2}  м", d_coil);

        d_stat = d / Cc / n33; d_max = 0.15 * d_coil;
    }

    private void Bar_n3_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
    {
        int m = Convert.ToInt32(Bar_n3.Value);
        n33 = n3_min + m * (n3_max - n3_min) / Bar_n3.Maximum;
        n3.Header = string.Format("{0:F2}", n33);

        d_stat = d / Cc / n33; d_max = 0.15 * d_coil;
    }
```

```
    private void Bar_C_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
    {
        int m = Convert.ToInt32(Bar_C.Value);
        Cc = C_min + m * (C_max - C_min) / Bar_C.Maximum;
        C.Header = string.Format("{0:F1}  Н/м", Cc);

        d_stat = d / Cc / n33; d_max = 0.15 * d_coil;
    }


    public zigzag()
    {
        InitializeComponent();
        C_min = 100.0; C_max = 500.0;
        Cc = MainWindow.C;
        Bar_C.Maximum = (int)(1 * C_max - 1 * C_min);
        Bar_C.Value = (int)(Bar_C.Maximum * (Cc - C_min) /
(C_max - C_min));
        C.Header = string.Format("{0:F1}  Н/м", Cc);

        dc_min = 1.00; dc_max = 2.00;
        d_coil = MainWindow.Dc;
        Bar_Dc.Maximum = (int)(100 * dc_max - 100 * dc_min);
        Bar_Dc.Value = (int)(Bar_Dc.Maximum * (d_coil - dc_min)
/ (dc_max - dc_min));
        Dc.Header = string.Format("{0:F2}  м", d_coil);

        n3_min = 0.60; n3_max = 0.90;
        n33 = MainWindow.n3;
        Bar_n3.Maximum = (int)(100 * n3_max - 100 * n3_min);
        Bar_n3.Value = (int)(Bar_n3.Maximum * (n33 - n3_min) /
(n3_max - n3_min));
        n3.Header = string.Format("{0:F2}", n33);

        d = MainWindow.m1 * MainWindow.g * MainWindow.r /
MainWindow.R;
        d_stat = d / Cc / n33;
        d_max = MainWindow.nC * MainWindow.Dc;
    }
}
```

ДОДАТОК Л
Додаткове вікно зміни розмірів тіл

```
<Window x:Class="DiplomDA.size"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:local="clr-namespace:DiplomDA"
        mc:Ignorable="d"
               Height="455" Width="725" Background="#a0b0a6">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <GroupBox x:Name="r1" Grid.Row="0"
          HorizontalAlignment="Left" Height="60"
          Header="Внутрішній радіус r " FontSize="16"
          VerticalAlignment="Top" Width="715">
            <Slider x:Name="Bar_r1" Height="50"
TickPlacement="BottomRight" ValueChanged="Bar_r1_ValueChanged"/>
        </GroupBox>
        <GroupBox x:Name="r2" Grid.Row="1"
   HorizontalAlignment="Left" Height="60"
   Header="Зовнішній радіус R " FontSize="16"
  VerticalAlignment="Top" Width="715">
            <Slider x:Name="Bar_r2" Height="50"
TickPlacement="BottomRight" ValueChanged="Bar_r2_ValueChanged"/>
        </GroupBox>
        <GroupBox x:Name="L" Grid.Row="2"
   HorizontalAlignment="Left" Height="60"
   Header="Довжина L стрижня 3 " FontSize="16"
    VerticalAlignment="Top" Width="715">
            <Slider x:Name="Bar_L" Height="50"
TickPlacement="BottomRight" ValueChanged="Bar_L_ValueChanged"/>
        </GroupBox>
        <GroupBox x:Name="H" Grid.Row="3"
```

```
   HorizontalAlignment="Left" Height="60"
   Header="Довжина H стрижня AB" FontSize="16"
   VerticalAlignment="Top" Width="715">
            <Slider x:Name="Bar_H" Height="50"
TickPlacement="BottomRight" ValueChanged="Bar_H_ValueChanged"/>
        </GroupBox>

        <Button  Grid.Row="4" Content="OK"
HorizontalAlignment="Center" Height="50" VerticalAlignment="Top"
Width="170" Click="Button_Click" Margin="0,14,0,0"/>
        <Button Grid.Row="4" Content="Відміна"
HorizontalAlignment="Left" Height="50" VerticalAlignment="Top"
Width="170" Click="Button_Click_1" Margin="545,14,0,0"/>
    </Grid>
</Window>
```

ДОДАТОК М
Лістинг програмної реалізації класу zigzag

```
  public partial class size : Window
  {
      private static double r_min, r_max, r;
      private static double R_min, R_max, R;

      private void Button_Click_1(object sender, RoutedEventArgs
e)
      {
          this.Close();
      }

      private void Button_Click(object sender, RoutedEventArgs
e)
      {
          MainWindow.r = r; MainWindow.R = R; MainWindow.L = L1;
MainWindow.H = H1;
          MainWindow.clean = true; this.Close();
      }

      private void Bar_H_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
      {
          int m = Convert.ToInt32(Bar_H.Value);
          H1 = H_min + m * (H_max - H_min) / Bar_H.Maximum;
          H.Header = string.Format("{0:F2}  м", H1);
      }

      private void Bar_L_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
      {
          int m = Convert.ToInt32(Bar_L.Value);
          L1 = L_min + m * (L_max - L_min) / Bar_L.Maximum;
          L.Header = string.Format("{0:F2}  м", L1);
      }

      private void Bar_r2_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
      {
          int m = Convert.ToInt32(Bar_r2.Value);
          R = R_min + m * (R_max - R_min) / Bar_r2.Maximum;
          r2.Header = string.Format("{0:F2}  м", R);

          d_stat = d * r / R;
```

```
        }

        private void Bar_r1_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            int m = Convert.ToInt32(Bar_r1.Value);
            r = r_min + m * (r_max - r_min) / Bar_r1.Maximum;
            r1.Header = string.Format("{0:F2}  м", r);

            d_stat = d * r / R;
        }

        private static double L_min, L_max, L1;
        private static double H_min, H_max, H1;
        private static double d, d_stat, d_max;
        public size()
        {
            InitializeComponent();

            r_min = 0.25; r_max = 0.75;
            r = MainWindow.r;
            Bar_r1.Maximum = (int)(100 * r_max - 100 * r_min);
            Bar_r1.Value = (int)(Bar_r1.Maximum * (r - r_min) /
(r_max - r_min));
            r1.Header =  string.Format("{0:F2}  м", r);

            R_min = 1.00; R_max = 2.00;
            R = MainWindow.R;
            Bar_r2.Maximum = (int)(100 * R_max - 100 * R_min);
            Bar_r2.Value = (int)(Bar_r2.Maximum * (R - R_min) /
(R_max - R_min));
            r2.Header = string.Format("{0:F2}  м", R);

            L_min = 2.00; L_max = 5.00;
            L1 = MainWindow.L;
            Bar_L.Maximum = (int)(100 * L_max - 100 * L_min);
            Bar_L.Value = (int)(Bar_L.Maximum * (L1 - L_min) /
(L_max - L_min));
            L.Header = string.Format("{0:F2}  м", L1);

            H_min = 3.00; H_max = 9.00;
            H1 = MainWindow.H;
            Bar_H.Maximum = (int)(100 * H_max - 100 * H_min);
            Bar_H.Value = (int)(Bar_H.Maximum * (H1 - H_min) /
(H_max - H_min));
            H.Header = string.Format("{0:F2}  м", H1);
```

```
            d = MainWindow.g * MainWindow.m1 * MainWindow.L /
MainWindow.C / MainWindow.Lc;
            d_stat = d * MainWindow.r / MainWindow.R;
            d_max = MainWindow.nC * MainWindow.Dc;
        }
    }
```

ДОДАТОК Н
Лістинг програмної реалізації класу VisualHost

```
   public class VisualHost : FrameworkElement
{
    private VisualCollection visuals;

    public VisualHost()
    {
        visuals = new VisualCollection(this);
    }

    public void AddVisual(Visual visual)
    {
        visuals.Add(visual);
    }

    protected override int VisualChildrenCount => visuals.Count;

    protected override Visual GetVisualChild(int index)
    {
        return visuals[index];
    }
}
```