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»

«Візуалізація коливань подвійного математичного маятника в середовищі Microsoft Visual Studio»

«Visualization of oscillations of a double mathematical pendulum in the Microsoft Visual Studio environment»

Виконав: здобувач денної (очної) форми навчання

спеціальності 126 Інформаційні системи та технології

Освітня програма Інформаційні системи та технології

Савельєв Єгор Дмитрович
(прізвище, ім'я, по-батькові здобувача)

Керівник ст. викладач Косирева Л.А. (підпис)
(науковий ступінь, вчене звання, прізвище, ініціали)

Рецензент викладач Палій К.С.
(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
_____ ОЛЕНА АРСІРІЙ
(підпис)

Одеса 2025

АНОТАЦІЯ

У кваліфікаційній роботі розробляється тема «Візуалізація коливань подвійного математичного маятника в середовищі Microsoft Visual Studio».

Робота присвячена комп'ютерному моделюванню коливань подвійного математичного маятника з урахуванням сили опору середовища. Створений Windows-додаток на мові C# в середовищі Microsoft Visual Studio 2022 із застосуванням 2D-графіки, який візуалізує коливання подвійного математичного маятника у режимі віртуального часу.

В роботі отримано систему двох нелінійних диференціальних рівнянь руху маятників, яка згодом вирішувалась чисельно методом Рунге-Кутта IV порядку точності. Для порівняння отримано аналітичне рішення у випадку малих коливань маятників без урахування сили опору для довільного співвідношення мас та довжин маятників.

Створений додаток дозволяє проводити розрахунки та візуалізацію як для випадку кінцевих коливань, так і для випадку малих коливань, будує траєкторію другого маятника та графіки кутових відхилень обох маятників.

Проведено порівняльний аналіз кінцевих та малих коливань, а також аналіз коливань маятників для різних геометричних параметрів.

ABSTRACT

The qualification thesis is dedicated to the topic "Visualization of oscillations of a double mathematical pendulum in the Microsoft Visual Studio environment."

The work focuses on computer modeling of the oscillations of a double pendulum, taking into account the resistance force of the surrounding medium. A Windows application was developed in C# using Microsoft Visual Studio 2022 and 2D graphics to visualize the oscillations of the double pendulum in real time.

A system of two nonlinear differential equations describing the motion of the pendulums was derived and solved numerically using the fourth-order Runge-Kutta method. For comparison, an analytical solution was obtained for the case of small oscillations without considering the resistance force, for arbitrary ratios of masses and pendulum lengths.

The developed application enables both the calculation and visualization of finite and small oscillations. It plots the trajectory of the second pendulum and the angular displacement graphs of both pendulums.

A comparative analysis was carried out between finite and small oscillations, as well as an analysis of the pendulums' behavior for various geometric parameters.

ЗМІСТ

ВСТУП

Подвійний математичний маятник - це, поза сумнівом, справжнє чудо природи. Вражаючий стрибок складності, який спостерігається при переході від простого поодинокого маятника до подвійного. Коливання простого маятника мають регулярний характер. При малих відхиленнях від рівноваги такі коливання є гармонійними і описуються функцією синус або косинус. У разі нелінійних коливань період залежить від амплітуди, але регулярність руху зберігається. Іншими словами, у разі простого маятника наближення малих коливань цілком відбиває істотні властивості системи.

Подвійний математичний маятник поводиться абсолютно інакше. Вже в режимі малих коливань у подвійного математичного маятника виникає таке нове явище, як ефект биття. А при збільшенні енергії характер коливань маятників міняється принципово - коливання стають хаотичними. Попри те, що подвійний маятник можна описати системою декількох звичайних диференціальних рівнянь, поява хаосу виглядає дуже незвично. Картина руху істотно залежить від співвідношення мас маятників.

Актуальність теми зумовлена широким застосуванням моделей з подібною кінематикою у сучасних інженерних задачах, зокрема в галузі робототехніки. Подвійні або багатоланкові маятникові системи можуть бути використані для моделювання шарнірних з’єднань маніпуляторів або інших механізмів із гнучким керуванням. Проведення попереднього комп’ютерного моделювання дозволяє уникнути ризиків, пов’язаних з експериментами на реальному обладнанні, зменшує вартість розробки та дає змогу оперативно тестувати зміну параметрів системи.

Незважаючи на достатню кількість програм, що демонструють поведінку подвійного маятника, більшість із них не враховують вплив сили опору середовища, що істотно знижує точність і реалістичність моделювання. На відміну від них, у межах цієї роботи розроблено Windows-додаток із засобами Microsoft Visual Studio, який дозволяє моделювати рух подвійного

математичного маятника з урахуванням сили опору. Це розширює можливості аналізу та наближує модель до реальних умов.

Метою даної роботи є дослідження динаміки подвійного математичного маятника з використанням програмного моделювання, а також створення візуального інструмента, що дозволяє аналізувати його поведінку в різних режимах.

Для досягнення поставленої мети виконано наступні завдання:

а) побудовано математичну модель подвійного маятника з урахуванням сили опору;

б) реалізовано чисельне розв’язання системи диференціальних рівнянь у середовищі Microsoft Visual Studio;

в) створено графічний інтерфейс користувача для зручного введення параметрів моделі;

г) реалізовано два режими візуалізації: для кінцевих та малих коливань;

д) досліджено вплив початкових умов на характер руху маятника та можливості практичного використання таких спостережень у контексті управління системами з подібною динамікою.

У роботі використаний метод Рунге-Кутта для розв’язання систем звичайних диференціальних рівнянь, а також інструменти об’єктно-орієнтованого програмування у середовищі Microsoft Visual Studio для побудови додатку з можливістю візуалізації результатів у реальному часі.

1. ПОДВІЙНИЙ МАТЕМАТИЧНИЙ МАЯТНИК

1.1 Постановка задачі

Матеріальна точка А маси m_1 підвішена за допомогою стержня довжини l_1 в нерухомій точці О і є математичним маятником, який може коливатися у вертикальній площині. Матеріальна точка В маси m_2, приєднана до точки А за допомогою стержня АВ довжини l_2, може також коливатися в цій же площині. Така конструкція називається *подвійним математичним маятником* [1-2]. Спрощену модель продемонстровано на рис.1.1.Стержні вважатимемо невагомими. Точкові тіла представлені кулями кінцевого радіусу. В точках підвісу тертя відсутнє. Положення точок А і В визначені за допомогою кутів α і φ, відлічуваних від вертикалі.

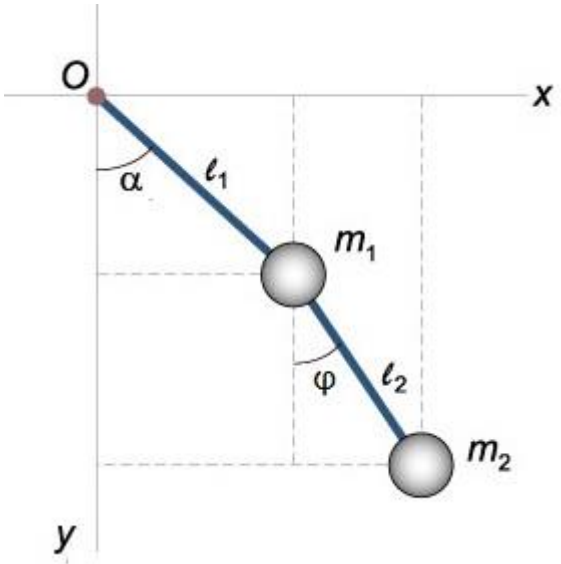

Рисунок 1.1 – Подвійний математичний маятник

Необхідно:

1) Скласти диференційні рівняння руху тіл, що становлять маятники, з урахуванням сили опору середовища, яка пропорційна першому ступеню швидкості.
2) Отримати рішення цих рівнянь для випадку малих коливань тіл без урахування сили опору середовища.
3) Створити Windows-додаток на мові C# в середовищі Microsoft Visual Studio, який відображає візуалізацію руху механізмів на екрані монітора у режимі віртуального часу.
4) Передбачити можливість візуалізації малих коливань маятника.
5) Передбачити можливість побудови траєкторії руху точки B подвійного математичного маятника.
6) Передбачити можливість побудови графіків залежності кутів відхилення маятників від часу.
7) Провести дослідження руху маятника.

1.2 Аналіз аналогів

У межах аналізу існуючих програмних рішень розглянуто два відкриті симулятори руху подвійного математичного маятника: SimuFisica[3] MyPhysicsLab[4]. Обидві системи надають базову функціональність для моделювання поведінки маятника, однак мають низку обмежень з точки зору наукового застосування.

MyPhysicsLab має такі переваги:

а) можливість обирати методи розв’язання диференціальних рівнянь;

б) можливість візуалізації енергії системи.

Проте ця система має суттєві недоліки:

а) відсутній режим малих коливань;

б) не враховується сила опору середовища;

в) відсутня функція побудови траєкторії;

г) немає візуального інтерфейсу для зміни мас маятників.

SimuFisica, у свою чергу, відзначається:

а) великою кількістю графіків залежностей;

б) можливістю зберігати числові дані експерименту;

в) можливістю задавати початкову кутову швидкість для обох маятників.

Однак недоліками цієї реалізації є:

а) відсутність графіку коливань обох вантажів одночасно;

б) неможливість врахування сили опору;

в) відсутність режиму малих коливань.

Таким чином, обидва розглянуті сервіси мають суттєві обмеження, які унеможливлюють їх ефективне використання для повноцінного наукового або прикладного дослідження динаміки подвійного маятника.

Передусім, відсутність моделювання опору середовища є критичним недоліком. У реальних умовах сила опору, зумовлена в'язкістю повітря чи іншим середовищем, істотно впливає на характер коливань — амплітуда зменшується з часом, а частота змінюється. Ігнорування цього чинника призводить до суттєвих розбіжностей між змодельованими результатами та поведінкою реального фізичного об'єкта. На практиці це означає, що симулятори не дозволяють перевірити стійкість або затухання системи, що є обов’язковими аспектами при проектуванні, наприклад, шарнірних вузлів у робототехніці або біомеханічних моделях.

MyPhysicsLab, попри деяку гнучкість у виборі чисельних методів інтегрування, не дає змоги вивчити основні особливості реального процесу. Відсутність функції візуалізації траєкторій і неможливість динамічної зміни мас маятників означає, що дослідник не може перевірити, як співвідношення мас впливає на характер руху — а це один із ключових параметрів у виникненні хаотичної поведінки. Крім того, режим малих коливань відсутній як окремий функціональний блок, що ускладнює перевірку роботи моделі в умовах, де можна отримати аналітичні розв’язки і зробити порівняння.

SimuFisica, хоча і має зручну візуалізацію графіків та дозволяє зберігати числові дані, не забезпечує одночасного відображення траєкторій обох маятників. Це ускладнює оцінку взаємного впливу частин системи та візуальне виявлення фазових зсувів. Також, відсутність врахування сили опору і неможливість налаштування точних умов для синхронного старту обох маятників із фіксованим енергетичним станом істотно обмежує експериментальну гнучкість. Наприклад, при моделюванні систем зі зворотним зв'язком, де потрібно точно задати початкові умови, такі обмеження роблять програму фактично непридатною для роботи.

Загалом, обидва сервіси є радше навчальними демонстраційними інструментами, а не повноцінними інструментами для дослідження чи проектування. Відповідно, їх використання обмежене, а для глибшого дослідження необхідне створення індивідуального програмного інструмента, адаптованого до конкретних задач.

1.3 Виведення системи рівнянь руху подвійного математичного маятника

Для виведення рівнянь руху подвійного математичного маятника використаємо рівняння Лагранжа II роду [1,2]:

$$\begin{cases} \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\alpha}}\right) - \frac{\partial T}{\partial \alpha} = Q_{\alpha}, \\ \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\varphi}}\right) - \frac{\partial T}{\partial \varphi} = Q_{\varphi}. \end{cases} \tag{1.1}$$

Кінетична енергія системи складається з суми кінетичних енергій обох маятників, тобто: $T = T_1 + T_2$.

Тут $T_1 = \frac{m_1 v_1^2}{2} = \frac{m_1 l_1^2 \dot{\alpha}^2}{2}$, $T_2 = \frac{m_2 v_a^2}{2}$,

де $\vec{\mathrm{V}}_{\mathrm{a}} = \vec{\mathrm{V}}_{\mathrm{r}} + \vec{\mathrm{V}}_{\mathrm{e}};\ \ V_r = l_2 * \dot{\varphi},\ \ V_e = l_1 * \dot{\alpha}.$

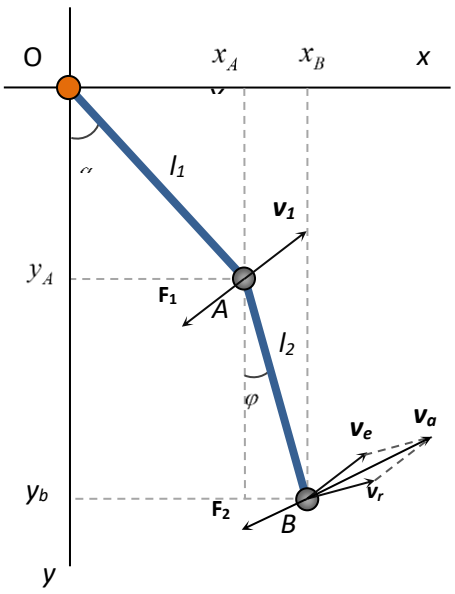

Рисунок 1.2 – Подвійний математичний маятник. Діючи сили та швидкості

З огляду на те, що $\left(\vec{V}_r, \vec{V}_e\right) = \varphi - \alpha$, то $\mathrm{V}_{\mathrm{a}}^2 = \mathrm{V}_{\mathrm{r}}^2 + \mathrm{V}_{\mathrm{e}}^2 + 2\mathrm{V}_{\mathrm{r}} V_e \cos(\varphi - \alpha)$, то кінетична енергія системи набуває вигляду:

$$\mathrm{T} = \frac{m_1 l_1^2 \dot{\alpha}^2}{2} + \frac{m_2}{2}(l_2^2\dot{\varphi}^2 + l_1^2\dot{\alpha}^2 + 2l_1 l_2 \dot{\alpha}\dot{\varphi}\cos(\varphi - \alpha)),\ \text{чи}$$

$$\mathrm{T} = \left(m_1 + m_2\right)\frac{l_1^2\dot{\alpha}^2}{2} + m_2\frac{l_2^2\dot{\varphi}^2}{2} + m_2 l_1 l_2 \dot{\alpha}\dot{\varphi}\cos(\varphi - \alpha). \qquad (1.2)$$

Знайдемо узагальнені сили. На систему діють сили тяжіння маятників, спрямовані вертикально вниз з точок A та B, сили опору середовища $\vec{F}_1$ і $\vec{F}_2$ та сили реакції у точках A і B.

Координати точки A: $x_A = l_1 \sin\alpha\ ;\ \ y_A = l_1 \cos\alpha;$

Координати точки B:

$$x_B = x_A + l_2 \sin\varphi = l_1 \sin\alpha + l_2 \sin\varphi\,;$$
$$y_B = y_A + l_2 \cos\varphi = l_1 \cos\alpha + l_2 \cos\varphi\,.$$

Обчислимо варіації координат точок A і B:

$$\delta x_A = l_1 \cos\alpha\ \delta\alpha, \quad \delta y_A = -l_1 \sin\alpha\,\delta\alpha,$$
$$\delta x_B = l_1 \cos\alpha\ \delta\alpha + l_2 \cos\varphi\,\delta\varphi,$$
$$\delta y_B = -l_1 \sin\alpha\,\delta\alpha - l_2 \sin\varphi\,\delta\varphi.$$

Спочатку надамо стрижню 1 віртуальне переміщення $\delta\alpha \neq 0$, при цьому покладено $\delta\varphi = 0$.

Сили опору, які діють на точки A та B, дорівнюють:

$$\vec{F}_1 = -k\vec{v}_A,\ F_1 = kv_A = kl_1\dot{\alpha}.$$
$$\vec{F}_2 = -k\vec{v}_{\mathrm{B}} = -k\left(\vec{V}_r + \vec{V}_e\right).$$

де k – коефіцієнт опору середовища.

Проекції цієї сили на декартові осі координат дорівнюють:

$$F_{2_x} = -k(l_1\dot{\alpha}\cos\alpha + l_2\dot{\varphi}\cos\varphi);$$
$$F_{2_y} = k(l_1\dot{\alpha}\sin\alpha + l_2\dot{\varphi}\sin\varphi).$$

Елементарні роботи сил тяжіння і опору на зазначених вище віртуальних переміщеннях точок системи, що відповідають віртуальному переміщенню $\delta\alpha$, будуть рівні:

$$\delta A_\alpha\left(\vec{P}_1\right) = P_1\delta y_A = -m_1 g l_1 \sin\alpha\,\delta\alpha;$$
$$\delta A_\alpha\left(\vec{P}_2\right) = P_2\delta y_B = -m_2 g l_1 \sin\alpha\,\delta\alpha;$$
$$\delta A_\alpha\left(\vec{F}_1\right) = \vec{F}_1 \cdot \delta\vec{s}_A = F_{1x}\delta x_A + F_{1y}\delta y_A\,,$$

де $F_{1x} = -F_1 \cos\alpha\,;\ F_{1y} = F_1 \sin\alpha.$

В цілому, елементарна робота силу опору $\vec{F}_1$ на зазначених вище віртуальних переміщеннях точок системи визначається виразом:

$$\delta A_\alpha(F_1) = -F_1 \cos\alpha\, l_1 \cos\alpha\, \delta\alpha - F_1 \sin\alpha\, l_1 \sin\alpha\, \delta\alpha =$$
$$= -F_1 l_1 (cos^2\alpha + sin^2\alpha)\delta\alpha = -F_1 l_1 \delta\alpha = -k l_1^2 \dot{\alpha}\delta.$$

Аналогічно для сили опору $\vec{F}_2$ отримаємо:

$$\delta A_\alpha(\vec{F}_2) = [-k(l_2\dot{\varphi}\cos\varphi + l_1\dot{\alpha}\cos\alpha)l_1\cos\alpha$$
$$-k(l_2\dot{\varphi}\sin\varphi + l_1\dot{\alpha}\sin\alpha)l_1\sin\alpha]\delta\alpha =$$
$$= \left[-kl_1\left(l_2\dot{\varphi}\,(\cos\varphi\cos\alpha + \sin\varphi\sin\alpha) + l_1\dot{\alpha}(cos^2\alpha + sin^2\alpha)\right)\right]\delta\alpha =$$
$$= [-kl_1(l_2\dot{\varphi}\cos(\varphi - \alpha) + l_1\dot{\alpha})]\delta\alpha$$

Робота сил реакції на своїх віртуальних переміщеннях дорівнює нулю, тому що сили перпендикулярні переміщенням.

Сумарна елементарна робота усіх сил на віртуальних переміщеннях точок системи, що відповідають віртуальному переміщенню $\delta\alpha$, виходить рівною:

$$\delta A_\alpha = [-m_1 g l_1 \sin\alpha - m_2 g l_1 \sin\alpha - \mu l_1^2\dot{\alpha} - \mu l_1(l_2\dot{\varphi}\cos(\varphi - \alpha) + l_1\dot{\alpha})]\delta\alpha =$$
$$= -[g l_1 \sin\alpha(m_1 + m_2) + k l_1(2l_1\dot{\alpha} + l_2\dot{\varphi}\cos(\varphi - \alpha))]\delta\alpha.$$

З цього виразу отримаємо вираження для узагальненої сили, що відповідає координаті α:

$$Q_\alpha = -[g l_1 \sin\alpha(m_1 + m_2) + k l_1(2l_1\dot{\alpha} + l_2\dot{\varphi}\cos(\varphi - \alpha))]. \qquad (1.3)$$

Тепер надаємо стрижню 2 віртуальне переміщення $\delta\varphi \neq 0$, при цьому покладемо $\delta\alpha = 0$. Елементарні роботи сил тяжіння і опору на віртуальних переміщеннях точок системи, що відповідають віртуальному переміщенню $\delta\varphi$, будуть рівні:

$$\delta A_\varphi(\vec{P}_1) = 0; \quad \delta A_\varphi(F_1) = 0\ ;$$
$$\delta A_\varphi(\vec{P}_2) = P_2\delta y_B = -m_2 g l_2 \sin\varphi\ \delta\varphi;$$
$$\delta A_\varphi(\vec{F}_2) = 0\ .$$

Враховуючи, що $V_{r_x} = l_2\dot{\varphi}\cos\varphi$; $V_{r_y} = -l_2\dot{\varphi}\sin\varphi$; $V_{e_x} = l_1\dot{\alpha}\cos\alpha$; $V_{e_y} = -l_1\dot{\alpha}\sin\alpha$, отримаємо вираз для елементарної роботи сили опору $\vec{F}_2$ на зазначених вище віртуальних переміщеннях точок системи:

$$\delta A_\varphi(\vec{F}_2) = \vec{F}_2 \cdot \delta\vec{s}_B = -k[(l_2\dot{\varphi}\cos\varphi + l_1\dot{\alpha}\cos\alpha)l_2\cos\varphi +$$
$$+ (l_2\dot{\varphi}\sin\varphi + l_1\dot{\alpha}\sin\alpha)\, l_2\sin\varphi]\delta\varphi = -kl_2[\, l_2\dot{\varphi}(cos^2\varphi + sin^2\varphi) +$$
$$+ l_1\dot{\alpha}(\cos\alpha\cos\varphi + \sin\alpha\sin\varphi)]\delta\varphi = -kl_2[l_2\dot{\varphi} + l_1\dot{\alpha}\cos(\varphi - \alpha)]\delta\varphi$$

Сумарна же елементарна робота усіх сил на віртуальних переміщеннях точок системи, що відповідають віртуальному переміщенню $\delta\varphi$, виходить рівною: $\delta A_\varphi = \left[-m_2 g l_2 \sin\varphi - kl_2[l_2\dot{\varphi} + l_1\dot{\alpha}\cos(\varphi - \alpha)]\right]\delta\varphi$.

З цього виразу отримаємо вираження для узагальненої сили, що відповідає координаті φ:

$$Q_\varphi = -m_2 g l_2 \sin\varphi - kl_2[l_2\dot{\varphi} + l_1\dot{\alpha}\cos(\varphi - \alpha)]. \qquad (1.4)$$

Знайдемо частинні похідні від T:

$$\frac{\partial T}{\partial \alpha} = m_2 l_1 l_2 \dot{\varphi}\dot{\alpha}\sin(\alpha-\varphi);$$
$$\frac{\partial T}{\partial \varphi} = -m_2 l_1 l_2 \dot{\alpha}\dot{\varphi}\sin(\varphi-\alpha); \quad (1.5)$$

$$\frac{\partial T}{\partial \dot{\alpha}} = l_1^2\dot{\alpha}(m_1+m_2) + m_2 l_1 l_2 \dot{\varphi}\cos(\varphi-\alpha);$$
$$\frac{\partial T}{\partial \dot{\varphi}} = l_2^2 m_2\dot{\varphi} + m_2 l_1 l_2 \dot{\varphi}\cos(\varphi-\alpha); \quad (1.6)$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\alpha}}\right) = l_1^2\ddot{\alpha}(m_1+m_2) + m_2 l_1 l_2 \ddot{\varphi}\, cos(\varphi-\alpha) - \\ - m_2 l_1 l_2 \dot{\varphi}\, sin(\varphi-\alpha)\,(\dot{\varphi}-\dot{\alpha}); \quad (1.7)$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\varphi}}\right) = l_2^2 m_2\ddot{\varphi} + m_2 l_1 l_2 \ddot{\alpha}\, cos(\varphi-\alpha) - \\ - m_2 l_1 l_2 \dot{\alpha}\, sin(\varphi-\alpha)\,(\dot{\varphi}-\dot{\alpha}). \quad (1.8)$$

Прирівнявши ліві і праві частини рівняння Лагранжа II роду, тобто підставивши формули (1.3) – (1.8) до (1.1), отримаємо диференційні рівняння руху маятника:

$$l_1^2\ddot{\alpha}(m_1+m_2) + m_2 l_1 l_2 \ddot{\varphi}\cos(\varphi-\alpha) - m_2 l_1 l_2 \dot{\varphi}^2\sin(\varphi-\alpha) \\ + m_2 l_1 l_2 \dot{\alpha}\dot{\varphi}\sin(\varphi-\alpha) - m_2 l_1 l_2 \dot{\alpha}\dot{\varphi}\sin(\varphi-\alpha) = \\ = -[g l_1 \sin\alpha(m_1+m_2) + k l_1(2 l_1\dot{\alpha} + l_2\dot{\varphi}\cos(\varphi-\alpha))];$$
$$l_2^2 m_2\ddot{\varphi} + m_2 l_1 l_2 \ddot{\alpha}\cos(\varphi-\alpha) - m_2 l_1 l_2 \dot{\alpha}^2\sin(\varphi-\alpha) - m_2 l_1 l_2 \dot{\alpha}\dot{\varphi}\sin(\varphi-\alpha) \\ + m_2 l_1 l_2 \dot{\alpha}\dot{\varphi}\sin(\varphi-\alpha) = \\ = -m_2 g l_2 \sin\varphi - k[l_2\dot{\varphi} + l_1\dot{\alpha}\cos(\varphi-\alpha)].$$

Таким чином, отримаємо систему двох диференційних рівнянь II порядку з двома невідомими α і φ. Після її спрощення отримаємо:

$$\begin{cases} l_1^2\ddot{\alpha}(m_1+m_2)+m_2l_1l_2\ddot{\varphi}\cos(\varphi-\alpha)-m_2l_1l_2\dot{\varphi}^2\sin(\varphi-\alpha)= \\ =-[gl_1\sin\alpha(m_1+m_2)+kl_1(2l_1\dot{\alpha}+l_2\dot{\varphi}\cos(\varphi-\alpha))]; \\ l_2^2m_2\ddot{\varphi}+m_2l_1l_2\ddot{\alpha}\cos(\varphi-\alpha)-m_2l_1l_2\dot{\alpha}^2\sin(\varphi-\alpha)= \\ =-m_2gl_2\sin\varphi-kl_2[l_2\dot{\varphi}+l_1\dot{\alpha}\cos(\varphi-\alpha)]. \end{cases} \quad (1.9)$$

Виведені рівняння є рівняннями руху подвійного математичного маятника. Вони використовуватимуться в Windows-додатку для розрахунку положень усіх тіл, що становлять механізм, в кожен момент часу.

Подальше рішення задачі зводиться до чисельного інтегрування отриманої системи (1.9) двох диференціальних рівнянь з наступними початковими умовами:

при t = 0: $\alpha(0)=\alpha_0$, $\dot{\alpha}(0)=\dot{\alpha}_0$, $\varphi(0)=\varphi_0$, $\dot{\varphi}(0)=\dot{\varphi}_0$.

1.4 Малі коливання подвійного маятника

Для випадку малих коливань без урахування сили опору середовища рівняння (1.9) спрощуються [1, 2], і система рівнянь стає лінійною. А для лінійної системи існує аналітичне рішення.

Знехтуємо в рівняннях (1.9) членами $\dot{\alpha}^2$ і $\dot{\varphi}^2$ та замінимо наступні величини наближеними їх значеннями:

$$\sin\alpha\approx\alpha,\ \sin\varphi\approx\varphi,\ \sin(\alpha-\varphi)\approx\alpha-\varphi,$$
$$\cos\alpha\approx1,\ \cos\varphi\approx1,\ \cos(\alpha-\varphi)\approx1.$$

В цьому випадку отримаємо наступну систему рівнянь руху для випадку малих коливань:

$$\begin{cases} (m_1 + m_2) l_1 \ddot{\alpha} + m_2 l_2 \ddot{\varphi} + (m_1 + m_2) g\alpha = 0; \\ l_1 \ddot{\alpha} + l_2 \ddot{\varphi} + g\varphi = 0. \end{cases} \qquad (1.10)$$

Цю систему рівнянь можна записати в компактній матричній формі [2]. Введено матриці:

$$\beta(t) = \begin{pmatrix} \alpha(t) \\ \varphi(t) \end{pmatrix}, \; M = \begin{pmatrix} (m_1 + m_2) l_1^2 & m_2 l_1 l_2 \\ m_2 l_1 l_2 & m_2 l_2^2 \end{pmatrix}, \; K = \begin{pmatrix} (m_1 + m_2) g l_1 & 0 \\ 0 & m_2 g l_2 \end{pmatrix}.$$

Таким чином система диференціальних рівнянь представляється у вигляді $M \ddot{\alpha} + K\alpha = 0.$

У разі одного тіла рівняння такого виду описує вільні незгасаючі коливання з певною частотою. У разі подвійного маятника рішення міститиме коливання з двома характерними частотами, які називаються нормальними модами. Вони є дійсною частиною комплексно значної векторної функції:

$$\beta(t) = \begin{pmatrix} \alpha(t) \\ \varphi(t) \end{pmatrix} = \mathrm{Re}\left[\begin{pmatrix} H_1 \\ H_2 \end{pmatrix} \exp(i\omega t) \right].$$

Тут H_1 і H_2 – власні вектори, ω - дійсна частота. Значення нормальних частот $\omega_{1\,2}$ визначаються з рішення характеристичного рівняння $\det(K - \omega^2 M) = 0.$

Виведемо загальні формули для циклічних частот ω_1, ω_2 у разі довільних мас m_1, m_2 і довжин l_1, l_2:

$$\begin{vmatrix} (m_1 + m_2) g l_1 - \omega^2 (m_1 + m_2) & l_1^2 & -\omega^2 m_2 l_1 l_2 \\ -\omega^2 m_2 l_1 l_2 & & m_2 g l_2 - \omega^2 m_2 l_2^2 \end{vmatrix} = 0,$$

чи

$$\begin{vmatrix} (m_1+m_2)(gl_1-\omega^2 l_1^2) & -\omega^2 m_2 l_1 l_2 \\ -\omega^2 m_2 l_1 l_2 & m_2(gl_2-\omega^2 l_2^2) \end{vmatrix} = 0,$$

Звідки

$$m_2(m_1+m_2)(gl_1-\omega^2 l_1^2)(gl_2-\omega^2 l_2^2)-\omega^4 m_2^2 l_1^2 l_2^2 = 0,$$

$$m_2 l_1 l_2\left[(m_1+m_2)(g^2-\omega^2 g l_1-\omega^2 g l_2+\omega^4 l_1 l_2)-\omega^4 m_2 l_1 l_2\right] = 0,$$

$$(m_1+m_2)g^2-\omega^2(m_1+m_2)(l_1+l_2)g+\omega^4(m_1+m_2)l_1 l_2-\omega^4 m_2 l_1 l_2 = 0,$$

$$(m_1+m_2)g^2-\omega^2(m_1+m_2)(l_1+l_2)g+\omega^4 m_1 l_1 l_2 = 0.$$

Отримаємо біквадратне рівняння для частот ω. Обчислимо дискримінант:

$$\begin{aligned} D &= (m_1+m_2)^2(l_1+l_2)^2 g^2 - 4m_1(m_1+m_2)g^2 l_1 l_2 \\ &= g^2(m_1+m_2)[(m_1+m_2)(l_1+l_2)^2 - 4m_1 l_1 l_2] \end{aligned}$$

Таким чином, квадрати нормальних частот $\omega_{1,2}$ дорівнюють:

$$\omega_{1,2}^2 = \frac{g}{2m_1 l_1 l_2}\left\{(m_1+m_2)(l_1+l_2) \pm \sqrt{(m_1+m_2)[(m_1+m_2)(l_1+l_2)^2 - 4m_1 l_1 l_2]}\right\}$$

$$\omega_{1,2}^2 = \frac{g}{2m_1 l_1 l_2} \times$$

$$\times\left\{(m_1+m_2) \pm \sqrt{(m_1+m_2)[(m_1+m_2)(l_1+l_2)^2 - 4m_1 l_1 l_2]}\right\}$$

Введемо позначення: $\mu = \frac{m_2}{m_1}$, $B = \frac{l_2}{l_1}$. Тоді останній вираз можна переписати в наступному вигляді:

$$\omega_{1,2}^2 = \frac{g}{2l_2}\{(1+\mu)(1+B) \pm \sqrt{(1+\mu)[(1+\mu)(1+B)^2 - 4B}\,. \qquad (1.11)$$

Тепер необхідно визначити значення власних векторів H_1 і H_2. Вони знаходяться з рішення векторно-матричного рівняння

$$(K - \omega^2 M)H = 0. \qquad (1.12)$$

Нехай власний вектор H_1 відповідає частоті ω_1, а вектор H_2 - частоті ω_2. Тоді для визначення H_1 отримано наступне рівняння $(K - \omega_1^2 M)H_1 = 0$, що призводить до системи двох рівнянь:

$$\begin{cases}(K_{11} - \omega_1^2 M_{11})H_{11} + (K_{12} - \omega_1^2 M_{12})H_{21} = 0,\\ (K_{21} - \omega_1^2 M_{21})H_{11} + (K_{22} - \omega_1^2 M_{22})H_{21} = 0.\end{cases}$$

З першого рівняння системи отримаємо:

$$\frac{H_{21}}{H_{11}} = -\frac{K_{11} - \omega_1^2 M_{11}}{K_{12} - \omega_1^2 M_{12}} = \frac{1+\mu}{\mu B} \cdot \frac{2B - (1+\mu)(1+B) - \sqrt{(1+\mu)[(1+\mu)(1+B)^2 - 4B]}}{(1+\mu)(1+B) + \sqrt{(1+\mu)[(1+\mu)(1+B)^2 - 4B]}}.$$

Таким чином, компоненти власного вектору H_1 рівні:

$$H_{11} = 1,$$

$$H_{21}=\frac{1+\mu}{\mu B}\cdot\frac{2B-(1+\mu)(1+B)-\sqrt{(1+\mu)[(1+\mu)(1+B)^2-4B]}}{(1+\mu)(1+B)+\sqrt{(1+\mu)[(1+\mu)(1+B)^2-4B}},$$ або

$$H_{21}=-\frac{1+\mu}{\mu B}\cdot\frac{(1+\mu)(1-B)-2\mu B-\sqrt{(1+\mu)[(1+\mu)(1+B)^2-4B]}}{(1+\mu)(1+B)+\sqrt{(1+\mu)[(1+\mu)(1+B)^2-4B]}}.$$

Аналогічно обчислюються і компоненти H_{12} та H_{22} власного вектору H_2:

$$H_{12}=1,$$

$$H_{22}=-\frac{1+\mu}{\mu B}\cdot\frac{(1+\mu)(1-B)-2\mu B+\sqrt{(1+\mu)[(1+\mu)(1+B)^2-4B]}}{(1+\mu)(1+B)-\sqrt{(1+\mu)[(1+\mu)(1+B)^2-4B]}}.$$

Тепер можна записати загальне рішення для малих коливань маятників:

$$\begin{aligned}\alpha(t)&=c_1H_{11}\cos(\omega_1t+\varphi_1)+c_2H_{12}\cos(\omega_2t+\varphi_2),\\ \varphi(t)&=c_1H_{21}\cos(\omega_1t+\varphi_1)+c_2H_{22}\cos(\omega_2t+\varphi_2).\end{aligned}\qquad(1.13)$$

Тут постійні c_1, c_2, φ_1, φ_2 залежать від початкових положень і швидкостей маятників. Нехай в початковий момент часу t=0 обидва маятники можуть мати відхилення від положення рівноваги, а швидкості їх дорівнюють нулю, тобто:

$$\alpha(0)=\alpha_0,\ \dot{\alpha}(0)=0,\varphi(0)=\varphi_0,\ \dot{\varphi}(0)=0.$$

В цьому випадку початкові фази дорівнюють нулю: $\varphi_1 = \varphi_2 = 0$. Визначено постійні c_1 і c_2. Підставивши початкові дані в рівняння (13), отримано: $\alpha_0=c_1+c_2,\ \varphi_0=c_1H_{21}+c_2H_{22}$.

З цих рівнянь отримаємо:

$$c_1=-\frac{\varphi_0-\alpha_0H_{22}}{H_{22}-H_{21}},\ c_2=\frac{\varphi_0-\alpha_0H_{21}}{H_{22}-H_{21}}.$$

1.5 Чисельне рішення рівнянь руху маятника

Для чисельного рішення отриманої системи диференціальних рівнянь руху подвійного маятника (1.9) застосовано метод Рунге-Кутта IV порядку точності [1]. Цей метод дозволяє ефективно моделювати динамічні процеси, а це надає можливість аналізувати поведінку навіть складних систем. Метод Рунге-Кутта є одно-кроковим методом. Хоча у нас диференціальні рівняння другого порядку, але можна до кожного з них застосувати метод Рунге-Кутта для звичайних диференціальних рівнянь першого порядку. Для цього необхідно рівняння другого порядку привести до системи рівнянь першого порядку:

$$\begin{cases} y' = w; \\ w' = f(x, y, w). \end{cases}$$

А далі вже до кожного з рівнянь необхідно застосувати метод Рунге-Кутта для звичайних диференціальних рівнянь першого порядку. Розрахункова схема методу Рунге-Кутта IV порядку для рівняння $y'' = f(x, y, y')$ має наступний вигляд:

$$\begin{cases} y'_{i+1} = y'_i + \dfrac{1}{6}\left(k_1 + 2\cdot k_2 + 2\cdot k_3 + k_4\right); \\ y_{i+1} = y_i + h\left(y'_i + \dfrac{1}{6}\left(k_1 + k_2 + k_3\right)\right); \\ k_1 = h\cdot f\left(x_i, y_i, y'_i\right); \\ k_2 = h\cdot f\left(x_i + \dfrac{h}{2}, y_i + \dfrac{h}{2}y'_i + \dfrac{h}{8}k_1, y'_i + \dfrac{k_1}{2}\right); \\ k_3 = h\cdot f\left(x_i + \dfrac{h}{2}, y_i + \dfrac{h}{2}y'_i + \dfrac{h}{8}k_2, y'_i + \dfrac{k_2}{2}\right); \\ k_4 = h\cdot f\left(x_i + h, y_i + hy'_i + \dfrac{h}{2}k_3, y'_i + k_3\right). \end{cases} \quad (1.14)$$

У задачі цей метод застосовується послідовно спочатку до одного рівняння системі (1.9), а потім до іншого. В якості параметра x береться параметр t, а в якості функції y - функції α та φ.

При чисельному розрахунку методом Рунге-Кутта даний проміжок часу розбивається на відрізки завдовжки h (для даної система це Δt). На цьому відрізку потрібно знайти рішення диференціального рівняння в кожній точці інтегральної кривої, що проходить через цю точку при виконанні початкових умов, причому існування і єдиність такої кривої забезпечується теоремою Пікара. А по значеннях функцій $\alpha(t)$ та $\varphi(t)$ та їх похідних на початку кожного тимчасового відрізку по формулах (1.14) розраховуються значення $\alpha(t+h)$ та $\varphi(t+h)$ та їх похідних у кінці відрізку часу.

2 ПРОГРАМНА РЕАЛІЗАЦІЯ WINDOWS-ДОДАТКУ

2.1 Обґрунтування вибору напряму дослідження та застосовуваних методів

Вибір подвійного математичного маятника як об’єкта дослідження зумовлений низкою факторів. По-перше, це система з невеликою кількістю ступенів вільності, яка водночас демонструє характерні риси складної нелінійної динаміки, включаючи перехід від регулярної до хаотичної поведінки. Ця властивість робить її ідеальною моделлю для вивчення нелінійних ефектів, які мають місце у значно складніших системах — таких як багатоланкові механізми, біомеханічні структури, а також автоматизовані керовані системи в робототехніці.

Практичне значення моделі полягає у тому, що кінематична структура подвійного маятника аналогічна конструкції шарнірних механізмів, які використовуються в маніпуляторах, підвісних стабілізаторах, ортопедичних тренажерах тощо. Знання динаміки таких систем дозволяє проєктувати їх точніше, з урахуванням стійкості, резонансних режимів і реакції на зміну зовнішніх сил чи початкових умов. Крім того, симуляція маятника дозволяє безпечне віртуальне тестування, уникаючи ризиків та затрат, пов’язаних із фізичними експериментами.

З математичної точки зору, задача опису руху подвійного маятника з урахуванням сили опору не має точного аналітичного розв’язку. Рівняння руху виводяться через рівняння Лагранжа другого роду і зводяться до системи нелінійних звичайних диференціальних рівнянь (СОДР) другого порядку. Для їх розв’язання найчастіше використовуються чисельні методи.

Серед чисельних методів для розв’язання СОДР доцільним є застосування методу Рунге–Кутта 4-го порядку. Цей метод забезпечує високу точність при помірному обчислювальному навантаженні і добре підходить для задач, де розв’язок потрібно отримувати поетапно із заданим кроком,

наприклад, для побудови графічної симуляції у реальному часі. Метод дозволяє стабільно інтегрувати як малі, так і великі за амплітудою коливання без значної втрати точності навіть у випадках різкої зміни динаміки системи.

З точки зору програмної реалізації вибір середовища Microsoft Visual Studio та мови програмування C# обґрунтований як технічними, так і практичними міркуваннями. Середовище Visual Studio забезпечує повноцінну підтримку розробки настільних застосунків з графічним інтерфейсом (WinForms), що дає змогу зручно реалізувати візуалізацію руху маятника, контроль параметрів моделі, обробку користувацького вводу. Мова C# поєднує продуктивність, об’єктно-орієнтовану структуру і доступність, дозволяючи реалізовувати складні обчислювальні алгоритми паралельно з побудовою інтерфейсу. Крім того, C# має хорошу підтримку бібліотек для графіки, обчислень і інтеграції з іншими .NET-компонентами, що відкриває перспективу подальшого розширення функціоналу проєкту.

У підсумку, обрана комбінація об'єкта, методів розв’язання та інструментів реалізації дозволяє побудувати якісну модель, що буде одночасно точною, гнучкою в управлінні та зручною для користувача. Такий підхід створює основу не лише для аналізу поведінки маятника, а й для потенційного практичного застосування в освітніх або інженерних задачах.

2.2 Програмна реалізація Windows додатку

Для комп'ютерної реалізації завдання розроблено Windows-додаток візуалізації руху подвійного математичного маятника в режимі віртуального часу в інтегрованому середовищі Microsoft Visual Studio 2022 на мові програмування C# [6-12]. Програма написана із використанням принципів об’єктно-орієнтованого програмування.

Створено клас DoublePendulum.cs (додаток Б) та SmallOscillation.cs (додаток В), де зосереджена математична частина додатку. Опис цих класів наведено нижче.

Клас *DoublePendulum* інкапсулює всю фізичну й чисельну логіку подвійного маятника. Насамперед оголошено константу *G*, що задає прискорення вільного падіння у СІ-системі. Далі розташовані параметри моделі, відкриті через авто-властивості: *H* — крок інтегрування, *L1* і *L2* — довжини обох тяг, *M1* і *M2* — маси вантажів, а також коефіцієнт лінійного опору *B*.

Окрему групу становлять змінні стану: кутові координати *Alpha* (верхня ланка) та *Fi* (нижня), і відповідні кутові швидкості *DAlpha*, *DFi*. Ці величини разом утворюють чотиривимірний фазовий вектор, який методи класу безпосередньо змінюють у ході інтегрування.

Чисельне інтегрування виконує публічний метод *RungeKut()*. Він реалізує схему Рунґе-Кутта четвертого порядку (1.14). У процесі виклику обчислюються чотири набори коефіцієнтів (k1…k4, l1…l4) за власними рівняннями руху, після чого координати й швидкості оновлюються на кроці *H*.

Праві частини рівнянь руху (1.9) задані двома приватними методами *Fx1()* та *Fx2()*. Вони повертають кутові прискорення відповідно верхньої та нижньої ланок, попередньо переводячи довжини й маси із технічних одиниць (сантиметри, грами) у метри та кілограми. Усередині методів додатково враховується коефіцієнт опору.

Для енергетичного контролю передбачено публічний метод *ComputeEnergy()*. Він обчислює кінетичну й потенціальну складові кожної маси, використовуючи поточні координати та швидкості, що дозволяє відстежувати сталість повної механічної енергії або, навпаки, її згасання під дією демпфування.

Клас *SmallOscillation* призначено для вирішення задачі розрахунку для опису малих коливань подвійного маятника.

У конструкторі задано фізичну константу *G*, що задає прискорення вільного падіння. Параметри моделі, доступні через властивості, охоплюють маси підвісів (*M1*, *M2*), їхнє відношення *Mu = M1 / M2*, ефективні довжини

підвісів (*Rad1, Rad2*) та безрозмірний коефіцієнт пружного зв'язку *B*. Початкові кути *Alpha* і *Fi* задаються в радіанах.

Метод *Compute()* виконує повний розрахунок нормальних координат. Спершу формуються точні вирази для власних частот *Omega1* і *Omega2*, що витікають з характеристичного рівняння лінійної системи другого порядку. Після цього обчислюються величини *H1* і *H2*, які зв'язують кути обох ланок у кожному нормальному режимі; далі, виходячи з початкових умов (*Alpha, Fi*), визначаються амплітудні множники *C1* і *C2*. У результаті клас повністю параметризовано для подальшого відтворення еволюції у часі.

Метод *UpdateAngles(time)* реалізує сам аналітичний розв'язок.

Таким чином *SmallOscillation* надає можливість миттєво отримати точну траєкторію для малих відхилень і слугує еталоном для перевірки чисельних результатів нелінійної моделі.

В процесі візуалізації руху механізму промальовування усіх складових механізму виконується в 2D-форматі з використанням простору імен System.Drawing.Drawing2D, що забезпечує розширені можливості двовимірної та векторної графіки.

2.3 Опис Windows додатку

На формі під назвою «Подвійний математичний маятник» розташовано компонент TabControl, який складається з двох сторінок: «Кінцеві коливання» та «Малі коливання». На першій сторінці можна переглянути візуалізацію кінцевих коливань подвійного математичного маятника з урахуванням сили опору середовища та без. На другій - випадок малих коливань подвійного маятника без урахування сили опору середовища. На кожній сторінці розташоване своє меню, яке складається з трьох пунктів: «Механізм», «Рух» та «Вихід».

Сторінки розроблені в одному стилі. На кожній сторінці присутні три панелі.

Зліва розташована панель з компонентами для завдання початкових параметрів маятника: кутів відхилення складових маятника, їх мас та довжин ниток, а також коефіцієнт опору середовища.

Нижче показаний вигляд вікна додатку:

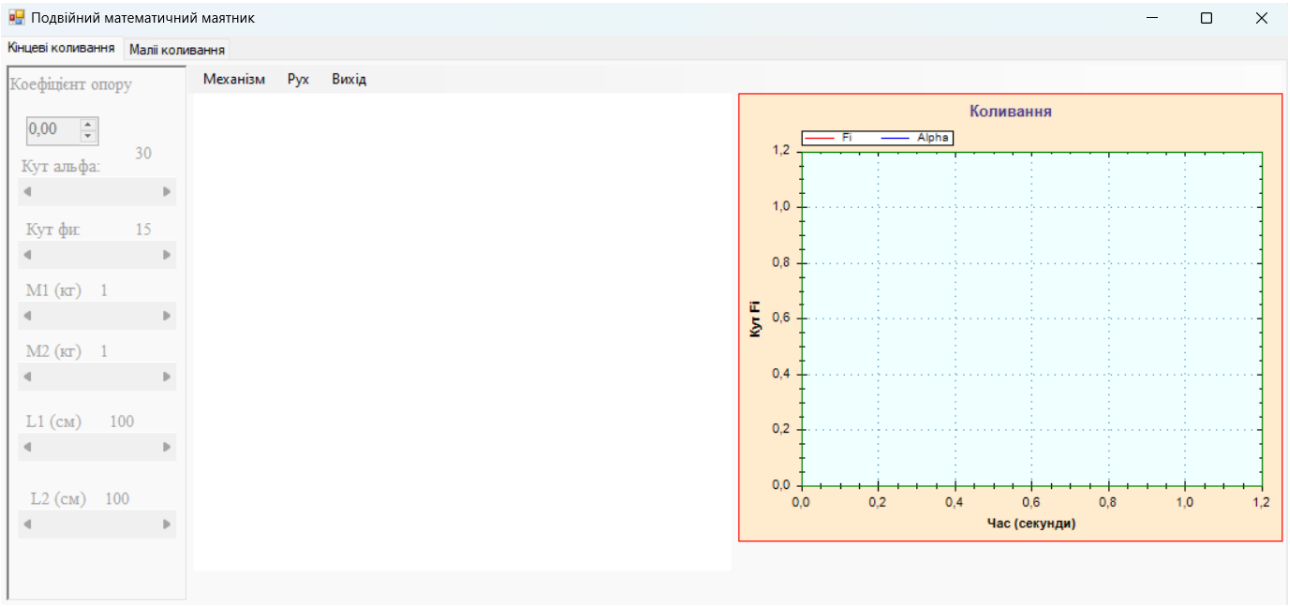

Рисунок 2.1 – Вікно додатку

Правіше розташована панель з компонентом pictureBox. На ньому повинен бути зображений маятник з деякими початковими параметрами. Згодом на цьому компонентові і проводитиметься візуалізація руху маятників. Для того, щоб малюнок з'явився на даній панелі, необхідно вибрати команду «Зобразити» у пункті меню «Механізм». Механізм з'явиться в положенні «за замовчуванням», в якому обидва маятники зображені відхиленими на 30 і 15 градусів відповідно. Маси цих маятників у цьому становищі дорівнюють по 1 кг. Довжини – по 100 см. З огляду на те, що істинні розміри ниток на екрані не можна відобразити, їх довжини зображуються в масштабі по відношенню до деякої довжини, обраної в якості характерного розміру. Для мас теж вибрано характерна маса, а на малюнку величини мас характеризують кульки відповідного розміру, які пропорційні масам маятників. В процесі вибору

вказаних параметрів на малюнку видозмінюється початкове положення маятника,

Користувач може за допомогою смужок прокрутки, які розташовані зліва, змінити задані маси і довжини ниток, вибравши їх необхідні значення у випадку кінцевих коливань чи співвідношення мас маятників і співвідношення їх довжин для малих коливань, а також початкові відхилення маятників. Початкові кутові швидкості маятників передбачаються рівними нулю. Для вибору цих параметрів застосовані компоненти типу hScrollBar, які не дозволяють вибирати величини вище деяких дозволених. Ці дозволені величини обрані з тих умов, щоб малюнок не виходив за межі відведеної області. Що ж до вибору величин початкових кутів відхилень, у процесі руху весь час має виконуватися умова додатньості сил натягу складових маятник ниток. У зв'язку з тим, що цей розрахунок супроводжується трудомісткими теоретичними обчисленнями, максимальні відхилення ниток у додатку встановлені приблизно за допомогою комп'ютерного експерименту.

З відображеним подвійним маятником сторінка виглядає вже таким чином:

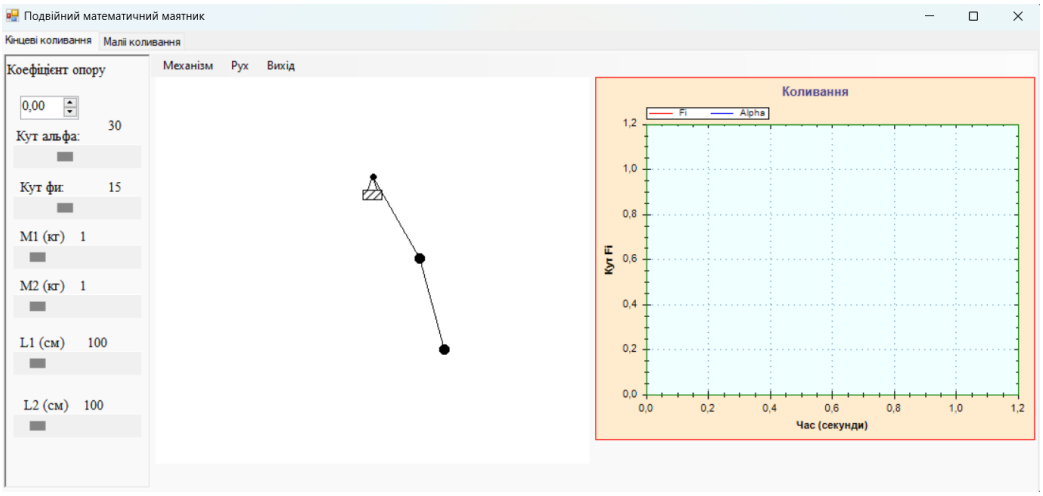

Рисунок 2.2 – Вікно додатку з зображенням маятників

Після того, як початкове положення маятника вибрано, можна запустити його в рух за допомогою команди «Запустити» пункту меню «Рух».

Праворуч у вікні розташована панель с компонентом *ZedGraph*, в якому у процесі візуалізації руху подвійного маятника відображуються графіки залежності кутів відхилення обох маятників α та φ від положення рівноваги за часом *t*.

У будь-який момент можна рух зупинити за допомогою команди «Зупинити», а потім знову відновити процес руху за допомогою команди «Запустити».

Для того, щоб змінити параметри маятника і почати новий експеримент, необхідно зупинити рух маятників та стерти механізм за допомогою команди «Витерти» пункту меню «Механізм».

І тоді знову стануть доступними лінійки прокрутки.

Для того, щоб перейти до візуалізації руху подвійного маятника з малими коливаннями, необхідно перейти на другу вкладку. Робота з маятником на цій вкладці аналогічна попередній. Ця сторінка має вигляд:

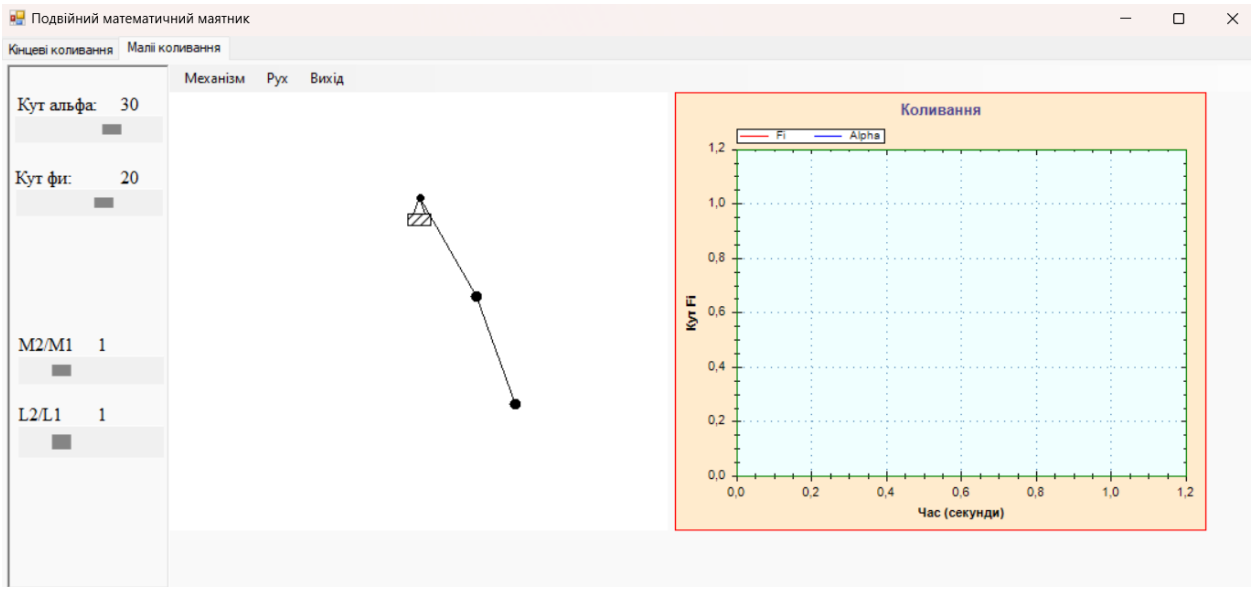

Рисунок 2.3 – Вікно додатку. Випадок малих коливань

Меню «Вихід» здійснює вихід із програми.

Далі буде розібрано по порядку усі пункти меню з точки зору їх програмної реалізації.

На обох сторінках реалізовано візуалізацію руху маятників в режимі віртуального часу.

Подієва функція *зобразитиToolStripMenuItem_Click()*, яка обробляє команду Механізм -> Зобразити, промальовує зображення механізму із заданими користувачем параметрами в компонент *pictureBox1* за допомогою функції *Picture_1()*.

Картина руху виводиться в компонент класу *PictureBox* теж за допомогою функції *Picture_1()*, алгоритм якої буде описаний нижче.

Рух маятників в програмі пов'язаний з компонентом класу *Timer*. Через заданий в програмі часовий інтервал викликається подія *OnTimer* таймера, обробником якого є подієва функція *timer1_Tick()*. Саме вона "примушує" механізм рухатися. Ця функція виконує основне завдання візуалізації механізму шляхом виклику функцій, які виконують конкретні розподілені завдання, пов'язані з візуалізацією.

Ідея реалізації процесу візуалізації руху механічної системи побудована таким чином: зображується механізм в початковому положенні, потім у кінці кожного тимчасового інтервалу $t+\Delta t$ перераховуються кути повороту кульок навколо точки підвісу маятника, перераховуються координати всіх складових механізму, потім екран очищується, і механізм зображується в новому положенні. Нового значення кутів повороту кульок набуваємо, застосовуючи метод Рунге-Кутта для вирішення системи диференційних рівнянь (1.9) у випадку кінцевих коливань або за формулами (1.13) у випадку малих коливань. Таким чином, на малюнку ми бачимо безперервну картину руху маятників навколо своїх вісей.

Промальовуванням усіх складових механізму займається функція *Picture_1(bool cl, BufferedGraphics _bufferedGraphics)*. Ця функція має два параметри.

Параметр логічного типу cl означає, яким кольором малювати всі складові механізму, чорним чи білим. Для того, щоб зобразити механізм наприкінці тимчасового інтервалу, ми повинні старе зображення стерти з екрану і замість нього намалювати нове. Існують 2 шляхи видалення зображення: або очистити увесь екран від зображення, або намалювати те ж зображення, але вже білим кольором. Таким чином колишнє зображення затреться. В своїй програмі я вибрав другий шлях.

Тому на кожному тимчасовому кроці механізм на компоненті pictureBox промальовується двічі. На початку кроку за часом викликається функція *Picture_1()* зі змінною *cl*, значення якої відповідає промальовуванню механізму білим, а після розрахунку кутів у кінці тимчасового інтервалу функція викликається зі змінною cl для чорного кольору.

Другий параметр *_bufferedGraphics* типу BufferedGraphics займається беспосередньо графікою. Промальовування механізмів відбувається з використанням 2D-графіки. Для цього C# на платформі Windows використовує графічний інтерфейс GDI+. Основою інтерфейсу GDI+ являється клас Graphics з простору імен *System.Drawing*. Клас *Graphics* інкапсулює поверхню для малювання GDI+. Клас *Graphics* підтримує ряд методів, що дозволяють малювати прості фігури. У методі *Picture_1()* використані наступні:

- DrawLine() - сполучає дві точки, що задаються парами координат, лінією.

- DrawRectangle() - малює прямокутник, який визначений парою координат, шириною і висотою.

- DrawEliipse() - малює еліпс/коло, визначуваний обмежуючим прямокутником, заданим за допомогою координат верхнього лівого кута прямокутника, висоти і ширини.

- FillEllipse(), FillRectangle() - заповнює внутрішню частину еліпса, внутрішню частину прямокутника відповідно.

Після обчислення чергового положення маятника, функція Traektotriya() відповідає за візуалізацію траєкторії другого (нижнього) тіла маятника. Для цього обчислюється положення центру тяжіння другої ланки в декартовій системі координат на підставі поточних значень кутів α та φ. Отримана точка додається до списку ListA, який зберігає послідовність усіх попередніх положень. На основі цього списку будується та відображається гладка червона крива, яка є візуальним представленням траєкторії руху нижньої ланки маятника. Це дозволяє користувачу в реальному часі оцінювати геометричну структуру руху системи, а також виявляти можливі переходи між регулярним та хаотичним режимами.

У програмі реалізована подвійна буферизація за допомогою об’єкту BufferedGraphics, що дозволяє уникнути ефекту мерехтіння при оновленні зображення. Завдяки цьому зображення маятника та його траєкторії оновлюються плавно, без небажаних візуальних артефактів. Буферна графіка використовується для попереднього промальовування всіх елементів сцени з подальшим одноразовим виведенням результату на екран, що істотно покращує сприйняття та зменшує навантаження на систему.

Таким чином, реалізована структура програмного коду забезпечує як точне чисельне моделювання фізичних процесів, так і якісне їх візуальне представлення, що є важливим у контексті аналізу поведінки нелінійних коливальних систем.

На рисунку 2.4 можна побачити цю траєкторію у довільний момент роботи механізму.

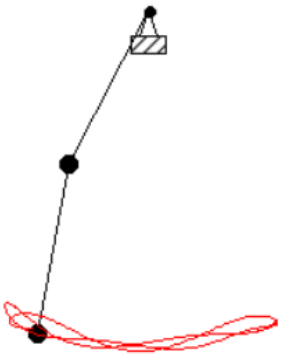

Рисунок 2.4 – Траєкторія руху центру тяжіння другого маятника з часом

Командам «Зупинити» і «Запустити» в меню «Рух» відповідають подієві функції, в яких таймер відключається і включається знову.

Для запуску додатка з іншими параметрами необхідно додаток зупинити, очистити екран, а потім задавати нові параметри, оскільки при очищенні екрану обнуляється і час. Таким чином все готується до виконання нового розрахунку. Очищення екрана можна здійснити за допомогою команди Механізм → Витерти.

Паралельно з візуалізацією руху подвійного маятника на панелі праворуч відображуються графіки залежності кутів відхилення обох маятників φ та α від положення рівноваги за часом. Це здійснюється за допомогою бібліотеки ZedGraph.dll, яка, будучи підключеною до проекту, інкапсулює компонент ZedGraphControl, розміщений у правій частині вікна. У нього виводяться два графіки – синім кольором залежність кута α від t, а червоним – залежність кута φ від t, які у процесі розрахунку.

На рисунку 2.5 показано, як виглядають ці графіки у довільний час у процесі розрахунку.

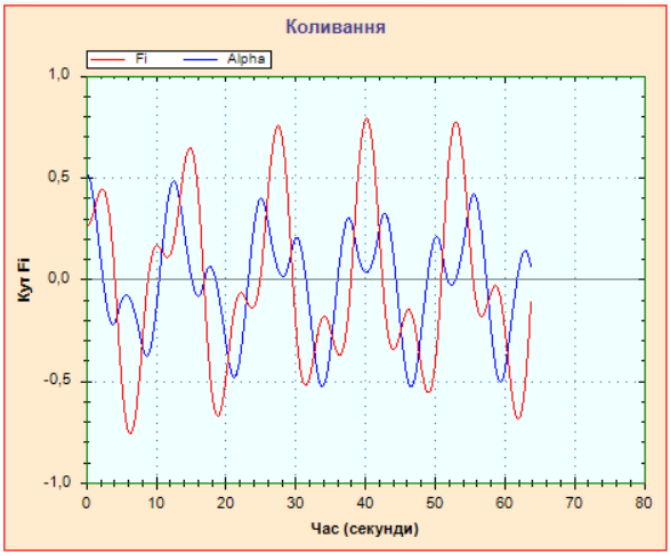

Рисунок 2.5 – Графіки залежності кутів нахилу маятників у компоненті ZedGraphControl

Виведенням даних графіків у компонент ZedGraphControl та налаштуванням його роботи займається функція Grid().

Всередині неї встановлюються кольори рамки для всього компонента та рамки навколо графіка, а також фону компонента, встановлюються кольори для графіків та всіх підписів у компоненті. Це здійснюється шляхом встановлення відповідних значень властивостей змінної pane типу GraphPane. Наприклад, сітка встановлюється за допомогою операторів:

```
pane.XAxis.MajorGrid.IsVisible = true;
pane.YAxis.MajorGrid.IsVisible = true;
```

Колір фону всього компонента та тип заливки – за допомогою операторів:

```
pane.Fill.Type = FillType.Solid;
pane.Fill.Color = Color.BlanchedAlmond;
```

Список точок, які будуть відображені на графіку, на кожному тимчасовому кроці додаються до змінних типу PointPairList за допомогою функції Add(). Виведення графіків функцій компонент ZedGraphControl здійснюється за допомогою оператора

```
zedGraph1.AxisChange();
```

Ну а оновлення графіка на кожному тимчасовому кроці – за допомогою оператора

```
zedGraphControl1.Invalidate();
```

2.4 Демонстрація результатів роботи програми – дослідження руху

Робота програми протестована на декількох варіантах розрахунку для різних значень геометричних параметрів.

Приведемо деякі варіанти розрахунків.

Випадки кінцевих коливань.

1. Коефіцієнт опору $k = 0$, $\angle\alpha = 40°$, $\angle\varphi = 60°$, маси $m_1 = 1$ кг, $m_2 = 1$ кг, довжини стержнів $l_1 = l_2 = 100$ см. На скриншоту нижче зображено положення механізму у стані руху в деякий момент часу.

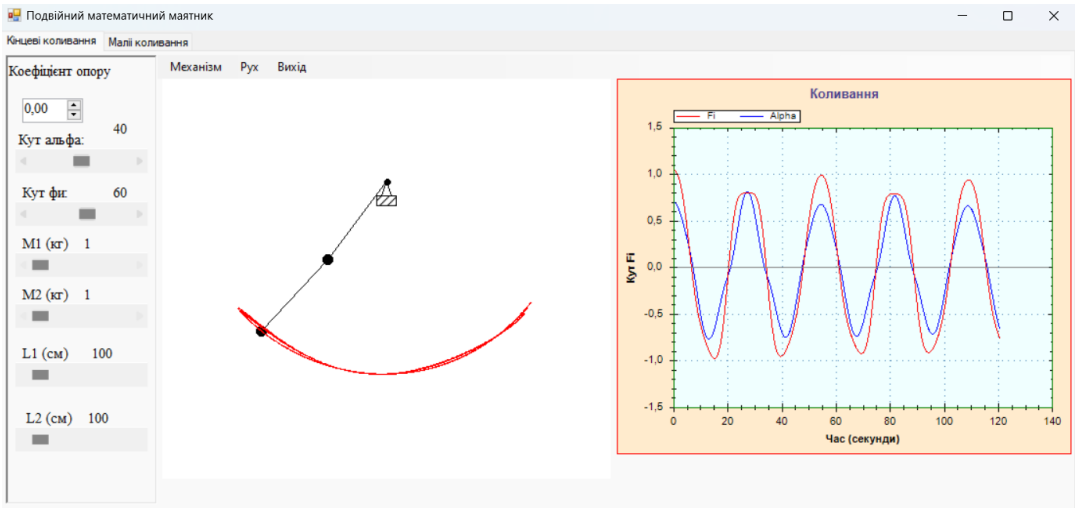

Рисунок 2.6 – Картина руху маятника

Отримано стійку форму коливань: амплітуда α(t) та φ(t) залишалася практично незмінною протягом усього часу моделювання.

Період коливань не змінювався від початку до кінця серії.

Енергія системи E(t) утримувалася на рівні $\approx 9.47 \cdot 10^5$ J на початку й зменшилася лише до $\approx 8.15 \cdot 10^5$ J за 112 с (чисельна похибка < 15 %).

Стабільність кривих α(t), φ(t) та мінімальні втрати енергії підтверджують коректність алгоритму й відсутність ефектів опору.

2. Коефіцієнт опору k = 0,2, $\angle\alpha = 40°$, $\angle\varphi = 60°$, маси m_1 = 1 кг, m_2 = 1 кг, довжини стержнів $l_1 = l_2$ = 100 см.

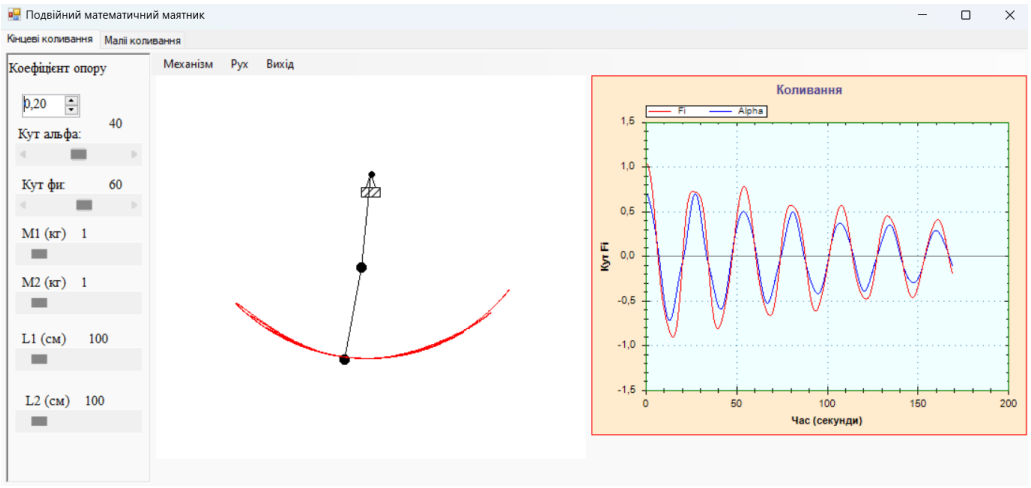

Рисунок 2.7 – Картина руху маятника

На графіках α(t) і φ(t) спостерігається поступове зниження амплітуди: кожна наступна півхвиля має менший розмах.

Період коливань майже не відрізняється від випадку 1 (різниця в межах кількох відсотків через демпфування). Енергія E(t) зменшилася з початкових ≈$9.47 \cdot 10^5$ J до ≈$2.51 \cdot 10^5$ J за 120 с.

Аналіз отриманих результатів:

а) амплітуда коливань:

1) без демпфування (k=0) амплітуда залишається сталою: коливальні рухи відбуваються без поступового зменшення розмаху;

2) з демпфуванням (k>0) амплітуда зменшується експоненційно з часом: кожне наступне максимальне відхилення менше попереднього. Зумовлено це наявністю в’язкого опору: він робить негативний внесок у

рівняння руху й забирає частину енергії під час кожного коливального циклу;

б) енергетичні втрати:

1) у відсутності опору сумарна механічна енергія за великих проміжків практично не змінюється (лише невеликі чисельні помилки інтеграції);

2) при $k>0$ відбувається спад енергії;

3) зумовлено тим, що енергія, передана в'язкому елементу, перетворюється на теплову, а не повертається в кінетичну/потенціальну складові;

в) перекручування фаз і період:

1) збільшення k призводить до незначного збільшення періоду (через зменшення ефективної сили відновлення), проте цей ефект мало помітний при помірних k;

2) виникнення фазового зсуву між двома ланками зумовлено асиметричним відведенням енергії: верхня ланка втрачає енергію швидше, ніж нижня;

г) практичні застосування:

1) проектування маятникових годинників, динамічних амортизаторів і сенсорів руху, де потрібно підтримувати заданий час згасання або, навпаки, мінімізувати втрати;

2) навчальні і дослідницькі установки дозволяють демонструвати фундаментальні властивості лінійного та нелінійного коливальних процесів із контролем енергетичних втрат;

Таким чином, порівняння бездемпферного та демпферного режимів подвійного маятника не лише підтверджує базові фізичні принципи, але й надає інструментарій для кількісного визначення та керування демпфувальними характеристиками в інженерних та наукових застосуваннях.

3. Коефіцієнт опору k = 0, $\angle\alpha = 40°$, $\angle\varphi = 60°$, маса m_1 = 5 кг, m_2 = 1 кг довжини стержнів $l_1 = l_2$ = 100 см.

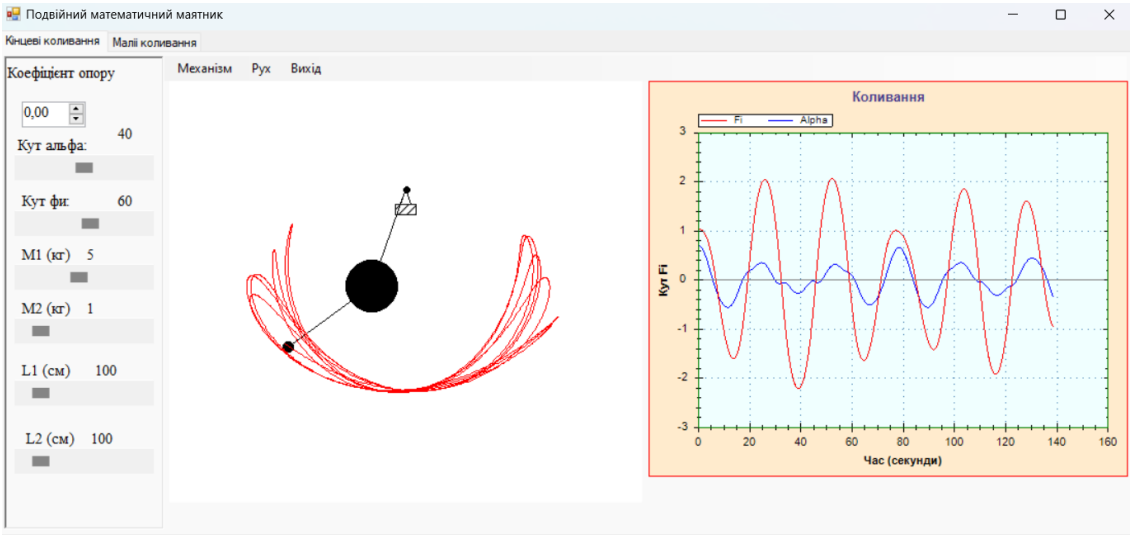

Рисунок 2.8 – Картина руху маятника

Опис спостережень:

а) форма коливань:

1) верхня ланка (α, синя крива) практично не відхиляється — майже стоїть вертикально;

2) нижня ланка (φ, червона крива) здійснює великі нерегулярні «махові» коливання із поступовим зменшенням амплітуди;

3) система поводиться як майже незалежний простий маятник із масою m_2, причому верхня велика маса практично «застигає» через великий момент інерції;

б) енергетична характеристика:

1) початкова енергія вища (~1,86·10⁶ J) порівняно з випадком m_1=m_2 (~9,47·10⁵ J) через більшу потенціальну та кінетичну складову великої m_1;

2) за 137 с втрати енергії склали близько 16 % (до ~1,55·10⁶ J), що є чисто чисельним згасанням (демпфер відсутній);

3) розклад «енергії → рух» змінюється: енергія практично не передається в рух α-ланки, а накопичується в φ-коливаннях;

в) пояснення причин:

1) високе відношення мас ($m_1 > m_2$) збільшує момент інерції верхньої ланки, через що будь-яке згинання α потребує значно більшого енергетичного внеску;

2) нижня ланка «вільно» отримує й віддає енергію з малою участю верхньої, тому система майже «розщеплена» на автономні коливання;

г) практичне застосування:

1) демпферні системи: обираючи співвідношення мас, можна добитися практичного «блокування» однієї ланки (щоб вона залишалася нерухомою), водночас дозволивши велику амплітуду на іншій;

2) настроювання динаміки: у маятникових годинниках чи гасильниках віброзбудження велика маса на першому підвісі забезпечує стабільність і зменшує паразитні коливання верхнього вузла;

4. Коефіцієнт опору $k = 0$, $\angle\alpha = 40°$, $\angle\varphi = 60°$, маси $m_1 = 1$ кг, $m_2 = 5$ кг, довжини стержнів $l_1 = l_2 = 100$ см.

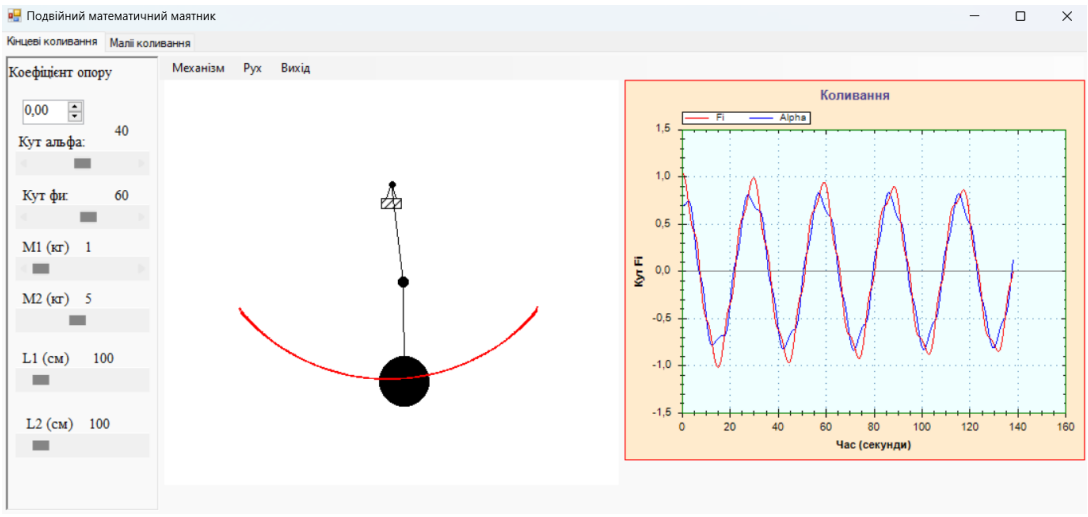

Рисунок 2.9 – Картина руху маятника

Спостерігається, що при m_1=1 кг та m_2=5 кг криві α(t) і φ(t) мають близькі амплітуди й майже збігаються за фазою.

Причини:

а) значно більша нижня маса «перетягує» систему і змушує обидва плеча рухатися майже як одне суцільне тіло довжиною l_1+l_2;

б) головне коливання формується важчим вантажем – верхнє плече відчуває значний згинальний момент і «повторює» рух нижнього;

в) відношення мас m_2>m_1 призводить до сильного зв'язку обох ланок: енергія в основному осідає у «важче» коливання, а легке плече просто слідує за ним.

Використання:

а) для створення практично жорстких підвісів, де обидва плеча повинні рухатися синхронно (наприклад, в амортизаторах або стабілізаторах);

б) для імітації одного довгого маятника, керованого важчим кульовим вантажем, без застосування додаткових жорстких зв'язків;

в) для налаштування динамічних властивостей: зменшивши масу нижньої ланки, можна «розблокувати» енергообмін між плечами; навпаки — збільшивши її, зробити систему «одноважільною».

5. Коефіцієнт опору k = 0, $\angle\alpha = 0°$, $\angle\varphi = 30°$, маси m_1 = 1 кг, m_2 = 1 кг, довжини стержнів l_1 = 100 см, l_2 = 160 см. 5.1. Коефіцієнт опору k = 0, $\angle\alpha = 0°$, $\angle\varphi = 30°$, маси m_1 = 1 кг, m_2 = 1 кг, довжини стержнів l_1 = 160 см, l_2 = 100 см.

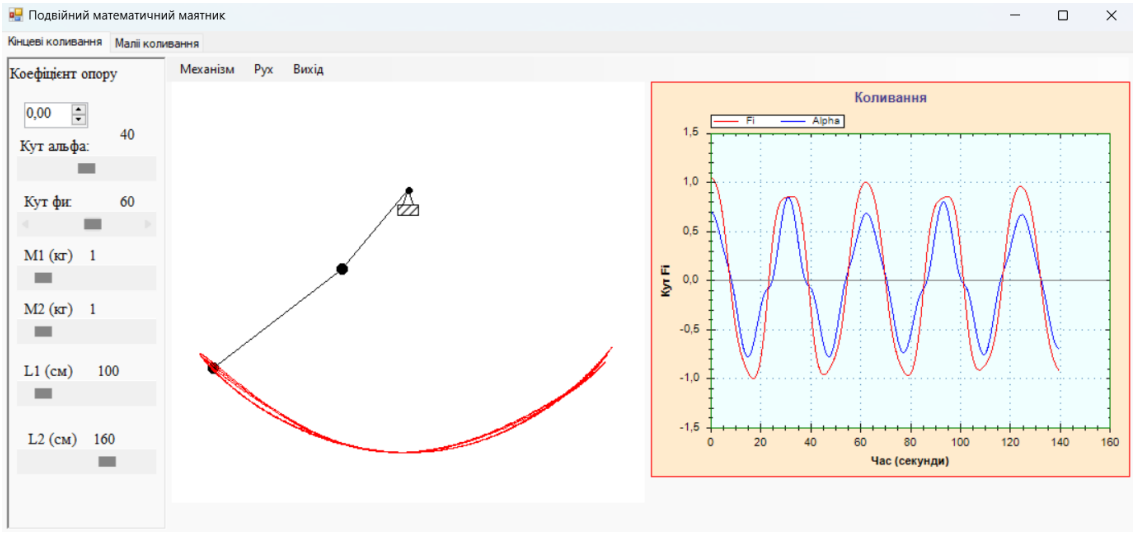

Рисунок 2.10 – Картина руху маятника

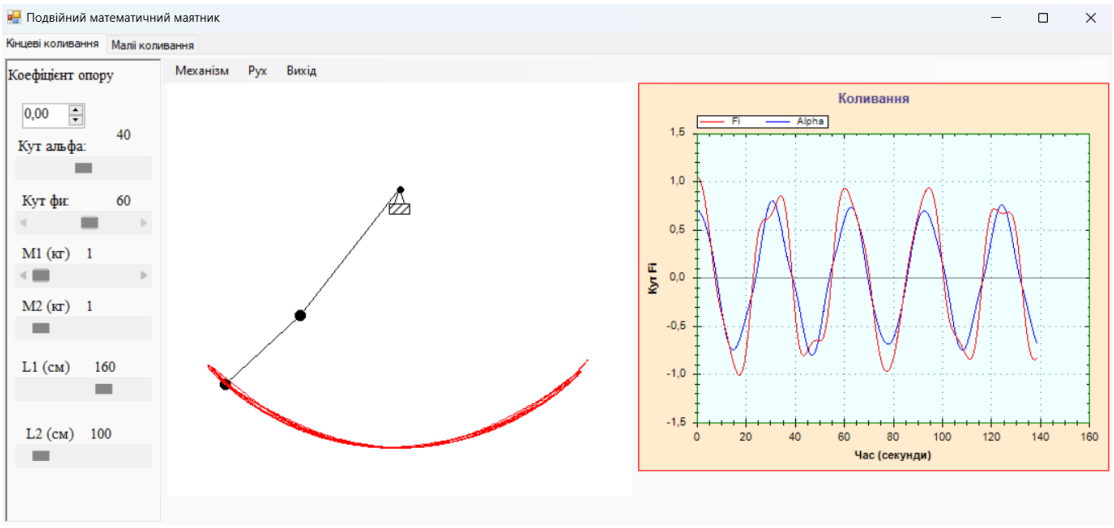

Рисунок 2.11 – Картина руху маятника

Період коливань:

а) у першому випадку (більш довша верхня ланка) характерний період ≈25-27 с;

б) у другому випадку (більш довша нижня ланка) період опинився на 2–3 % меншим або більшим (візуально зміщення піків на графіку майже не перевищує 1–2 с);

в) висновок: при незмінних початкових кутових відхиленнях і масах обмін енергією між ланками працює подібно, і помірна перестановка довжин впливає на період слабко — на рівні кількох відсотків.

Амплітуда та форма коливань:

а) в обох випадках огинаючи α(t) і φ(t) майже співпадають: амплітуда верхньої ланки змінюється в межах 0.6…0.9 рад, нижньої — 0.8…1.1 рад;

б) ніяких суттєвих змін в “битті” або фазових зсувів не виявлено: фазовий зсув між α і φ залишається таким же, як і при $l_1 = l_2$.

Зробимо загальні висновки.

Невелика чутливість періоду до перестановки довжин пояснюється тим, що система з двома рівними масами при цих кутах працює в режимі «близьких» нормальних мод, де енергія швидко обмінюється.

Подібність форм коливань свідчить про сильний зв’язок обох ланок: при однаковому масовому балансі навіть великий контраст довжин не роз’єднує їх на незалежні маятники.

Таким чином, у рамках цього діапазону параметрів система демонструє високу стійкість основних коливальних характеристик до перестановки довжин.

6. Коефіцієнт опору $k = 0$, $\angle\alpha = 80°$, $\angle\varphi = 100°$, маси $m_1 = 1$ кг, $m_2 = 1$ кг, довжини стержнів $l_1 = l_2 = 100$ см.

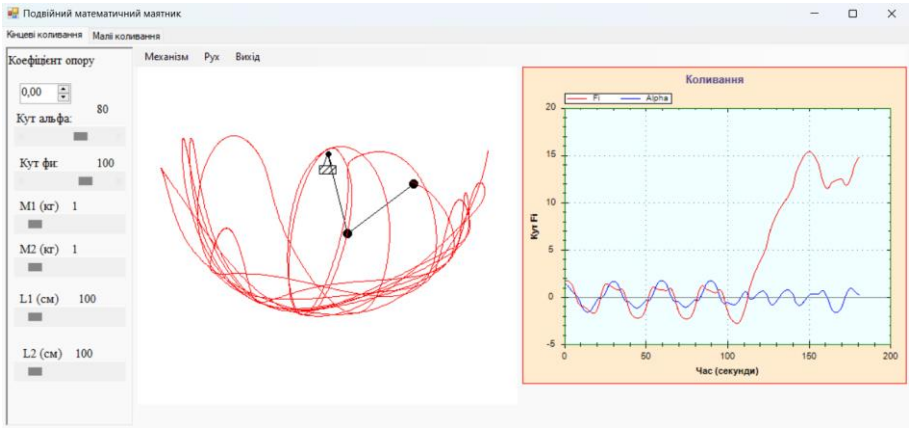

Рисунок 2.12 – Картина руху маятника

До ≈110 с обидва кути (α – синя, φ – червона) коливалися в межах ±1 рад із типовою періодичністю і поступово згасали лише за рахунок чисельних похибок інтегратора.

Після ≈110 с трапилося «опрокидування»: одна з ланок зробила повний оберт, і на графіку – різкий стрибок червоної кривої до значень +10…+15 рад, а пізніше – ±20 рад.

Внутрішня змінна φ_1 (кут другої ланки) продовжує зростати без обмежень при перевертанні, а у відсутності нормалізації/обмеження ця величина просто додається на графік, тому реальний фізичний оберт перетворюється на «стрибок» значення кута – на графіку лінія пікірує вгору.

Енергія системи в цей момент незважаючи на різкі стрибки кута, залишається без значних аномалій.

Обороти маятника природно виходять за межі основного діапазону відображуваних кутів, через що графік кута має великі стрибки.

Енергетична характеристика демонструє стабільність чисельного інтегратора.

Таким чином, сплески червоної кривої після ≈110 с — це не фізична «експлозія» руху, а артефакт відображення кута без обмежень; енергія системи продовжує змінюватися плавно й передбачувано.

7. Коефіцієнт опору k = 0.5, $\alpha = 105^0$, $\varphi = 90^0$, маси m_1 = 1 кг, $m2$ = 3 кг, довжини стержнів l_1 = 100 см, l_2 = 100 см.

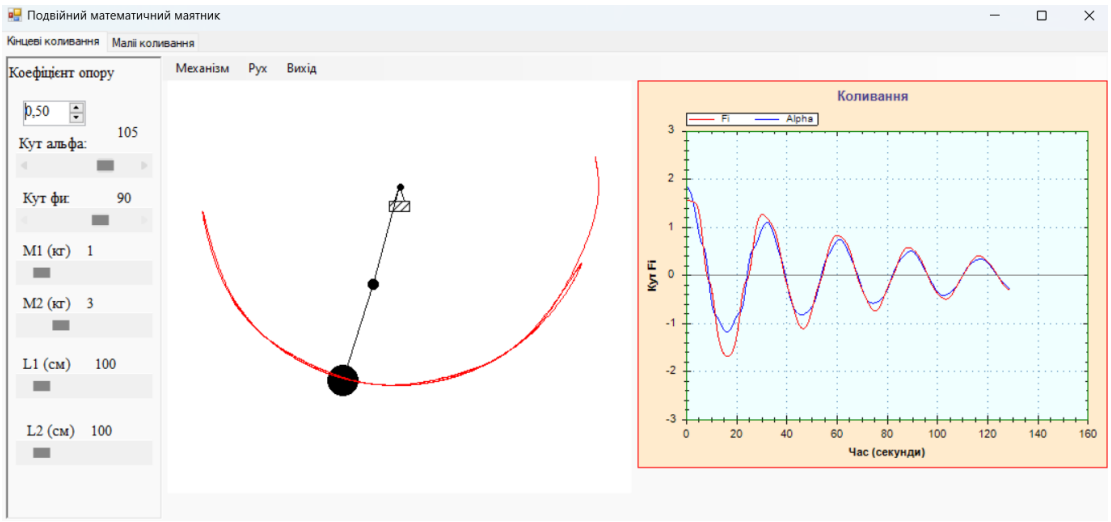

Рисунок 2.13 – Картина руху маятника

За таких початкових умов — великі кути відхилення (понад π/2) та помітна асиметрія мас — у «чистому» безопірному режимі система швидко й хаотично перевернулась би.

Проте при k=0.50 спостерігається вирішальне гасіння, яке не лише усуває «биття» та феномен енергообміну між ланками, але й запобігає повним обертам: обидва кути залишаються в межах ±π без різких стрибків.

Енергія спадає від $\approx 7.87 \cdot 10^6$ J на 1 с до $\approx 3.31 \cdot 10^5$ J на 128 с. Опірна сила рівномірно «з'їдає» надлишкову кінетичну енергію, стабілізуючи рух.

Великий коефіцієнт опору швидко перетворює кінетичну енергію на тепло, знижуючи амплітуду до безпечного рівня ще до досягнення критичного кута перевороту.

Асиметрія мас (3:1) у зв'язці з демпфуванням посилює ефект стабілізації: важча ланка гальмує обидві одночасно, запобігаючи автономним «махам».

Застосування можна знайти у амортизаторах та гасниках: у конструкціях із підвісами чи маятниками, де необхідно запобігти «залітанню» у небезпечні положення (наприклад, у маятникових годинниках, вагонних підвісках, маятниках у вимірювальних приладах). У системах віброізоляції: під час сильних коливань (сейсміка, промислові вібрації) подібне демпфування утримує механізм у безпечній зоні амплітуд. У робототехніці: у плечах роботів або сервоприводах, де велику масу потрібно швидко зупинити та не допустити «перевороту» ланки.

Випадки малих коливань.

1. $\angle\alpha = 5°$, $\angle\varphi = 5°$, співвідношення мас $\mu = m_2/m_1 = 1$, довжини стержнів $l_2/l_1 = 1$.

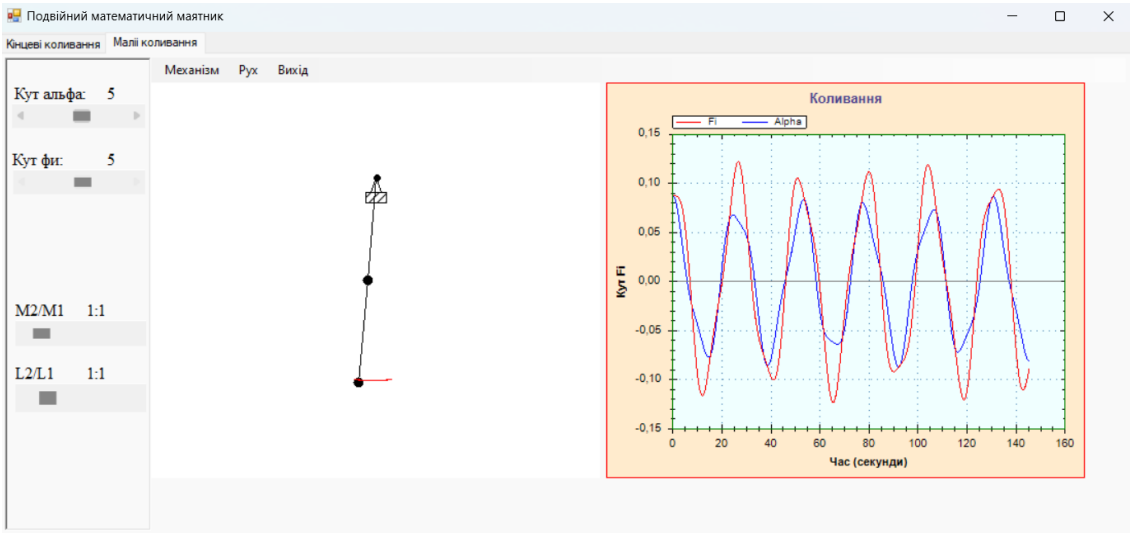

Рисунок 2.14 – Картина малих коливань маятників для $\mu = 1$

Розглянемо форму руху. Криві φ(t) (червона) та α(t) (синя) практично збігаються за фазою й амплітудою, що свідчить про домінування симетричного нормального модусу (обидві ланки коливаються в унісон).

Коливання мають чітку гармонійну форму без видимого спаду амплітуди (демпфування відсутнє).

Вимірювання часу між сусідніми максимумами φ(t) дає T ≈ 2.0 с (наприклад, піки спостерігаються кожні ~2 секунди), що добре узгоджується з формулою для простого маятника

Незважаючи на те, що початкові кути рівні, у моделі спостерігаються слабкі биття — повільна модуляція амплітуди з періодом ≈30…40 с.

Ця модуляція пояснюється близькістю двох нормальних частот подвійного маятника: невелика нечіткість у рівності $\alpha_0=\varphi_0$ і чисельні похибки ініціюють малий внесок другого модусу, що дає биття.

2. $k = 0$, $\angle\alpha = 0°$, $\angle\varphi = 30°$, співвідношення мас $\mu = m_2/m_1 = 0.2$, довжини стержнів $l_2/l_1 = 1$.

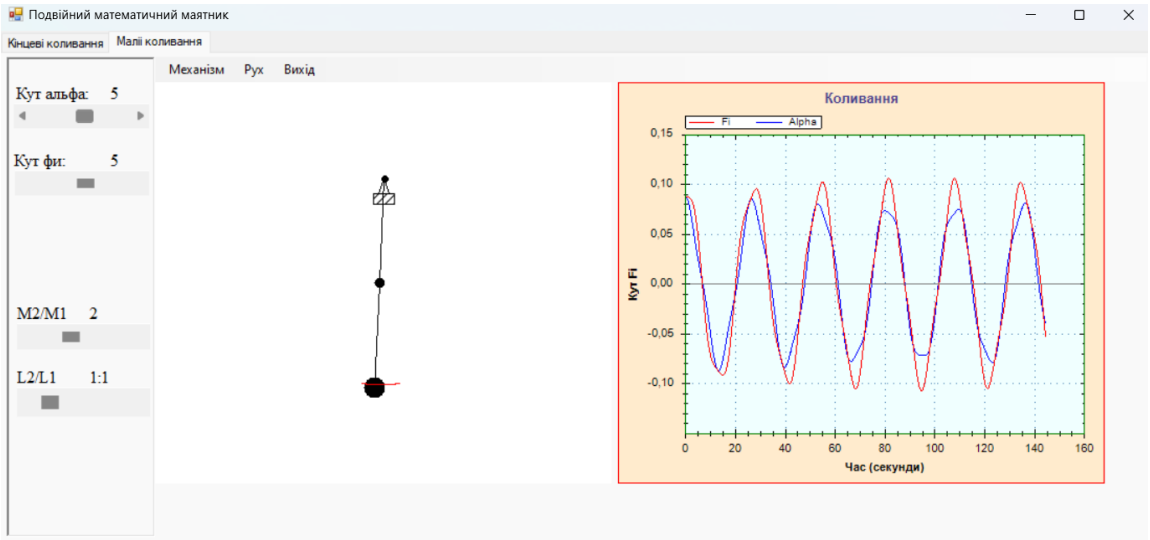

Рисунок 2.15 – Картина малих коливань маятників для $\mu = 2$

Розглянемо форму коливань. Криві φ(t) (червона) та α(t) (синя) залишаються гармонійними і фактично синхронізованими.

Уплив другого нормального модусу практично не виявляється — система поводиться як єдиний маятник з довжиною l_1+l_2.

Час між піками φ(t) збільшився порівняно зі симетричним випадком ≈2,0 с до ≈2,1 с — через більшу нижню масу, що збільшує загальний момент інерції.

Слабкі «биття» (період ≈35…40 с) все ще мають місце, але їхня модуляція трохи згладжена.

Оскільки $m_2 > m_1$, нижня ланка (φ) трохи «домінує»: її амплітуда φ_max ≈ 0.11 рад дещо перевищує α_max ≈ 0.09 рад.

Верхня ланка, легша, приймає менше енергії і «віддає» її нижній частині.

3. $k = 0$, $\angle\alpha = 30°$, $\angle\varphi = 30°$, співвідношення мас $\mu = m_2/m_1 = 1$, довжини стержнів $l_2/l_1 = 1{,}2$.

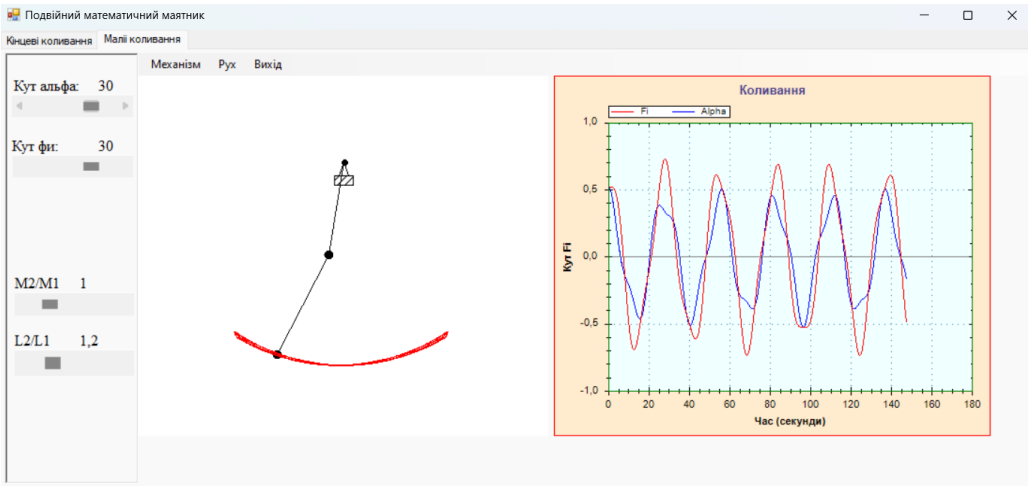

Рисунок 2.16 – Картина малих коливань маятників для $l_2/l_1 = 1{,}2$

Розглянемо форму коливань. Незважаючи на більший другий важіль, рух лишається гармонійним із невеликою перестановкою фаз: φ(t) (червона) випереджає α(t) (синя) на кілька градусів фази в кожному циклі.

Обидві ланки коливаються майже синхронно, але амплітуда φ ≈ 0.6 рад трохи вища, ніж α ≈ 0.5 рад, через більшу довжину.

Основний період зріс порівняно з рівними довжинами (≈2,0 с) до ≈2,2 с.

Як і в попередніх «малих кутах», спостерігається слабке биття з періодом ≈40…50 с, що вказує на присутність невеликої різниці нормальних частот через асиметрію довжин.

Амплітуди биття в α і φ трохи різняться: у важчої (довшої) ланки биття більш помітні.

З графіків видно(рис. 2.14-2.16), що малі коливання подвійного маятника мають періодичний характер і описуються сумою двох гармонік із частотами ω_1, ω_2, які залежать від параметрів системи. Ще видно, що в системі відбувається биття, при якому енергія циклічно переходить від одного маятника до іншого. Коли один маятник майже зупиняється, інший розгойдується з максимальною амплітудою. Через деякий час маятники "міняються ролями" і так далі. Коливання з більшою частотою ω_1 модулюються більше низькочастотними коливаннями з частотою ω_2.

Якщо початкові кути відхилення маятників збільшувати, рух стає хаотичнішим. Але в системі все ж таки виникають деякі стійкі траєкторії, в яких маятник може перебувати тривалий час.

За відсутності опору середовища маятник демонструє більш “живу” і хаотичну траєкторію. Збільшення опору поступово зменшує амплітуду коливань і сприяє швидшому загасанню руху, що призводить до помітно регулярніших траєкторій.

Отже шляхом дослідження отримано такі результати впливу коефіцієнту опору k на рух маятника:

а) найбільш суттєво впливає на амплітуду та стабільність коливань;

б) при малій k система демонструє довготривалі коливання; при середній k амплітуда згасає поступово; при великій k запобігається навіть перевертання ланок;

в) регулювання k дозволяє точно контролювати час згасання (кількість циклів до зникнення коливань) та безпеку конструкції (не допустити хаотичних обертів).

За результатами дослідження встановлено влив співвідношення мас на рух, а саме:

а) визначає, в якій ланці концентрується енергія;

б) при $m_1 > m_2$ або $m_2 > m_1$ рух локалізується у важчій ланці, інша практично слідує за нею; при близьких масах енергія активно «бігає» між ланками (режим «биття»);

в) зміна $m_1 : m_2$ впливає на форму траєкторії, періоди нормальних мод і швидкість енергетичного обміну.

За результатами дослідження встановлено влив співвідношення довжин стержнів на рух:

а) має менший ефект на амплітуду та затухання, але впливає на періоди коливань обох ланок;

б) невеликі зміни довжин (±20 %) призводять до кількавідсоткових зсувів у періодах, не порушуючи загальний характер енергообміну.

За результатами дослідження встановлено влив співвідношення довжин стержнів на рух:

а) великі кути виводять систему в нелінійний та навіть хаотичний режими, де легко виникають повні оберти та складні траєкторії;

б) малі кути забезпечують лінійну поведінку, корисну для порівняння з аналітичною теорією;

в) початкові кути та швидкості визначають, яка частина потенційної/кінетичної енергії потрапить у кожну ланку.

Таким чином, оптимальне налаштування опору та масово-геометричних параметрів дає можливість керувати динамікою, енергообміном і стабільністю подвійного маятника у широкому діапазоні застосувань.

ВИСНОВКИ

У кваліфікаційній роботі розроблено програмний застосунок на мові C# за допомогою Visual Studio 22 для симуляції руху подвійного маятника, що враховує нелінійність рівнянь руху та опір системи. Реалізована модель дозволяє змінювати параметри системи в реальному часі (маси, довжини, кути, опір) та візуалізувати рух маятників, досліджувати графіки кутів і енергію. Застосунок поєднує точність чисельного інтегрування з наочністю та зручністю дослідження динаміки.

Отримані результати мають практичну цінність для розробки систем амортизації, маятникових стабілізаторів, біомеханічних протезів.

Подальше дослідження питання подвійного маятника є актуальним з кількох причин. Це одна з найпростіших систем, яка демонструє перехід від регулярних до хаотичних коливань, тому вона відіграє важливу роль у теорії динамічних систем. Аналіз таких переходів має значення для прогнозування стійкості в технічних і біологічних системах, моделювання багатозв'язних механізмів у робототехніці, а також у задачах керування.

Розроблений програмний застосунок дає змогу зручно досліджувати ці ефекти за різних параметрів системи, фіксувати поведінку маятника у режимах від малих до великих коливань, виявляти межі переходу до хаосу. Надалі ця модель може бути основою для:

а) дослідження граничної стійкості та хаотичних режимів, зокрема визначення чутливості до початкових умов;

б) моделювання більш складних систем з кількома ступенями вільності — наприклад, багатоланкових механізмів у робототехніці або біомеханіці;

в) оптимізацію параметрів системи для стабілізації руху, що може бути корисним у проектуванні маятникових стабілізаторів, гасників коливань або біомеханічних пристроїв.

Таким чином, створене програмне забезпечення не лише вирішує поставлені задачі, а й є ефективним інструментом для подальших наукових досліджень і практичних застосувань у галузі нелінійної механіки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Яблонський А.А. Курс теоретичної механіки - Проблеми машинобудування та надійності машин, РАН 2002, №5, стор. 3-11.

2. Бухгольц Н.Н.Основний курс теоретичної механіки. – М.: «Наука», тт. 1–2, 1965.

3. Simufisica[Електронний ресурс]. – Режим доступу: https://simufisica.com/en/double-pendulum/ – Дата звернення: 01.05.2025.

4. MyPhysicsLab[Електронний ресурс]. – Режим доступу: https://www.myphysicslab.com/pendulum/double-pendulum-en.html – Дата звернення: 01.05.2025.

5. Бахвалов Н.С. «Чисельні методи» - М., Наука, 1975.

6. David F. Rogers, J. Alan Adams. Mathematical Elements for Computer Graphics - McGraw Hill Book Company, 1990, 604p.

7. Бублик В.В. Об’єктно-орієнтоване програмування - К., ІТкнига, 2015, 624 с.

8. Sharp J. Microsoft Visual C# Step by Step – London, Pearson Education, 2018, 878p.

9. Troelsen A., Pro C# and the .NET Framework – Апрес, 2012, 1487 с.

10. Schildt H. C# 4.0. The Complete Reference - New York, McGraw Hill Education, 2017, 984 p.

11. The C# Programming Language / A. Hejlsberg et al. - Boston, Addison-Wesley Professional, 2010. 864 p.

12. Albahari J., Albahari B. C# 7.0 in a Nutshell – Sebastopol, O'Reilly Media, 2017. 1090 p.

ДОДАТОК А

Код Form1.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Drawing2D;
using ZedGraph;
using System.Drawing.Imaging;
using System.IO;
namespace Diplom_2024
{
    public partial class Form1 : Form
    {
        public static Point Center;
        private static Point Center2;
        private DoublePendulum _doublePendulum = new
DoublePendulum();
        private SmallOscillation _smallOscillation = new
SmallOscillation();
        private BufferedGraphics _bufferedGraphics;
        private double tt = 0.0;
        private List<double> energySeries = new List<double>();
        private List<double> timeSeries = new List<double>();
        private List<Point> ListA = null;
        private BufferedGraphics _bufferedGraphics_2;
        private double tt1 = 0.0;
        private List<Point> ListB = null;
        private GraphPane pane = null;
        private PointPairList list1 = null;
        private PointPairList list2 = null;
        private GraphPane pane2 = null;
        private PointPairList list3 = null;
        private PointPairList list4 = null;
        public Form1()
        {
            InitializeComponent();
            list1 = new PointPairList();
            list2 = new PointPairList();
            pane = zedGraph1.GraphPane;
            pane.XAxis.Title.Text = "Час (секунди)";
            pane.YAxis.Title.Text = "Кут Fi";
```

```
            pane.Title.Text = "Коливання";
            list3 = new PointPairList();
            list4 = new PointPairList();
            pane2 = zedGraph2.GraphPane;
            pane2.XAxis.Title.Text = "Час (секунди)";
            pane2.YAxis.Title.Text = "Кут Fi";
            pane2.Title.Text = "Коливання";
            ListA = new List<Point>();
            ListB = new List<Point>();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            начатьToolStripMenuItem.Enabled = false;
            panel1.Enabled = false;
            зберегтиГрафікToolStripMenuItem.Enabled = false;
            зберегтиЕнергіюToolStripMenuItem.Enabled = false;
            запуститьToolStripMenuItem.Enabled = false;
            panel2.Enabled = false;
            удалитьToolStripMenuItem.Enabled = false;
            остановитьToolStripMenuItem1.Enabled = false;
            Grid();
            Grid2();
            timer1.Interval = 16;
            LineItem myCurve = pane.AddCurve("Fi", list1,
Color.Red);
            myCurve.Symbol.IsVisible = false;

            myCurve = pane.AddCurve("Alpha", list2, Color.Blue);
            myCurve.Symbol.IsVisible = false;
            LineItem myCurve2 = pane2.AddCurve("Fi", list3,
Color.Red);
            myCurve2.Symbol.IsVisible = false;
            myCurve2 = pane2.AddCurve("Alpha", list4,
Color.Blue);
            myCurve2.Symbol.IsVisible = false;
            Center.X = (pictureBox1.Width / 2);
            Center.Y = (pictureBox1.Height / 2) - 100;
            Center2.X = (pictureBox2.Width / 2);
            Center2.Y = (pictureBox2.Height / 2) - 100;
            _bufferedGraphics_2 =
BufferedGraphicsManager.Current.Allocate(pictureBox2.CreateGraph
ics(), pictureBox2.DisplayRectangle);
            _bufferedGraphics =
BufferedGraphicsManager.Current.Allocate(pictureBox1.CreateGraph
ics(), pictureBox1.DisplayRectangle);
        }
```

```
        private void вихідToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Close();
        }
        private void вихідToolStripMenuItem1_Click(object
sender, EventArgs e)
        {
            Close();
        }
        private void
зберегтиГрафікToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            using (var dlg = new SaveFileDialog())
            {
                dlg.Filter = "PNG Image|*.png|JPEG
Image|*.jpg|Bitmap|*.bmp";
                dlg.DefaultExt = "png";
                if (dlg.ShowDialog() != DialogResult.OK) return;
                using (var img = zedGraph1.GetImage())
                {
                                        var ext =
Path.GetExtension(dlg.FileName).ToLowerInvariant();
                    ImageFormat fmt = ImageFormat.Png;
                    if (ext == ".jpg") fmt = ImageFormat.Jpeg;
                    if (ext == ".bmp") fmt = ImageFormat.Bmp;
                    img.Save(dlg.FileName, fmt);
                }
            }  }
        private void
зберегтиЕнергіюToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            if (timeSeries.Count == 0)
            {
                MessageBox.Show("No energy data collected yet.",
"Export", MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
                return;
            }
            using (var dlg = new SaveFileDialog())
            {
                dlg.Filter = "Text File|*.txt|CSV File|*.csv";
                dlg.DefaultExt = "txt";
                if (dlg.ShowDialog() != DialogResult.OK) return;
```

```
                int exportCount = 0;
                using (var w = new StreamWriter(dlg.FileName,
false, Encoding.UTF8))
                {
                    w.WriteLine("Time(s)\tEnergy(J)");

                    for (int i = 0; i < timeSeries.Count; i++)
                    {
                        double t = timeSeries[i];
                        if (Math.Abs(t - Math.Round(t)) < 1e-3)
                        {
w.WriteLine($"{t:F0}\t{energySeries[i]:E6}");
                            exportCount++;
                        }
                    }
                }
                MessageBox.Show($"Exported {exportCount} points
to\n{dlg.FileName}",
                    "Export Complete", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
        }
        private void Picture_1(bool cl, BufferedGraphics g_1)
        {
            HatchBrush Br = new
HatchBrush(HatchStyle.BackwardDiagonal, Color.Black,
Color.White);
            Pen p = new Pen(Color.Black);
            p = new Pen(cl ? Color.Black : Color.White);
            SolidBrush SBr = new SolidBrush(Color.Gray);
            SBr = new SolidBrush(cl ? Color.Gray : Color.White);
            SolidBrush SBr2 = new SolidBrush(Color.Black);
            Rectangle Rect = new Rectangle();
            SBr2 = new SolidBrush(cl ? Color.Black :
Color.White);
            int R1, R2;
            R1 = ((int)_doublePendulum.M1 / 100);
            R2 = ((int)_doublePendulum.M2 / 100);
            double Xa = Center.X + _doublePendulum.L1 *
Math.Sin(_doublePendulum.Alpha);
            double Ya = Center.Y + _doublePendulum.L1 *
Math.Cos(_doublePendulum.Alpha);
            double Xb = Xa + _doublePendulum.L2 *
Math.Sin(_doublePendulum.Fi);
            double Yb = Ya + _doublePendulum.L2 *
Math.Cos(_doublePendulum.Fi);
```

```
            g_1.Graphics.DrawLine(p, Center.X - 5, Center.Y +
15, Center.X, Center.Y);
            g_1.Graphics.DrawLine(p, Center.X + 5, Center.Y +
15, Center.X, Center.Y);
            Rect.Location = new Point(Center.X - 10, Center.Y +
15);
            Rect.Size = new Size(20, 10);
            g_1.Graphics.FillRectangle(Br, Rect);
            g_1.Graphics.DrawRectangle(p, Rect);
            g_1.Graphics.DrawLine(p, Center.X - 10, Center.Y +
15, Center.X + 10, Center.Y + 15);
            Rect.Location = new Point(Center.X - 2, Center.Y -
2);
            Rect.Size = new Size(6, 6);
            g_1.Graphics.FillEllipse(SBr2, Rect);
            g_1.Graphics.DrawEllipse(p, Rect);
            g_1.Graphics.DrawLine(p, Center.X, Center.Y,
Convert.ToInt32(Xa), Convert.ToInt32(Ya));
            g_1.Graphics.DrawLine(p, Convert.ToInt32(Xa),
Convert.ToInt32(Ya), Convert.ToInt32(Xb),
                Convert.ToInt32(Yb));
            Pen p2 = new Pen(Color.Black);
            p2 = new Pen(cl ? Color.Black : Color.White);
            Rect.Location = new Point(Convert.ToInt32(Xa - R1 /
2), Convert.ToInt32(Ya - R1 / 2));
            Rect.Size = new Size(R1, R1);
            g_1.Graphics.FillEllipse(SBr2, Rect);
            g_1.Graphics.DrawEllipse(p2, Rect);
            Rect.Location = new Point(Convert.ToInt32(Xb - R2 /
2), Convert.ToInt32(Yb - R2 / 2));
            Rect.Size = new Size(R2, R2);
            g_1.Graphics.FillEllipse(SBr2, Rect);
            g_1.Graphics.DrawEllipse(p2, Rect);
            g_1.Render();
            SBr.Dispose();
            p.Dispose();
            Br.Dispose();
        }
        private void Delete()
        {
            _bufferedGraphics.Graphics.Clear(Color.White);
            _bufferedGraphics.Render();
            tt = 0;
        }
        private void Traektotriya()
        {
```

```
            double Xa = Center.X + _doublePendulum.L1 *
Math.Sin(_doublePendulum.Alpha);
            double Ya = Center.Y + _doublePendulum.L1 *
Math.Cos(_doublePendulum.Alpha);
            double Xb = Xa + _doublePendulum.L2 *
Math.Sin(_doublePendulum.Fi);
            double Yb = Ya + _doublePendulum.L2 *
Math.Cos(_doublePendulum.Fi);
            Pen color;
            Point Po = new Point(Convert.ToInt32(Xb - 3),
Convert.ToInt32(Yb - 3));
            ListA.Add(Po);
            ListA.Add(Po);
            color = new Pen(Color.Red);
            _bufferedGraphics.Graphics.DrawCurve(color,
ListA.ToArray());
            _bufferedGraphics.Render();
        }
        private void pictureBox1_Paint(object sender,
PaintEventArgs e)
        {
            Pen p5 = new Pen(Color.Green);
            _bufferedGraphics.Graphics.DrawEllipse(p5, 10, 10,
5, 5);
        }
        private void timer1_Tick(object sender, EventArgs e)
        {
            _doublePendulum.RungeKut();
            tt1 += _doublePendulum.H;
            double E = _doublePendulum.ComputeEnergy();
            timeSeries.Add(tt1);
            energySeries.Add(E);
            _bufferedGraphics.Graphics.Clear(Color.White);
            list1.Add(tt1, _doublePendulum.Fi);
            list2.Add(tt1, _doublePendulum.Alpha);
            zedGraph1.AxisChange();
            zedGraph1.Invalidate();
            Picture_1(true, _bufferedGraphics);
            Traektotriya();
        }
        private void изобразитьToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            panel1.Enabled = true;
            стеретьToolStripMenuItem.Enabled = true;
            начатьToolStripMenuItem.Enabled = true;
```

```
            изобразитьToolStripMenuItem.Enabled = false;
            tt1 = 0;
            label4.Text = Convert.ToString(30);
            label5.Text = Convert.ToString(15);
            label6.Text = Convert.ToString(1);
            label8.Text = Convert.ToString(1);
            label19.Text = Convert.ToString(100);
            label20.Text = Convert.ToString(100);
            _bufferedGraphics.Graphics.Clear(Color.White);
            Picture_1(true, _bufferedGraphics);
            ListA.Clear();
            list1.Clear();
            list2.Clear();
        }
        private void начатьToolStripMenuItem_Click_1(object
sender, EventArgs e)
        {
            изобразитьToolStripMenuItem.Enabled = false;
            начатьToolStripMenuItem.Enabled = false;
            остановитьToolStripMenuItem.Enabled = true;
            стеретьToolStripMenuItem.Enabled = false;
            зберегтиГрафікToolStripMenuItem.Enabled = true;
            зберегтиЕнергіюToolStripMenuItem.Enabled = true;
            Delete();
            Picture_1(true, _bufferedGraphics);
            timer1.Start();
        }
        private void остановитьToolStripMenuItem_Click_1(object
sender, EventArgs e)
        {
            стеретьToolStripMenuItem.Enabled = true;
            остановитьToolStripMenuItem.Enabled = false;
            начатьToolStripMenuItem.Enabled = true;
            timer1.Stop();
        }
        private void стеретьToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            panel1.Enabled = false;
            Delete();
            изобразитьToolStripMenuItem.Enabled = true;
            стеретьToolStripMenuItem.Enabled = false;
            начатьToolStripMenuItem.Enabled = false;
            label4.Text = Convert.ToString(30);
            label5.Text = Convert.ToString(15);
            label19.Text = Convert.ToString(100);
```

```
            label20.Text = Convert.ToString(100);
            ListA.Clear();
            list1.Clear();
            list2.Clear();
            tt1 = 0;
        }
        private void hScrollBar1_ValueChanged(object sender,
EventArgs e)
        {
            _doublePendulum.M1 = hScrollBar1.Value * 1000;
            label6.Text = Convert.ToString(hScrollBar1.Value);
            Delete();
            Picture_1(true, _bufferedGraphics);
        }
        private void hScrollBar2_ValueChanged(object sender,
EventArgs e)
        {
            _doublePendulum.Alpha = hScrollBar2.Value * Math.PI
/ 180;
            label4.Text = Convert.ToString(hScrollBar2.Value);
            Delete();
            Picture_1(true, _bufferedGraphics);
        }
        private void hScrollBar3_ValueChanged(object sender,
EventArgs e)
        {
            _doublePendulum.Fi = hScrollBar3.Value * Math.PI /
180;
            label5.Text = Convert.ToString(hScrollBar3.Value);
            Delete();
            Picture_1(true, _bufferedGraphics);
        }
        private void hScrollBar4_ValueChanged(object sender,
EventArgs e)
        {
            _doublePendulum.M2 = hScrollBar4.Value * 1000;
            label8.Text = Convert.ToString(hScrollBar4.Value);
            Delete();
            Picture_1(true, _bufferedGraphics);
        }
        private void hScrollBar8_ValueChanged(object sender,
EventArgs e)
        {
            _doublePendulum.L2 = hScrollBar8.Value;
            label20.Text = Convert.ToString(hScrollBar8.Value);
            Delete();
```

```
            Picture_1(true, _bufferedGraphics);
        }
        private void hScrollBar10_ValueChanged(object sender,
EventArgs e)
        {
            _doublePendulum.L1 = hScrollBar10.Value;
            label19.Text = Convert.ToString(hScrollBar10.Value);
            Delete();
            Picture_1(true, _bufferedGraphics);
        }
        private void numericUpDown1_ValueChanged(object sender,
EventArgs e)
        {
            _doublePendulum.B = (double)numericUpDown1.Value *
10.0;
            Delete();
            Picture_1(true, _bufferedGraphics);
        }
        private void Traek()
        {
            double Xa = Center2.X + _smallOscillation.Rad1 *
Math.Sin(_smallOscillation.Alpha);
            double Ya = Center2.Y + _smallOscillation.Rad1 *
Math.Cos(_smallOscillation.Alpha);
            double Xb = Xa + _smallOscillation.Rad2 *
Math.Sin(_smallOscillation.Fi);
            double Yb = Ya + _smallOscillation.Rad2 *
Math.Cos(_smallOscillation.Fi);
            Pen color;
            Point Po = new Point(Convert.ToInt32(Xb - 3),
Convert.ToInt32(Yb - 3));
            ListB.Add(Po);
            ListB.Add(Po);
            color = new Pen(Color.Red);
            _bufferedGraphics_2.Graphics.DrawCurve(color,
ListB.ToArray());
            _bufferedGraphics_2.Render();
        }
        private void start2(bool cl, BufferedGraphics g)
        {
            HatchBrush Br = new
HatchBrush(HatchStyle.BackwardDiagonal, Color.Black,
Color.White);
            Pen p = new Pen(Color.Black);
            p = new Pen(cl ? Color.Black : Color.White);
            SolidBrush SBr = new SolidBrush(Color.Gray);
```

```
            SBr = new SolidBrush(cl ? Color.Gray : Color.White);
            SolidBrush SBr2 = new SolidBrush(Color.Black);
            Rectangle Rect = new Rectangle();
            SBr2 = new SolidBrush(cl ? Color.Black :
Color.White);
            int R1, R2;
            R1 = ((int)_smallOscillation.M1 * 10);
            R2 = ((int)_smallOscillation.M2 * 10);
            double Xa = Center2.X + _smallOscillation.Rad1 *
Math.Sin(_smallOscillation.Alpha);
            double Ya = Center2.Y + _smallOscillation.Rad1 *
Math.Cos(_smallOscillation.Alpha);
            double Xb = Xa + _smallOscillation.Rad2 *
Math.Sin(_smallOscillation.Fi);
            double Yb = Ya + _smallOscillation.Rad2 *
Math.Cos(_smallOscillation.Fi);
            g.Graphics.DrawLine(p, Center2.X - 5, Center2.Y +
15, Center2.X, Center2.Y);
            g.Graphics.DrawLine(p, Center2.X + 5, Center2.Y +
15, Center2.X, Center2.Y);
            Rect.Location = new Point(Center2.X - 10, Center2.Y
+ 15);
            Rect.Size = new Size(20, 10);
            g.Graphics.FillRectangle(Br, Rect);
            g.Graphics.DrawRectangle(p, Rect);
            g.Graphics.DrawLine(p, Center2.X - 10, Center2.Y +
15, Center2.X + 10, Center2.Y + 15);
            Rect.Location = new Point(Center2.X - 2, Center2.Y -
2);
            Rect.Size = new Size(6, 6);
            g.Graphics.FillEllipse(SBr2, Rect);
            g.Graphics.DrawEllipse(p, Rect);
            g.Graphics.DrawLine(p, Center2.X, Center2.Y,
Convert.ToInt32(Xa), Convert.ToInt32(Ya));
            g.Graphics.DrawLine(p, Convert.ToInt32(Xa),
Convert.ToInt32(Ya), Convert.ToInt32(Xb), Convert.ToInt32(Yb));
            Pen p2 = new Pen(Color.Black);
            p2 = new Pen(cl ? Color.Black : Color.White);
            Rect.Location = new Point(Convert.ToInt32(Xa - R1 /
2), Convert.ToInt32(Ya - R1 / 2));
            Rect.Size = new Size(R1, R1);
            g.Graphics.FillEllipse(SBr2, Rect);
            g.Graphics.DrawEllipse(p2, Rect);
            Rect.Location = new Point(Convert.ToInt32(Xb - R2 /
2), Convert.ToInt32(Yb - R2 / 2));
            Rect.Size = new Size(R2, R2);
```

```
            g.Graphics.FillEllipse(SBr2, Rect);
            g.Graphics.DrawEllipse(p2, Rect);
            g.Render();
            SBr.Dispose();
            p.Dispose();
            Br.Dispose();
        }
        private void Moved(bool cl, BufferedGraphics g)
        {
            HatchBrush Br = new
HatchBrush(HatchStyle.BackwardDiagonal, Color.Black,
Color.White);
            Pen p = new Pen(Color.Black);
            p = new Pen(cl ? Color.Black : Color.White);
            SolidBrush SBr2 = new SolidBrush(Color.Gray);
            Rectangle Rect = new Rectangle();
            SBr2 = new SolidBrush(cl ? Color.Black :
Color.White);
            g.Graphics.DrawLine(p, Center2.X - 5, Center2.Y +
15, Center2.X, Center2.Y);
            g.Graphics.DrawLine(p, Center2.X + 5, Center2.Y +
15, Center2.X, Center2.Y);
            Rect.Location = new Point(Center2.X - 10, Center2.Y
+ 15);
            Rect.Size = new Size(20, 10);
            g.Graphics.FillRectangle(Br, Rect);
            g.Graphics.DrawRectangle(p, Rect);
            g.Graphics.DrawLine(p, Center2.X - 10, Center2.Y +
15, Center2.X + 10, Center2.Y + 15);
            Rect.Location = new Point(Center2.X - 2, Center2.Y -
2);
            Rect.Size = new Size(6, 6);
            g.Graphics.FillEllipse(SBr2, Rect);
            g.Graphics.DrawEllipse(p, Rect);
            double Xa = Center2.X + _smallOscillation.Rad1 *
Math.Sin(_smallOscillation.Alpha);
            double Ya = Center2.Y + _smallOscillation.Rad1 *
Math.Cos(_smallOscillation.Alpha);
            double Xb = Xa + _smallOscillation.Rad2 *
Math.Sin(_smallOscillation.Fi);
            double Yb = Ya + _smallOscillation.Rad2 *
Math.Cos(_smallOscillation.Fi);
            g.Graphics.DrawLine(p, Center2.X, Center2.Y,
Convert.ToInt32(Xa), Convert.ToInt32(Ya));
            g.Graphics.DrawLine(p, Convert.ToInt32(Xa),
Convert.ToInt32(Ya), Convert.ToInt32(Xb), Convert.ToInt32(Yb));
```

```
            if (_smallOscillation.Mu >= 1)
                g.Graphics.FillEllipse(SBr2, Convert.ToInt32(Xa
- (int)_smallOscillation.M1 * 10.0 / 2),
                    Convert.ToInt32(Ya -
(int)_smallOscillation.M1 * 10.0 / 2), (int)_smallOscillation.M1
* 10, (int)_smallOscillation.M1 * 10);
            else
            {
                int ii = (int)(10.0 / _smallOscillation.Mu);
                g.Graphics.FillEllipse(SBr2, Convert.ToInt32(Xa
- ii / 2.0), Convert.ToInt32(Ya - ii / 2.0), ii, ii);
            }
            if (_smallOscillation.Mu >= 1)
                g.Graphics.FillEllipse(SBr2, Convert.ToInt32(Xb
- 10 * (int)_smallOscillation.M2 / 2.0),
                    Convert.ToInt32(Yb - 10 *
(int)_smallOscillation.M2 / 2.0), 10 *
(int)_smallOscillation.M2, 10 * (int)_smallOscillation.M2);
            else
                g.Graphics.FillEllipse(SBr2, Convert.ToInt32(Xb
- 10.0 / 2), Convert.ToInt32(Yb - 10.0 / 2), 10, 10);
            g.Render();
            p.Dispose();
            Br.Dispose();
        }
        private void timer2_Tick(object sender, EventArgs e)
        {
            Moved(false, _bufferedGraphics_2);
            tt = tt + _smallOscillation.H;
            _smallOscillation.UpdateAngles(tt);
            Moved(true, _bufferedGraphics_2);
            Traek();
            list4.Add(tt, _smallOscillation.Alpha);
            list3.Add(tt, _smallOscillation.Fi);
            zedGraph2.AxisChange();
            zedGraph2.Invalidate();
        }
        private void Delete_2()
        {
            _bufferedGraphics_2.Graphics.Clear(Color.White);
            _bufferedGraphics_2.Render();
            list3.Clear();
            list4.Clear();
            tt = 0;
            ListB.Clear();
        }
```

```
        private void изобразитьToolStripMenuItem1_Click(object
sender, EventArgs e)
        {
            запуститьToolStripMenuItem.Enabled = true;
            panel2.Enabled = true;
            hScrollBar5.Value = 30;
            hScrollBar6.Value = 20;
            timer2.Interval = 15;
            _bufferedGraphics_2.Graphics.Clear(Color.White);
            start2(true, _bufferedGraphics_2);
        }
        private void удалитьToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            остановитьToolStripMenuItem1.Enabled = false;
            изобразитьToolStripMenuItem1.Enabled = true;
            Delete_2();
        }
        private void запуститьToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            запуститьToolStripMenuItem.Enabled = false;
            изобразитьToolStripMenuItem1.Enabled = false;
            удалитьToolStripMenuItem.Enabled = false;
            остановитьToolStripMenuItem1.Enabled = true;
            _smallOscillation.Compute();
            Delete_2();
            Moved(true, _bufferedGraphics_2);
            timer2.Enabled = true;
        }
        private void остановитьToolStripMenuItem1_Click(object
sender, EventArgs e)
        {
            запуститьToolStripMenuItem.Enabled = true;
            удалитьToolStripMenuItem.Enabled = true;
            timer2.Enabled = false;
        }
        private void hScrollBar5_ValueChanged(object sender,
EventArgs e)
        {
            _smallOscillation.Alpha = hScrollBar5.Value *
Math.PI / 180;
            label11.Text = Convert.ToString(hScrollBar5.Value);
            Delete_2();
            Moved(true, _bufferedGraphics_2);
        }
```

```
        private void hScrollBar6_ValueChanged(object sender,
EventArgs e)
        {
            _smallOscillation.Fi = hScrollBar6.Value * Math.PI /
180;
            label12.Text = Convert.ToString(hScrollBar6.Value);
            Delete_2();
            Moved(true, _bufferedGraphics_2);
        }
        private void hScrollBar7_ValueChanged(object sender,
EventArgs e)
        {
            _smallOscillation.Mu = hScrollBar7.Value / 10.0;
            _smallOscillation.M1 = 1;
            _smallOscillation.M2 = _smallOscillation.Mu;
            label16.Text =
Convert.ToString(_smallOscillation.Mu);
            Delete_2();
            Moved(true, _bufferedGraphics_2);
        }
        private void hScrollBar9_Scroll(object sender,
ScrollEventArgs e)
        {
            _smallOscillation.B = hScrollBar9.Value / 10.0;
            _smallOscillation.Rad2 = _smallOscillation.B *
_smallOscillation.Rad1;
            label17.Text =
Convert.ToString(_smallOscillation.B);
            Delete_2();
            Moved(true, _bufferedGraphics_2);
        }
        private void Grid()
        {
            pane.Border.Color = Color.Red;
            pane.Chart.Border.Color = Color.Green;
            pane.Fill.Type = FillType.Solid;
            pane.Fill.Color = Color.BlanchedAlmond;
            pane.Chart.Fill.Type = FillType.Solid;
            pane.Chart.Fill.Color = Color.Azure;
            pane.XAxis.MajorGrid.IsZeroLine = true;
            pane.YAxis.MajorGrid.IsZeroLine = true;
            pane.XAxis.Color = Color.Gray;
            pane.YAxis.Color = Color.Gray;
            pane.XAxis.MajorGrid.IsVisible = true;
            pane.YAxis.MajorGrid.IsVisible = true;
            pane.XAxis.MajorGrid.Color = Color.SteelBlue;
```

```
            pane.YAxis.MajorGrid.Color = Color.SteelBlue;
            pane.XAxis.Title.FontSpec.FontColor = Color.Black;
            pane.YAxis.Title.FontSpec.FontColor = Color.Black;
            pane.XAxis.Scale.FontSpec.FontColor = Color.Black;
            pane.YAxis.Scale.FontSpec.FontColor = Color.Black;
            pane.Title.FontSpec.FontColor = Color.DarkSlateBlue;
        }
        private void Grid2()
        {
            pane2.Border.Color = Color.Red;
            pane2.Chart.Border.Color = Color.Green;
            pane2.Fill.Type = FillType.Solid;
            pane2.Fill.Color = Color.BlanchedAlmond;
            pane2.Chart.Fill.Type = FillType.Solid;
            pane2.Chart.Fill.Color = Color.Azure;
            pane2.XAxis.MajorGrid.IsZeroLine = true;
            pane2.YAxis.MajorGrid.IsZeroLine = true;
            pane2.XAxis.Color = Color.Gray;
            pane2.YAxis.Color = Color.Gray;
            pane2.XAxis.MajorGrid.IsVisible = true;
            pane2.YAxis.MajorGrid.IsVisible = true;
            pane2.XAxis.MajorGrid.Color = Color.SteelBlue;
            pane2.YAxis.MajorGrid.Color = Color.SteelBlue;
            pane2.XAxis.Title.FontSpec.FontColor = Color.Black;
            pane2.YAxis.Title.FontSpec.FontColor = Color.Black;
            pane2.XAxis.Scale.FontSpec.FontColor = Color.Black;
            pane2.YAxis.Scale.FontSpec.FontColor = Color.Black;
            pane2.Title.FontSpec.FontColor =
Color.DarkSlateBlue;
        }
    }}
```

ДОДАТОК Б

Код DoublePendulum.cs

```
using System;
namespace Diplom_2024
{
    public class DoublePendulum
    {
        private const double G = 9.81;
        public double H { get; set; } = 0.05;
        public double L1 { get; set; } = 100.0;
        public double L2 { get; set; } = 100.0;
        public double M1 { get; set; } = 1000.0;
        public double M2 { get; set; } = 1000.0;
        public double B { get; set; }
        public double Alpha { get; set; } = 30 * Math.PI / 180;
        public double Fi { get; set; } = 15 * Math.PI / 180;
        public double DAlpha { get; set; } = 0.0;
        public double DFi { get; set; } = 0.0;
        public void RungeKut()
        {
            double k1 = H * Fx1(Alpha, DAlpha, Fi, DFi);
            double l1 = H * Fx2(Fi, DFi, Alpha, DAlpha);
            double y_half = Alpha + H * DAlpha / 2.0 + k1 / 8.0;
            double dy_half = DAlpha + k1 / 2.0;
            double z_half = Fi + H * DFi / 2.0 + l1 / 8.0;
            double dz_half = DFi + l1 / 2.0;
            double k2 = H * Fx1(y_half, dy_half, z_half,
dz_half);
            double l2 = H * Fx2(z_half, dz_half, y_half,
dy_half);
            y_half = Alpha + H * DAlpha / 2.0 + k2 / 8.0;
            dy_half = DAlpha + k2 / 2.0;
            z_half = Fi + H * DFi / 2.0 + l2 / 8.0;
            dz_half = DFi + l2 / 2.0;
            double k3 = H * Fx1(y_half, dy_half, z_half,
dz_half);
            double l3 = H * Fx2(z_half, dz_half, y_half,
dy_half);
            double y_full = Alpha + H * DAlpha + k3 / 2.0;
            double dy_full = DAlpha + k3;
            double z_full = Fi + H * DFi + l3 / 2.0;
            double dz_full = DFi + l3;
```

```
            double k4 = H * Fx1(y_full, dy_full, z_full,
dz_full);
            double l4 = H * Fx2(z_full, dz_full, y_full,
dy_full);
            Alpha = Alpha + H * DAlpha + (k1 + 2 * k2 + 2 * k3 +
k4) * (H / 6.0);
            DAlpha = DAlpha + (k1 + 2 * k2 + 2 * k3 + k4) / 6.0;
            Fi = Fi + H * DFi + (l1 + 2 * l2 + 2 * l3 + l4) * (H
/ 6.0);
            DFi = DFi + (l1 + 2 * l2 + 2 * l3 + l4) / 6.0;
        }
        private double Fx1(double alpha, double dalpha, double
fi, double dfi)
        {
            double m = M1 / 1000.0;
            double M = M2 / 1000.0;
            double delta = alpha - fi;
            double Mtot = 2 * m + M;
            double denom = L1 * (Mtot - M * Math.Cos(2 *
delta));
            double num = -G * Mtot * Math.Sin(alpha)
                         - M * G * Math.Sin(alpha - 2 * fi)
                         - 2 * Math.Sin(delta) * M * (dfi * dfi
* L2 + dalpha * dalpha * L1 * Math.Cos(delta));
            double Qd1 = B * dalpha;
            return (num - Qd1) / denom;
        }
        private double Fx2(double fi, double dfi, double alpha,
double dalpha)
        {
            double m = M1 / 1000.0;
            double M = M2 / 1000.0;
            double delta = alpha - fi;
            double Mtot = 2 * m + M;
            double denom = L2 * (Mtot - M * Math.Cos(2 *
delta));
            double num = 2 * Math.Sin(delta) * (
                dalpha * dalpha * L1 * (m + M)
                + G * (m + M) * Math.Cos(alpha)
                + dfi * dfi * L2 * M * Math.Cos(delta)
            );
            double Qd2 = B * dfi;
            return (num - Qd2) / denom;
        }
        public double ComputeEnergy()
        {
```

```
            double x1 = L1 * Math.Sin(Alpha);
            double y1 = -L1 * Math.Cos(Alpha);
            double x2 = x1 + L2 * Math.Sin(Fi);
            double y2 = y1 - L2 * Math.Cos(Fi);
            double vx1 = L1 * DAlpha * Math.Cos(Alpha);
            double vy1 = L1 * DAlpha * Math.Sin(Alpha);
            double vx2 = vx1 + L2 * DFi * Math.Cos(Fi);
            double vy2 = vy1 + L2 * DFi * Math.Sin(Fi);
            double T1 = 0.5 * M1 * (vx1 * vx1 + vy1 * vy1);
            double T2 = 0.5 * M2 * (vx2 * vx2 + vy2 * vy2);
            double U1 = M1 * G * (y1 + L1);
            double U2 = M2 * G * (y2 + L1 + L2);
            return T1 + T2 + U1 + U2;
        }
    }
}
```

ДОДАТОК В

Код SmallOscillation.cs

```
using System;
namespace Diplom_2024
{
    public class SmallOscillation
    {
        private const double G = 9.81;
        public double H { get; set; } = 0.05;
        public double M1 { get; set; } = 1.0;
        public double M2 { get; set; } = 1.0;
        public double Mu { get; set; } = 1.0;
        public double Rad1 { get; set; } = 100.0;
        public double Rad2 { get; set; } = 100.0;
        public double B { get; set; } = 1.0;
        public double C1 { get; set; } = 1.0;
        public double C2 { get; set; } = 1.0;
        public double H1 { get; set; }
        public double H2 { get; set; }
        public double Omega1 { get; set; }
        public double Omega2 { get; set; }
        public double Alpha { get; set; } = 30 * Math.PI / 180;
        public double Fi { get; set; } = 15 * Math.PI / 180;
        public void Compute()
        {
            double pMu = 1 + Mu;
            double pB = 1 + B;
            Omega1 = Math.Sqrt(G / Rad1) * Math.Sqrt((pMu * pB +
Math.Sqrt(pMu * pMu * pB * pB - 4 * B * pMu)) / 2 / B);
            Omega2 = Math.Sqrt(G / Rad1) * Math.Sqrt((pMu * pB -
Math.Sqrt(pMu * pMu * pB * pB - 4 * B * pMu)) / 2 / B);
            H1 = pMu / Mu / B * (2 * B - pMu * pB -
Math.Sqrt(pMu * (pMu * pB * pB - 4 * B))) /
                 (pMu * pB + Math.Sqrt(pMu * (pMu * pB * pB - 4
* B)));
            H2 = pMu / Mu / B * (2 * B - pMu * pB +
Math.Sqrt(pMu * (pMu * pB * pB - 4 * B))) /
                 (pMu * pB - Math.Sqrt(pMu * (pMu * pB * pB - 4
* B)));
            C1 = (Fi - Alpha * H2) / (H1 - H2);
            C2 = (Fi - Alpha * H1) / (H2 - H1);
        }
        public void UpdateAngles(double time)
```

```
        {
            Alpha = C1 * Math.Cos(Omega1 * time) + C2 *
Math.Cos(Omega2 * time);
            Fi = C1 * H1 * Math.Cos(Omega1 * time) + C2 * H2 *
Math.Cos(Omega2 * time);
        }
    }
}
```