Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Автоматики і комп'ютеризованих технологій_____
(повна назва)

Кафедра __Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки___
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

Розробка програмного забезпечення для комп'ютерного моделювання
_____кіберфізичних систем з використанням колекції компіляторів GNU
(тема)

Виконав:
здобувач ____2_____ року навчання,
групи КІТПВм-24-2_____

_____Назарій ПІВЕНЬ_____
(власне ім'я, прізвище)

Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка_____
(код і повна назва спеціальності)

Тип програми ____освітньо-професійна_____

Освітня програма Комп'ютерно-інтегровані технологічні процеси і виробництва_____
(повна назва освітньої програми)

Керівник _проф. Юрій РОМАШОВ_____
(посада, власне ім'я, прізвище)

Допускається до захисту
Завідувач кафедри КІТАР _____ __Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Я, Півень Назарій Павлович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу з академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«20» грудня 2025р Назарій ПІВЕНЬ

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти другий (магістерський)
Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(код і повна назва)
Тип програми Освітньо-професійна
Освітня програма Комп'ютерно-інтегровані технологічні процеси і виробництва
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачу Півню Назарію Павловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення для комп'ютерного моделювання кіберфізичних систем з використанням колекції компіляторів GNU затверджена наказом університету від «10» листопада 2025 р. №. 1029Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 19 грудня 2025 р.
3. Вихідні дані до роботи
3.1 Модель системи комп'ютерного моделювання кіберфізичних систем з використанням колекції компіляторів GNU
3.2 Реалізація моделі у вигляді програмного засобу
3.3 Система комп'ютерного моделювання кіберфізичних систем
3.4 Система управління колекцією компіляторів GNU
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.2 Аналіз предметної області
4.3 Аналіз технічного завдання
4.4 Розробка концепції системи комп'ютерного моделювання
4.5 Розробка програмного забезпечення системи комп'ютерного моделювання
4.6 Охорона праці
4.7 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

5.1 Графічний демонстраційний матеріал в форматі powerpoint (*.ppt). ФорматуА4 – 10 сторінок

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Актуальність роботи та постановлення задачі	01.09.25-15.09.25	Виконано
2	Визначення мети, об'єкту і предмету розробки	15.09.25-23.09.25	Виконано
3	Аналіз предметної області	24.09.25-29.10.25	Виконано
4	Аналіз технічого завдання	30.09.25-15.10.25	Виконано
5	Аналіз та обґрунтованість вибору технологій розробки	16.10.25-01.11.25	Виконано
6	Розробка програмного забезпечення для комп'ютерного моделювання кіберфізичних систем	01.11.25-15.11.25	Виконано
7	Оформлення пояснювальної записки	16.11.25-30.11.25	Виконано
8	Подання роботи на перевірку StrikePlagiarism	01.12.25-03.12.25	Виконано
9	Подання роботи на рецензію	03.12.25-04.12.25	Виконано
10	Подання роботи на підпис завідуючого кафедри	04.12.25-05.12.25	Виконано
11	Подання кваліфікаційної роботи в екзаменаційну комісію	06.12.25-08.12.25	Виконано

Дата видачі завдання «01» вересня 2025 р.

Здобувач _____ Назарій ПІВЕНЬ
(підпис)

Керівник роботи _____ проф. Юрій РОМАШОВ
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 85 с., 1 табл., 47 рис., 5 додатків, 44 джерела.

КІБЕРФІЗИЧНА СИСТЕМА, КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ, АВТОМАТИЧНА ГЕНЕРАЦІЯ КОДУ.

Метою роботи є автоматизація процесу комп'ютерного моделювання кіберфізичних систем для скорочення часу розробки та аналізу динамічних процесів функціонування за рахунок реалізації гібридної архітектури трансляції математичних виразів у виконуваний програмний код без використання пропрієтарного програмного забезпечення.

Об'єкт дослідження – процес автоматизованої обробки, трансляції, обчислення та візуалізації математичних виразів у багатотехнологічних програмних середовищах.

Предмет дослідження – методи побудови і аналізу абстрактних синтаксичних дерев на основі валідації і трансляції математичних виразів у виконуваний програмний код з використанням інтероперабельної взаємодії.

У роботі розглянуто актуальні питання автоматизації процесу комп'ютерного моделювання, синтаксичного аналізу виразів і організації високопродуктивних обчислень у багатотехнологічних середовищах. Використано методи побудови абстрактних синтаксичних дерев на основі математичних виразів, трансляції та інтероперабельності мов програмування. Обрано технологій для виконання поставленої задачі та виконано практичну реалізацію десктопного програмного застосунку з підтримкою динамічної компіляції GNU.

Практична цінність роботи полягає у створенні програмного засобу для автоматизації процесу комп'ютерного моделювання, що значно спрощує проведення обчислювальних експериментів.

Результати роботи відповідають Цілям сталого розвитку 9: «Промисловість, інновації та інфраструктура» та 4: «Якісна освіта».

ABSTRACT

The thesis contains: 85 p., 1 table, 47 figures, 5 appendices, 44 sources.

CYBER-PHYSICAL SYSTEM, COMPUTER MODELING, AUTOMATIC CODE GENERATION.

The purpose of the work is to develop software to automate the process of computer modeling of cyber-physical systems using open software technologies that provide analysis, translation of mathematical expressions into executable program code, export of calculation results, their specification and graphic interpretation.

The object of the study is the process of automated processing, translation, calculation and visualization of mathematical expressions in multi-technological software environments.

The subject of the study is methods for constructing and analyzing abstract syntax trees based on validation and translation of mathematical expressions into executable program code using interoperable interaction.

The work considers current issues of automating the process of computer modeling, syntactic analysis of expressions and organizing high-performance computing in multi-technological environments. Methods for constructing abstract syntax trees based on mathematical expressions, translation and interoperability of programming languages are used. Technologies were selected to perform the task and a practical implementation of a desktop software application with support for GNU dynamic compilation was performed.

The practical value of the work lies in the creation of a software tool for automating the computer modeling process, which significantly simplifies the conduct of computational experiments.

The results of the work correspond to Sustainable Development Goals 9: “Industry, Innovation and Infrastructure” and 4: “Quality Education”.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ДБН – державні будівельні норми;

ДСанПіН – державні санітарні правила і норми;

ДСТУ – державний стандарт України;

ЖЦ – життєвий цикл;

КФС – кіберфізична система;

ПЗ – програмне забезпечення;

AST – abstract syntax tree (абстрактне синтаксичне дерево);

CAD – computer-aided design (система автоматизованого проєктування);

GCC – GNU compiler collection (колекція компіляторів GNU);

GNU – GNU’s Not Unix (вільна операційна система у контексті роботи набір інструментів та компіляторів);

HTML – HyperText Markup Language (мова розмітки гіпертексту);

IDEF0 – Integration Definition for Function Modeling (методологія функціонального моделювання);

ISO – International Organization for Standardization (міжнародна організація зі стандартизації);

JS – JavaScript (мова програмування);

JSON – JavaScript Object Notation (текстовий формат обміну даними);

MBD – Model-Based Design (модельно-орієнтоване проєктування);

UCH – Universal Chart Data (універсальний формат даних діаграм ,розроблений формат файлів).

ВСТУП

Стрімкий розвиток концепцій Індустрії 4.0 та 5.0, зростання складності технологічних процесів та вимоги до надійності керування зумовлюють широке впровадження кіберфізичних систем (КФС). Передумовою ефективного проєктування таких систем є етап комп'ютерного моделювання, який дозволяє верифікувати алгоритми керування та передбачити поведінку фізичних об'єктів у критичних режимах без ризику пошкодження реального обладнання.

Традиційні підходи до моделювання часто базуються на використанні пропрієтарних програмних комплексів (MATLAB/Simulink і ANSYS), які, попри високу функціональність, мають суттєві недоліки: високу вартість ліцензій, закритість програмного коду ядра та значні вимоги до апаратних ресурсів.

Водночас ручне програмування чисельних методів для моделювання вимагає від інженерів глибоких знань специфічних мов програмування, що підвищує поріг входження та ймовірність допущення помилок.

Існує нагальна потреба у створенні доступних, відкритих та автоматизованих інструментів, які забезпечують пряму трансляцію математичних моделей у високопродуктивний машинний код, поєднуючи зручність високорівневого опису з ефективністю низькорівневих обчислень.

Мета роботи – підвищити ефективність та доступність процесу проєктування кіберфізичних систем шляхом розробки програмного забезпечення для автоматизації комп'ютерного моделювання з використанням відкритих програмних технологій, що забезпечують аналіз, трансляцію математичних виразів у виконуваний програмний код та графічну інтерпретацію результатів.

Для досягнення мети передбачається розв'язати такі завдання:

– провести аналіз предметної області та існуючих інструментів моделювання КФС, визначити їхні переваги та недоліки;

– обґрунтувати вибір технологій реалізації гібридної архітектури системи;

– розробити методи побудови та аналізу абстрактних синтаксичних дерев для валідації та конвертації LaTeX-виразів у програмний код;

– реалізувати алгоритми автоматичної генерації коду, його компіляції засобами GNU GFortran та організації обчислювального процесу;

– створити програмний модуль для візуалізації результатів моделювання та експорту даних у зручні формати; – провести тестування розробленого програмного забезпечення на прикладі типових задач динаміки та оцінити точність і швидкодію обчислень;

– оформити кваліфікаційну роботу згідно з методичними вказівками з підготовки й оформлення кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп’ютерно-інтегровані технології та робототехніка освітньої програми «Комп’ютерно-інтегровані технологічні процеси і виробництва» [1] , положення про організацію освітнього процесу у ХНУРЕ [2] та ДСТУ 3008:2015 [3] .

Об’єкт дослідження – процес автоматизованої обробки, трансляції, обчислення та візуалізації математичних виразів у багатотехнологічних програмних середовищах.

Предмет дослідження – методи побудови і аналізу абстрактних синтаксичних дерев на основі валідації і трансляції математичних виразів у виконуваний програмний код з використанням інтероперабельної взаємодії.

Наукова новизна та практичне значення роботи полягає у розробці унікального підходу до автоматизації моделювання, який поєднує зручність вводу математичних моделей у форматі LaTeX з максимальною продуктивністю компільованого коду Fortran завдяки використанню гібридної архітектури.

Практична цінність роботи полягає у створенні десктопного програмного засобу, що значно спрощує проведення обчислювальних експериментів.

Розроблена система дозволяє дослідникам та інженерам зосередитись на параметрах моделі та аналізі результатів, делегуючи рутинні етапи програмному засобу, що забезпечує високу точність і швидкість розрахунків без необхідності поглиблених знань програмування у користувача, що сприяє оптимізації процесів розробки КФС.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Принцип роботи системи моделювання

Кіберфізична система (КФС) – система, в якій комп’ютерні алгоритми тісно інтегровані з фізичними процесами, що дозволяє програмному забезпеченню (ПЗ) контролювати реальні фізичні об’єкти та планувати подальші дії шляхом аналізу надісланих даних у програму [4].

Моделювання – процес створення та дослідження гібридних математичних моделей, що інтегрують безперервну динаміку фізичних процесів з подієво-керованою логікою обчилювальних алгоритмів керування [5].

Суть полягає в взаємодії між інформаційною та матеріальною складовими у єдиному віртуальному середовищі, що дозволяє проводити верифікацію проєктних рішень, прогнозувати поведінку системи у критичних режимах та оптимізувати параметри функціонування до етапу створення фізичного прототипу системи і отримати спосіб створення віртуального представлення реальної системи.

Головна ідея полягає в зворотному зв’язку компонентів системи один з одним, завдяки чому постійно здійснюють обмін даними та миттєво адаптують свою поведінку.

Для моделювання таких систем використовуються інструменти, здатні інтегрувати моделювання фізичних процесів з моделюванням кіберчастин.

Три найвідоміших програмних забезпечення для моделювання кіберфізичних систем є:

– MATLAB/Simulink [6;7];

– Modelica/OpenModelica [8;9];

– ANSYS [10].

Дані інструменти дозволяють створювати віртуальні копії реальних систем, тестувати їхні алгоритми керування та оптимізувати продуктивність перед розгортанням у фізичному світі.

Основний принцип роботи програмного забезпечення для моделювання КФС полягає в таких етапах життєвого циклу (ЖЦ):

– моделювання фізичної системи;
– моделювання кіберсистеми;
– верифікація та валідація в циклах;
– розгортання та експлуатація.

Для моделювання фізичної системи створюється фізична модель для опису динаміки реального об’єкта за допомогою диференціальних рівнянь.

Для моделювання кіберсистеми створюється модель керуючого алгоритму.

Отже, на цьому етапі обидві моделі працюють у єдиному програмному середовищі, обмінюючись даними на кожному кроці симуляції.

Далі здійснюється заміна моделі керуючого алгоритму на реальний керуючий код оскільки фізична модель все ще виконується в симуляційному програмному забезпеченні та взаємодіє з компільованим кодом контролера. Це дозволяє перевірити правильність роботи написаного ПЗ у віртуальному фізичному середовищі.

Далі відбувається підключення мікроконтролера до ПЗ симулятора, що слугуватиме справжньою апаратною частиною для керування системою.

Симулятор перетворюється на емулятор фізичного світу, адже він надсилає дані з віртуальних датчиків на входи реального контролера.

Контролер в свою чергу, обробивши дані, повертає команди керування назад у симулятор. Це дозволяє перевірити не лише код, а й продуктивність апаратного забезпечення контролера в умовах максимально наближених до реальних.

Саме для розгортання, експлуатації та підтримки кінцевого продукту використовується моделювання.

З готової та верифікованої моделі керування автоматично генерується оптимізований код для кінцевого вбудованого пристрою.

Модель симуляції у сучасних кіберфізичних системах використовується як цифровий двійник – віртуальна копія реального об’єкта, що працює паралельно з ним.

Віртуальна копія отримує реальні дані з фізичного об’єкта, аналізує його стан, допомагає оптимізувати роботу, виявляє аномалії та планує обслуговування, що забезпечує постійний зворотній зв’язок між комп’ютерною та фізичною складовою системи протягом усього життєвого циклу кінцевого продукту.

На рисунку 1.1 зображено основний принцип роботи кіберфізичної системи, підкреслюючи взаємодію між кібернетичним та фізичном світом.

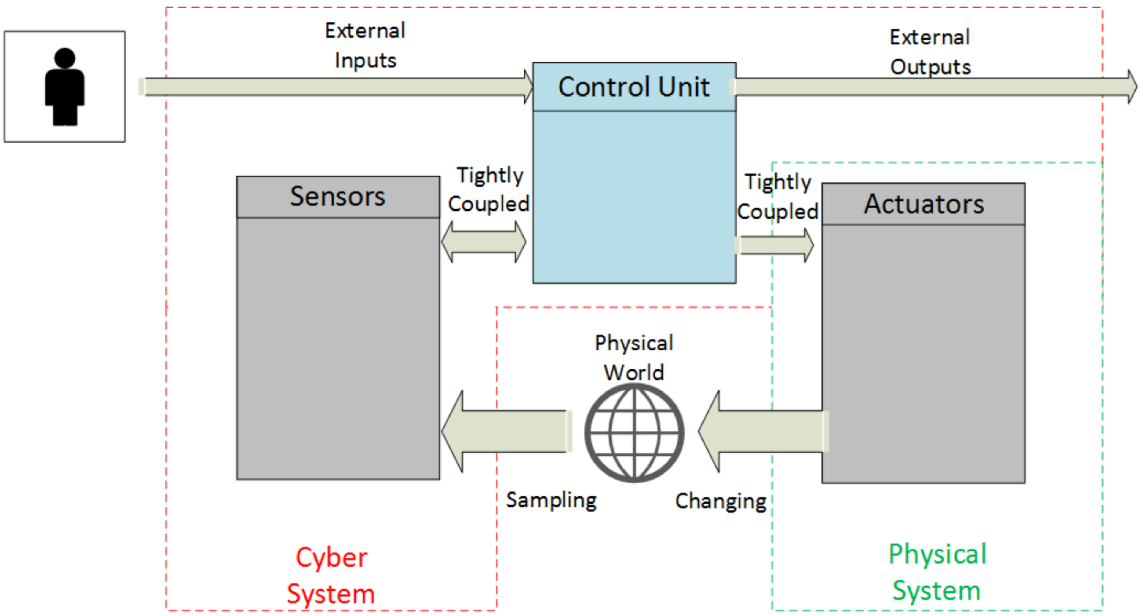

Рисунок 1.1 – Основний принцип роботи кіберфізичних систем [11]

Схема, зображена на рисунку 2.1, демонструє ключовий цикл зворотного зв’язку, який є основою КФС.

Фізичний світ (англ. Physical world) передає інформацію про параметри зовнішнього середовища (англ. Sampling) в датчики (англ. Sensors), відбувається перехід від фізичного світу до кібернетичного.

Датчики обмінюються тісно пов’язаними (англ. Tightly coupled) даними з контрольним пунктом (англ. Control unit), який обробляє і передає дані в виконавчі механізми (англ. Actuators).

Виконавчі механізми передають дані в фізичний світ і здійснюють фізичну дію (англ. Changing), впливаючи на навколишнє середовище, завершують цикл.

Зона, обведена червоною пунктирною лінією (англ. Cyber System), та зона, обведена зеленою пунктирною лінією (англ. Physical System), чітко розмежовують кібернетичні та фізичні компоненти, що є типовим підходом у теорії КФС.

Отже, функціональні вимоги до програмного забезпечення для комп'ютерного моделювання кіберфізичних систем визначають які функції та операції повинні бути доступні користувачу та системі в цілому.

Основні функціональні вимоги:

– моделювання фізичного домену;
– моделювання кібернетичного домену;
– візуалізація та аналіз.

Моделювання фізичного домену функціонально означає можливість інтегрувати та симулювати моделі, що описують фізичні процеси, часто використовуючи мови на основі рівнянь з здатністю виконувати моделювання з жорсткими часовими обмеженнями, необхідними для верифікації керуючих алгоритмів та можливістю об'єднання та синхронізації моделей, розроблених у різних доменах і різних інструментах.

Моделювання кібернетичного домену дозволяє генерувати програмний код безпосередньо з моделі керування для подальшого розгортання на мікроконтролерах.

Це забезпечує можливість заміни симульованого контролера на реальний код, скомпільований у GCC або іншою технологією, і обмін даними в реальному часі з фізичним мікроконтролером [12].

Візуалізація функціонально означає можливість надання інтерфейсу для створення, редагування та відображення результатів симуляції.

Варто зауважити, що інтеграція віртуальної копії та фізичного об'єкта перетворює КФС з простого механізму автоматизації на інтелектуальну екосистему, здатну до самоадаптації. Використання концепції цифрового двійника дозволяє не лише реагувати на поточні зміни, а й прогнозувати майбутній стан системи за допомогою методв машинного навчання. Це мінімізує ризики критичних збоїв та значно скорочує витрати на етапі прототипування.

В свою чергу нефункціональні вимоги визначають, як саме має працювати програмне забезпечення.

Основні нефункціональні вимоги:

– продуктивність;
– надійність;
– масштабованість;
– безпека;
– зручність використання;
– інтероперабельність.

Продуктивність означає здатність виконувати симуляцію швидше, ніж відбувається фізичний процес, або синхронізуватись із зовнішнім обладнанням. Це необхідно для мінімізації часу, необхідного для проведення обчислень.

Під надійністю мається на увазі коректна обробка числових помилок і збоїв під час інтеграції різних типів моделей.

Відтворюваність являє собою гарантію того, що симуляція з тими самими вхідними даними завжди даватиме ідентичний результат.

Масштабованість являє собою здатність системи ефективно змінюватися без втрат продуктивності, надійності та узгодженості між компонентами

Безпека з точки зору нефункціональних вимог являє собою комплекс заходів, спрямованих на захист фізичної та кібернетичної складових з метою запобігання несанкціонованому доступу чи будь-якому впливу, що створює загрозу для системи.

Інтуїтивно зрозумілий графічний інтерфейс, документація та підтримка програмного забезпечення на всіх його життєвих циклах забезпечують зручність використання.

Інтероперабільність забезпечує здатність компонентів кіберфізичної системи ефективно взаємодіяти між собою та із зовнішніми системами забезпечуючи сумісність та безпечний обмін даними [13].

1.2 Аналіз структури програмного забезпечення моделювання кіберфізичної системи

Системи комп'ютерного моделювання є критично важливими інструментами для вирішення задач, де інші експерименти є коштовними, небезпечними, тривалими і часто неможливими.

Розглянемо структуру типового програмного забезпечення для моделювання кіберфізичних систем на прикладі програмного середовища для багатодоменного моделювання MATLAB від MathWorks [14].

MATLAB – це високопродуктивна мова програмування для технічної інтеграції обчислень, візуалізації та програмування в однойменному середовищі, де всі проблеми і рішення виражено в математичному вигляді.

Звичайний сценарій використання включає в себе:

– розробку програмного забезпечення;
– робота з інженерною графікою;
– аналіз інформації, дослідження та візуалізація;
– розробка алгоритмів;
– робота з обчисленнями.

MATLAB призначено для одночасного моделювання безперервних у часі фізичних процесів та моделювання дискретних у часі обчислювальних алгоритмів. Він поєднує в собі інструменти для побудови фізичних і кібернетичних моделей, верифікації та автоматизованого розгортання систем керування.

Отже, структура такого програмного забезпечення може бути розділена на такі ключові компоненти:

– модуль симуляції;
– графічне середовище моделювання;
– бібліотеки компонентів;
– підсистема візуалізації та аналізу;
– інтерфейси інтеграції та розгортання.

Модуль симуляції являє собою програмне забезпечення реалізації гібридного принципу обчислень, бо забезпечує одночасне моделювання безперервних у часі фізичних подій та дискретних подій у часі у кіберчастині.

Графічне середовище моделювання надає інструменти для візуальної збірки моделі. являє собою інтерфейс, з яким працює користувач.

Бібліотеки компонентів являють собою набір елементів для багатодоменного моделювання для реалізації функціональних вимог.

Підсистема візуалізації та аналізу відповідає за обробку і відображення параметрів та результатів моделювання симуляції.

Інтерфейси інтеграції та розгортання є набором інструментів для забезпечення взаємодії ПЗ з реальним обладнанням та зовнішніми системами.

На рисунку 1.2 наведено схему моделювання кіберфізичної системи автоматичного керування літаком F-14 [15].

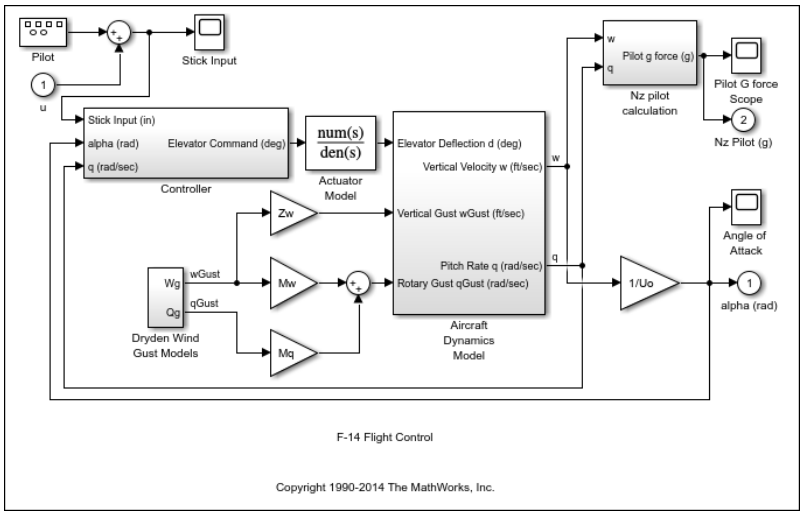

Рисунок 1.2 – Кіберфізична система автопілот для літака F-14 [9]

Розглянемо призначення ключових блоків моделі кіберфізичної системи автопілоту літака, що зображена на рисунку 1.2:

– пілот (англ. Pilot) – блок-джерело, який генерує вхідний бажаний сигнал (команду) від пілота, наприклад, бажаний кут атаки або перевантаження;

– блок суми (англ. Sum) – обчислює сигнал помилки, віднімаючи фактичний виміряний стан сигнал alpha зі зворотного зв’язку від бажаної команди сигнал від пілота;

– контролер (англ. Controller) – це кіберчастина системи, реалізована як підсистема, що приймає сигнал помилки та обчислює необхідну зважувальну команду (англ. Elevator command) для виконавчого механізму;

– модель приводу (англ. Actuator Model) – це підсистема, що імітує фізичний виконавчий механізм (наприклад, гідравліку), який не може миттєво виконати команду, а має власну динаміку (затримку, швидкість реакції);

– блок num(s)/den(s) – це блок передавальна функція, який математично описує динаміку лінійної системи у просторі Лапласа;

– моделі поривів вітру Драйдена (англ. Dryden Wind Gust Models) – це підсистема, що генерує зовнішні збурення (сигнали wGust та qGust), які імітують реалістичну турбулентність та пориви вітру, що впливають на літак;

– модель динаміки літака (англ. Aircraft Dynamics Model) – це основна фізична модель, яка містить диференціальні рівняння руху літака та обчислює його реакцію на дії рулів та вітру;

– блок підсилення 1/Uo – виконує математичне перетворення;

– лінія від alpha до суматора «+-» – це лінія зворотного зв’язку, яка повертає фактичний стан системи на вхід, дозволяючи контролеру коригувати свої дії;

– обчислювальний блок розрахунку перевантаження Nz пілота (англ. Nz pilot calculation) – це обчислювальний блок, який розраховує вертикальне перевантаження (англ. G-force), що діє на пілота, на основі параметрів руху літака (w та q);

– осцилографи Pilot G force Scope та кут атаки (англ. Angle of Attack) – це блоки візуалізації, які показують графіки сигналів, дозволяючи проаналізувати поведінку системи в часі.

Отже, в цій моделі реалізовано кіберфізичне моделювання, де кібер-система через контролер безперервно обчислює керуючі команди, базуючись на даних зворотного зв’язку, що надходять від фізичної системи, представленої модулями «Aircraft Dynamics Model» та «Actuator Model», яка, у свою чергу, також зазнає впливу зовнішніх збурень.

1.3 Модельно-орієнтоване програмування та методи автоматичного генерування програмного коду

Сучасний підхід до розробки програмного забезпечення являє собою перехід від класичного написання програмного кода до декларативного опису моделей. В рамках цього підходу відбувається фокус не на реалізації алгоритмічних структур, а на формалізації певної предметної області. Такий рівень абстракції дозволяє прибрати розрив семантики між математичними моделями та програмним кодом.

Все більш актуальними стають технології, що здатні автоматично перетворювати специфікації високого рівня безпосередньо в програмний код. Використання математичних виразів, в тому числі LaTeX-розмітки [16] , в якості вхідної мови програмування фактично перетворює систему в предметно-орієнтовану мову, що дозволяє такій системі самостійно визначати методи численних рішень, порядок виконання операцій та керувати ресурсами, звільняючи користувача від рутинних задач.

Заміна ручного написання програмного коду автоматичною генерацією коду на основі абстрактних синтаксичних дерев збільшує надійність та верифікованість в кіберфізичних системах. Автоматизований транслятор виключає клас помилок, пов’язаних з людським фактором під час переносу формул в програмний код. Такий підхід забезпечує гнучке моделювання, що проявляється в зміні логіки поведінки системи завдяки редагуванню одного рядку формули без необхідності рекомпіляції всього програмного забезпечення.

Ітерації між моделюванням та симуляцією можуть покращити якість проєктування системи на ранній стадії, зменшуючи кількість помилок, виявлених пізніше в процесі проєктування.

У модельно-орієнтованому проєктуванні модель системи знаходиться в центрі робочого процесу. Модельно-орієнтоване проєктування дозволяє швидко та економічно ефективно розробляти динамічні системи, включаючи системи керування, системи обробки сигналів та системи зв’язку.

На рисунку 1.3 зображено загальну схему V-Model.

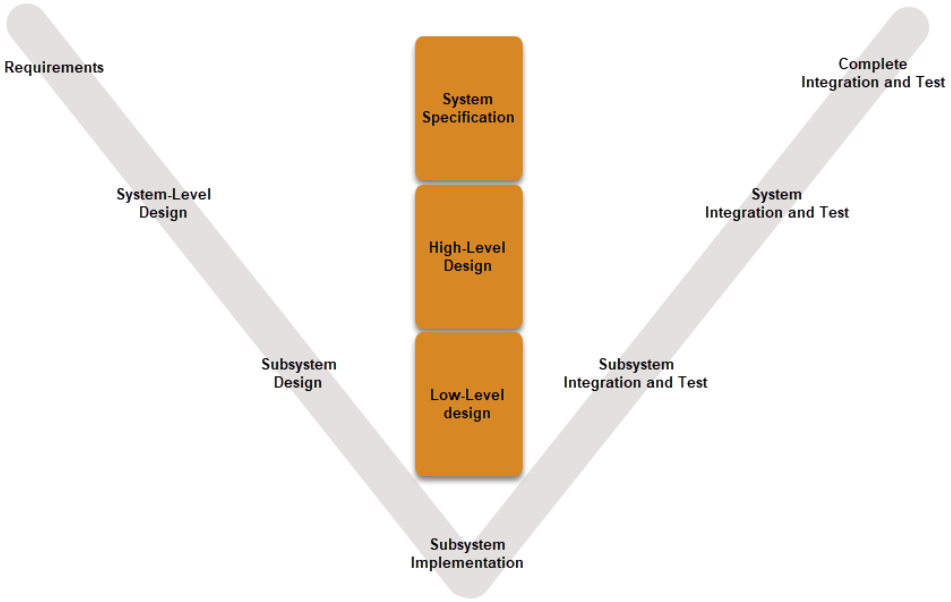

Рисунок 1.3 – Загальна схема V-Model [17]

Наведена схема ілюструє фундаментальні задачі в рамках методології Model-Based Design(MBD). Процес складається з наступних етапів:

– визначення системи та компонування;
– моделювання та валідація системи;
– проєктування системи.

Визначення системи являє собою ідентифікування цілей моделювання, визначення необхідних компонентів та розробку загальної структури системи.

Моделювання та валідація системи включають в себе: створенн моделей компонентів, їх тестування, інтеграцію в єдину систему після комплексної перевірки.

Проєктування системи являє собою розробку та тестування нових компонентів у середовищі моделювання.

Початкові етапи моделі дають можливість моделювання існуючої системи та створення контексту для проєктування нових елементів. Наступним кроком є реалізація розроблених компонентів, де можуть використовуватись методи швидкого прототипування та автоматичної генерації коду для нітеграції з фізичним обладнанням.

На рисунку 1.4 зображено приклад робочого процесу проєктування на основі моделі Simulink.

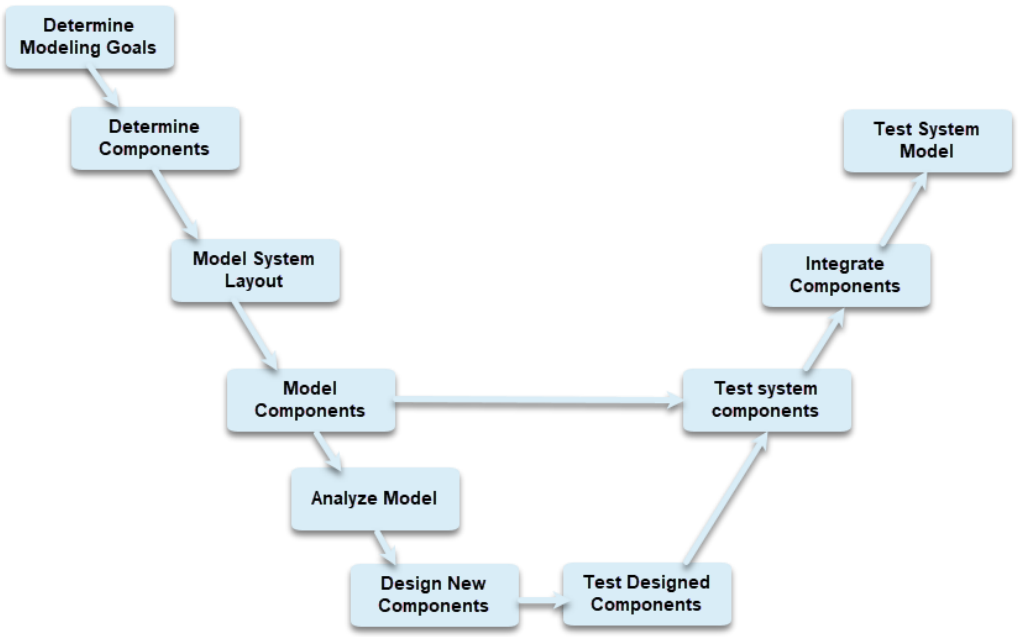

Рисунок 1.4 – Робочий процес проєктування на основі моделі Simulink [17]

Як видно з рисунку, замість того, щоб покладатися на фізичні прототипи та текстові специфікації, – модельно-орієнтоване проєктування використовує системну модель як виконувану специфікацію протягом усієї розробки. Воно підтримує проєктування та моделювання на рівні системи та компонентів, автоматичну генерацію коду, а також безперервне тестування та верифікацію.

Отже, модельно-орієнтоване проєктування дозволяє :

– використовувати спільне середовище проєктування для всіх проєктних команд;

– безпосередньо пов’язувати проєкти з вимогами;

– постійно виявляти та виправляти помилки шляхом інтеграції тестування з проєктуванням;

– удосконалювати алгоритми за допомогою багатодоменного моделювання;

– автоматично генерувати вбудований програмний код та документацію;

– розробляти та повторно використовувати набори тестів.

1.4 Висновки до розділу 1

Кіберфізичні системи характеризуються тісною інтеграцією обчислювальних алгоритмів із фізичними процесами.

Важливим етапом життєвого циклу КФС є комп’ютерне моделювання, яке дозволяє проводити верифікацію проєктних рішень без ризику пошкодження реального обладнання.

На основі аналізу існуючих програмних рішень було виділено ключові структурні компоненти, необхідні для побудови ефективної системи моделювання: модуль синтаксичного розбору математичних моделей, обчислювальне ядро для розв’язання диференціальних рівнянь та підсистема візуалізації результатів.

Обґрунтовано доцільність застосування підходів модельно-орієнтованого проєктування та показано, що сучасні тенденції розробки КФС передбачають перехід від ручного кодування до автоматичної генерації програмного коду на основі формальних моделей або математичних специфікацій.

Аналіз методів автоматичної генерації коду підтвердив актуальність створення програмного комплексу, що використовує абстрактні синтаксичні дерева для аналізу введених користувачем виразів та подальшої трансляції їх у високоефективний машинний код, що дозволить поєднати зручність високорівневого опису моделі з продуктивністю низькорівневих обчислень.

В умовах стрімкого розвитку кіберфізичних систем перехід до методології модельно-орієнтованого проєктування є критично важливою необхідністю сучасної інженерії.

Відмова від написання програмного коду на користь автоматичного генерування коду математичними моделями дозволяє нівелювати вплив людського фактору, забезпечує верифікованість алгоритмів та суттєво скорочує час виведення продукту на ринок, що є ключовим фактором підвищення ефективності, надійності та безпеки новітніх технологічних комплексів.

2 РОЗГЛЯД СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯ КІБЕРФІЗИЧНИХ СИСТЕМ

2.1 Аналіз існуючих аналогів

Моделювання КФС є комплексним завданням інтеграції фізичних процесів, що описуються безперервними диференціальними рівняннями, та кібернетичних компонентів з дискретною логікою керування.

Існуючі комерційні і відкриті програмні комплекси, такі як MATLAB/Simulink, OpenModelica та ANSYS, дозволяють створювати складні багатодоменні моделі, що дає змогу верифікувати поведінку системи до її фізичної реалізації.

ПЗ цього класу надають графічні інтерфейси для проєктування, потужні можливості інженерних обчислень та інструменти аналізу. Таким чином вони дозволяють значно спростити процес розробки, перевірки алгоритмів керування та оптимізації параметрів системи, що зменшує вартість та час виходу продукту на ринок.

Водночас такі рішення часто є пропрієтарними, мають високу вартість, можуть вимагати значних обчислювальних ресурсів і не завжди надають достатньої гнучкості для глибокої оптимізації коду ядра симуляції.

На сьогоднішній день відомим є наступне ПЗ для моделювання КФС:

– MATLAB/Simulink – ПЗ від MathWorks є галузевим стандартом у сфері моделювання, аналізу та проєктування складних динамічних систем. Simulink забезпечує візуальне середовище моделювання, підтримує численні бібліотеки компонентів та інтеграцію з апаратними засобами для реального часу;

– OpenModelica – відкрите ПЗ, що реалізує мову для моделювання багатофізичних систем. Надає середовище для символьного та чисельного моделювання, автоматичної генерації коду і сумісності з іншими інструментами моделювання Розробляється та підтримується Open Source Modelica Consortium;

– ANSYS – комерційне ПЗ від компанії ANSYS Inc., який застосовується для інженерного аналізу методами кінцевих елементів, використовується для моделювання структурної, термічної, електромагнітної та гідродинамічної поведінки систем, а також підтримує цифрові двійники та інтеграцію з CAD-системами [18].

У таблиці 2.1 наведено переваги та недоліки відомого ПЗ для моделювання.

Таблиця 2.1 – Переваги та недоліки відомого ПЗ моделювання

Система	Переваги	Недоліки	Особливості
MATLAB/ Simulink	Стандарт галузі	Висока вартість ліцензії; Лімітований контроль над ядром симуляції	Стандарт в аерокосмічній та автомобільній сфері
	Величезні бібліотеки компонентів	Пропрієтарність	
	Інтуїтивний графічний інтерфейс	Лімітований контроль над ядром симуляції	
	Генерація коду для моделювання КФС.		
OpenModelica	Відкритий код	Вищий поріг входження;	Підходить для академічних і наукових проєктів
	Безкоштовність	Менш стабільний інтерфейс користувача	
	Моделювання через Modelica		
ANSYS	Висока точність моделювання	Складність	Орієнтований на великі корпорації
	Інтеграція з цифровими двійниками	Висока вартість	Високоточне фізичне моделювання
	Сумісність з продуктами ANSYS		

Аналіз існуючих рішень показує, що хоча комерційні продукти є потужними, вони мають суттєві обмеження для розробки легковагових, вбудованих або специфічних симуляторів:закритий вихідний код, неможливість глибокої оптимізації генерованого коду та залежність від дорогих ліцензій [19].

2.2 Концепція програми для вирішення поставлених вимог

Враховуючи те, що метою кваліфікаційної роботи є розробка програмного забезпечення для комп’ютерного моделювання кіберфізичних систем з використанням колекції компіляторів GNU – для досягнення поставленої мети необхідно вирішити наступні завдання:

– розробити структурну схему програмного комплексу, що описує цикл обробки даних від синтаксичного аналізу LaTeX-виразів та побудові абстрактних синтаксичних дерев [20] та автоматичної генерації коду до виконання розрахунків обчислювальним ядром і візуалізації результатів;

– забезпечити зрозуміле введення формул для обчислень як у Matlab і Ansys [21] ;

– обґрунтувати вибір технологій та дослідити результати роботи обчислювального ядра з використанням колекції компіляторів GNU для забезпечення високої продуктивності розрахунків математичних моделей і можливості кросплатформеної компіляції;

– визначити та реалізувати формат файлів для обміну даними та збереження результатів симуляції, що генеруються обчислювальним ядром;

– розробити програмний модуль візуалізації для автоматизації задачі перегляду результатів розрахунків, представлених у вигляді набору файлів, отриманих від обчислювального ядра;

– провести комп’ютерні експерименти для верифікації коректності роботи розробленого програмного забезпечення шляхом перевірки візуалізації моделювання тестових кіберфізичних систем та аналізу отриманих результатів.

На відміну від вищезгаданих платформ, система, що розробляється, пропонує гібридну архітектуру, яка поєднує зручність сучасних інтерфейсів та продуктивність низькорівневих мов програмування.

Для реалізації обрано модульний підхід, що складається з взаємопов’язаних підсистем: підсистема керування та трансляції, обчислювальне ядро та модуль візуалізації і постаналізу.

Виведену структурну схему взаємодії компонентів та потоків даних програмного комплексу зображено на рисунку 2.1.

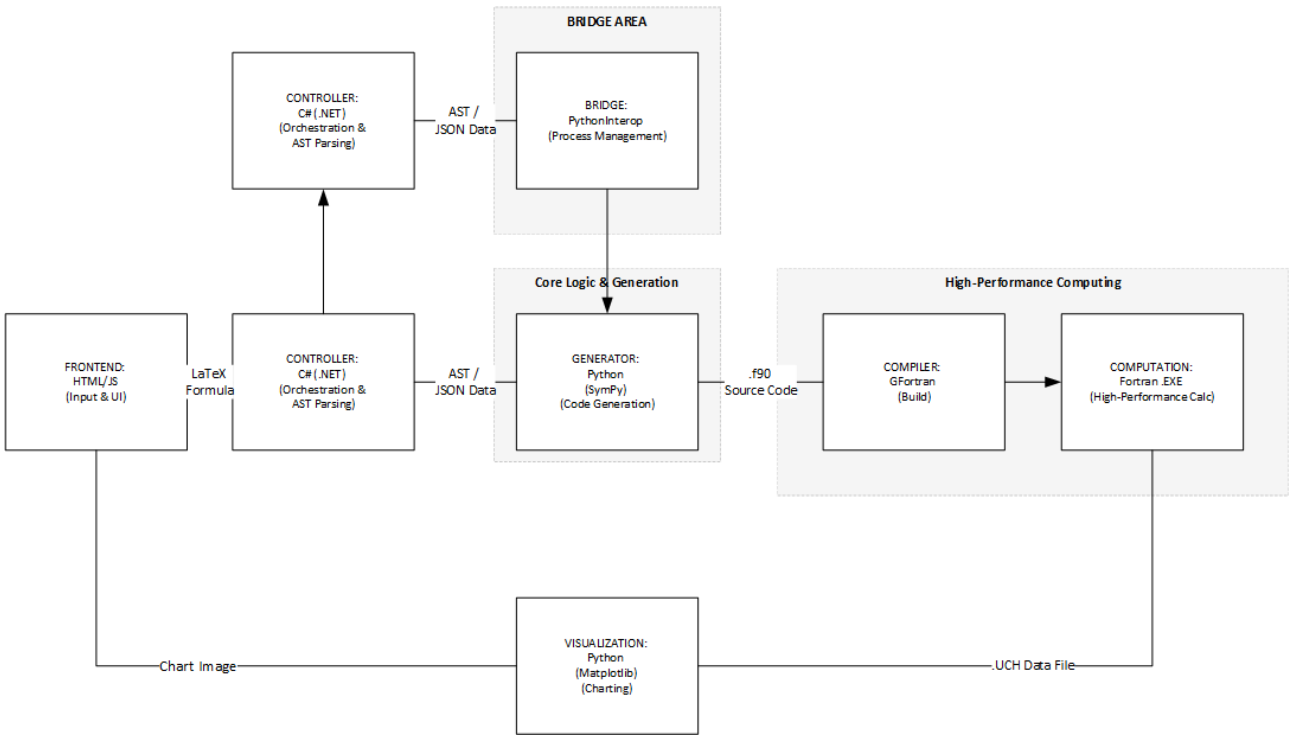

Рисунок 2.1 – Структурна схема взаємодії компонентів та потоків даних програмного комплексу

Підсистема керування і трансляції, написана мовою C# [22], є точкою входу для користувача, адже вона забезпечує відображення графічного інтерфейсу для введення LaTeX-виразів в компоненті WebView [23] у вікні програми. Така підсистема виконує перетворення введених LaTeX-виразів в абстрактне синтаксичне дерево шляхом аналізу вхідних даних JSON [24], отриманих з вікна MathField [25], транслює оброблені вузли дерева спочатку у Python [26], а далі відбувається перетворення виразу для обчислення в вихідний код обчислення GFortran.

Обчислювальне ядро використовує згенерований програмний код, компілює його та виконує як окремий процес. Вибір ланцюга Fortran та колекції компіляторів GNU є фундаментальним, адже дозволяє:

– досягти максимальної продуктивності обчислень завдяки агресивній оптимізації циклів [27] та роботі з пам’яттю, що є критичним для розв’язання диференціальних рівнянь;

– використовувати відкриті стандарти без ліцензійних обмежень;

– стає можливим забезпечення повної кросплатформеності на рівні вихідного коду.

Модуль візуалізації та постаналізу, що написаний мовою програмування Python, являє собою результати моделювання, збережені ядром обчислень у файли даних. Файли даних обробляються скриптами, бо бібіліотеки дозволяють будувати складні графіки та проводити символьні обчислення значно простіше ніж компільовані мови програмування.

Стає можливим зміна логіки аналізу без перекомпілювання основного обчислювального ядра.

Отже, така архітектура дозволяє чітко розподілити відповідальність. C# керує процесом, Fortran виконує важку математичну роботу, Python візуалізує результат.

2.3 розробка контекстної діаграми системи та її декомопозиція

Було розроблено контекстну IDEF0 [28] діаграму A-0, де основний процес – виконувати комп’ютерне моделювання кіберфізичної системи, що зображено на рисунку 2.2.

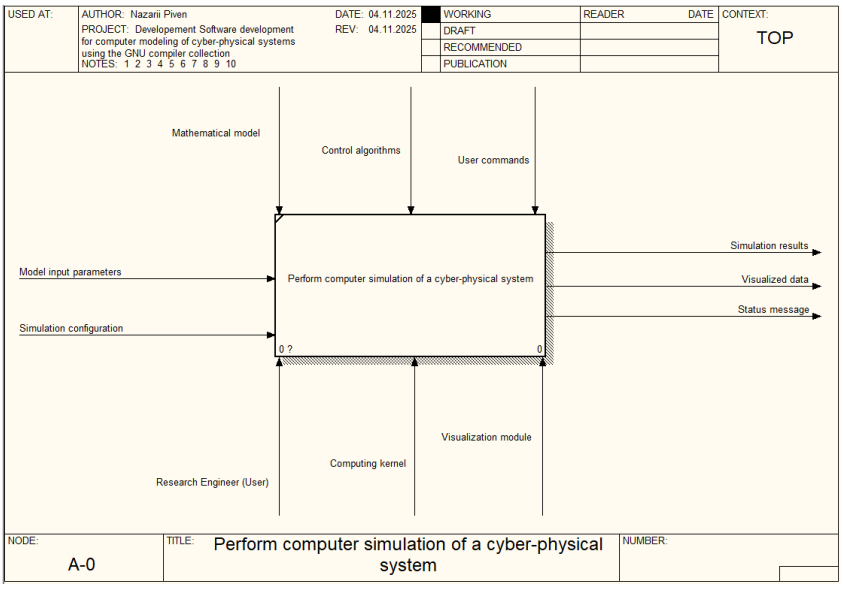

Рисунок 2.2 – Контекстна діаграма ПЗ комп’ютерного моделювання кіберфізичних систем

Декомпозицію основного процесу зображено на рисунку 2.3.

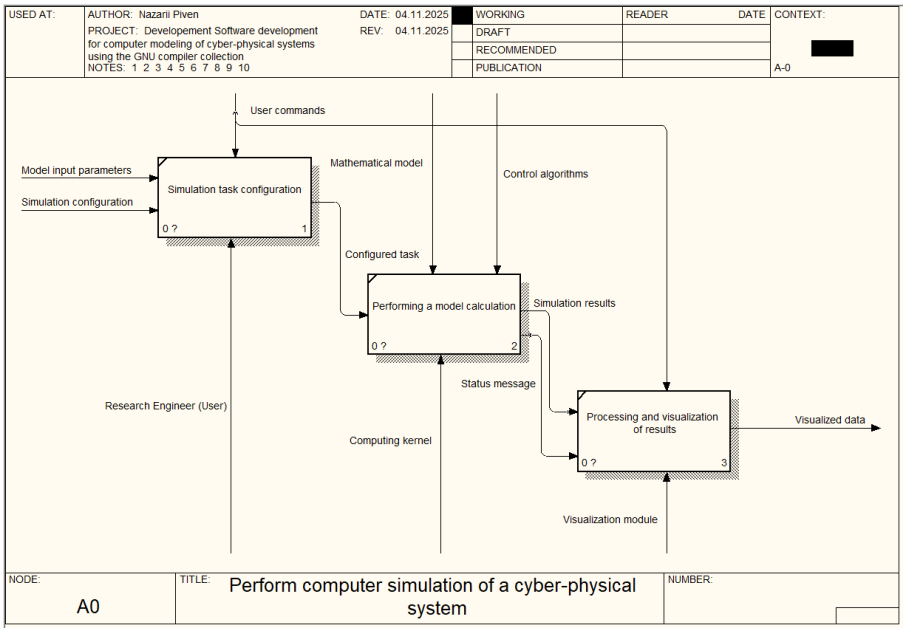

Рисунок 2.3 – Декомпозована модель

2.4 Опис структур даних

У розробленій системі реалізовано гібридний підхід обробки даних, що обумовлено використанням різноманіття технологій. Ключовими елементами архітектури даних є внутрішнє представлення математичних виразів та спеціалізований формат файлового обміну результатами розрахунків.

Для забезпечення коректної трансляції математичних формул між інтерфейсом користувача та обчислювальним ядром використовується модель побудови абстрактних синтаксичних дерев, де формула зберігається як ієрархічна структура об’єктів, що відображає логічні зв’язки між операторами та операндами.

Процес перетворення проходить наступні етапи.

1. Фронтенд. Введений користувачем вираз парситься бібліотекою Compute Engine [29] та серіалізується в формат JSON. Структура JSON-об’єкта містить типізовані вузли.

Приклад JSON формату:

```
{
 «type»: «Expression»,
 «root»: {
  «kind»: «chain»,
  «items»: [
   { «kind»: «symbol», «name»: «x» },
   { «kind»: «op», «name»: «+» },
   { «kind»: «number», «value»: «1» }
  ] }
}
```

2. Бекенд (C#). Отриманий JSON серіалізується у відповідні класи C#, що утворюють деревоподібну структуру в пам’яті програми. Базовим класом є абстрактний клас вузлу виразу ExprNode від якого наслідуються конкретні реалізації:

– ChainNode: для лінійних виразів та арифметичних ланцюгів;

– BigOpNode: для складних операторів з межами (інтеграл, сума, добуток);

– AtomNode: для змінних та констант.

Така структура дозволяє програмно обходити дерево виразу для валідації, модифікації параметрів і генерації коду для цільових обчислювальних ядер.

Взаємодія між обчислювальним модулем та підсистемою візуалізації реалізована через файловий обмін. Використання оперативної пам’яті або сокетів для передачі великих масивів даних від Fortran є недоцільним через складність реалізації інтерфейсів.

Замість стандартних форматів, розроблено власний текстовий формат файлів з розширенням .UCH (Universal Chart Data), який поєднує метадані та числові результати.

Структура файлу .UCH складається з двох секцій.

1 Блок метаданих, що містить налаштування візуалізації та параметри розрахунку у форматі «Ключ Значення»:

– chart, xlabel, ylabel – підписи графіків та осей;

– latex – вихідна формула для відображення;

– iterations, steps – параметри циклу обчислень;

– curves – кількість кривих на графіку;

– curve, color, line, markers – стилістичні параметри для кожної серії даних.

2. Блок даних, що починається ключовим словом data із зазначенянм кількості записів. Наступні рядки містять розраховані значення змінних у форматі наукової нотації, розділені крапкою з комою.

Цей формат дозволяє візуалізатору однозначно інтерпретувати дані без додаткових запитів до бази даних або основного додатка, забезпечуючи слабку зв’язність модулів системи.

Приклад фрагмента файлу .UCH:

```
chart My Chart
latex x+x+x
xlabel Time
ylabel Value
xgrid auto
ygrid auto
legend 0;1
curves 000000002
iterations 1000
steps 0.01
curve X
min -10
max 10
color 242207121
line 11.0
markers 016
data 000001001
 -1.000000000000000E+001; -1.000000000000000E+001
 -9.990000000000000E+000; -9.990000000000000E+000
```

Крім того, генерується супровідний файл .OUT, який містить розширений звіт про параметри моделювання у зручному для читання людиною вигляді.

Приклад фрагмента файлу .OUT:

```
NewProject1
INPUT DATA GENERALISED REPRESENTATION
.....
  COUNT OF THE VARIABLES = 04
       VALUES'S LABEL = My Chart
      VALUES'S CAPTION = CALCULATION FOR EQUATION x+x+x
..... ..... .....
VARIABLES' NAMES                VARIABLES' MEANNINGS       VARIABLES' VALUES
..... ..... .....
         t0      INITIAL VALUE (Start)                      -10.0
         tf      FINAL VALUE (End)                          10.0
         dt      INTEGRATING STEP                            0.01
          X      INDEPENDENT VARIABLE                       NewProject1.UCH
..... ..... .....
.....
OUTPUT DATA GENERALISED REPRESENTATION
.....
  COUNT OF THE VARIABLES = 01
       VALUES'S LABEL = My Chart
      VALUES'S CAPTION = RESULT OF x+x+x
..... ..... .....
VARIABLES' NAMES                VARIABLES' MEANINGS       VARIABLES' VALUES
..... ..... .....
       RESULT      CALCULATED FUNCTION VALUE                 NewProject1.UCH
..... ..... .....
.....
```

2.5 Розробка алгоритмів

Алгоритм розбору LaTeX-виразу для подальшої обробки базується на принципі розподіленого конвеєра, що забезпечує послідовну трансформацію введеної користувачем формули з текстового формату LaTeX у виконуваний машинний код.

Процес обробки складається з трьох ієрархічних рівнів, кожен з яких виконує свою специфічну задачу:

– нормалізація та серіалізація: Первинний лексичний аналіз LaTeX-рядка, відокремлення умов та побудова проміжного JSON-представлення виразу;

– структурування та UI-генерація: Десеріалізація JSON у суворі типи даних C#, аналіз дерева об'єктів для виявлення змінних та динамічного створення елементів керування інтерфейсом;

– математична інтерпретація та трансляція: глибокий символьний розбір виразу бібліотекою SymPy [30] та генерація оптимізованого вихідного коду мовою Fortran [31].

Деталізовану логіку взаємодії модулів та потоки даних на кожному етапі наведено на блок-схемі, що на рисунку 2.4.

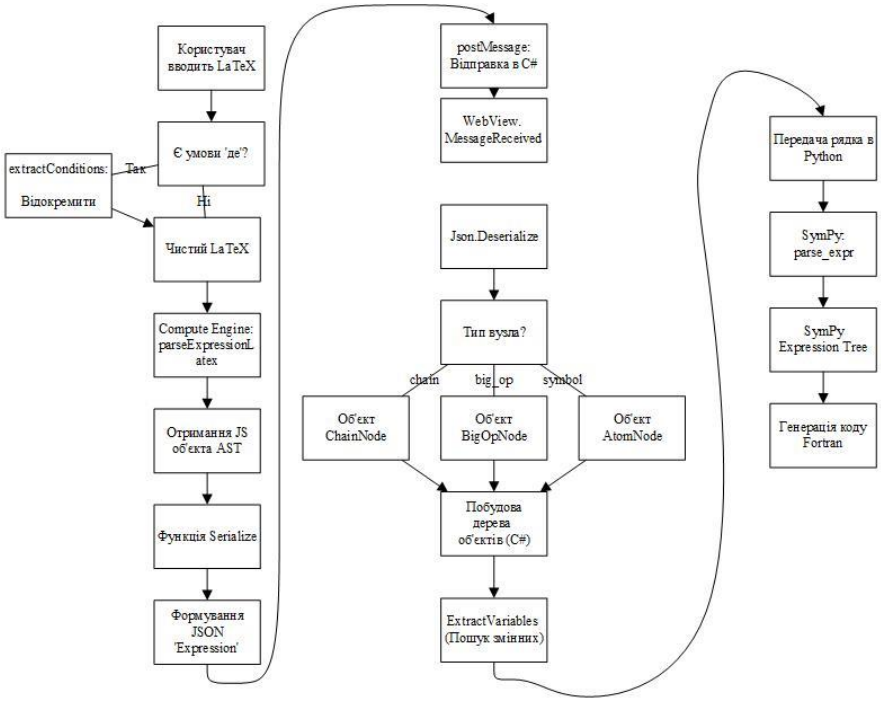

Рисунок 2.4 – Алгоритм розбору LaTeX-виразу

Алгоритм генерації коду як дерева реалізовано шляхом використання підсистеми обчислень, що використовує технологію транспіляції, що забезпечує автоматичне перетворення математичних моделей з абстрактного опису LaTeX-виразу у високоефективний машинний код.

Алгоритм базується на патерні шаблонного методу та включає етапи символьного парсингу бібліотекою SymPy мови програмування Python, трансляції синтаксису у стандарт Fortran 90 мови програмування Fortran та ін’єкції коду в шаблон програми.

Завершальними етапом є компіляція згенерованого файлу засобами GNU GFortran із застосуванням прапора оптимізації -O3, що дозволяє досягти більшої швидкодії обчислень, що обумовлено необхідністю мінімізації часу виконання циклічних обчислень.

Послідовність етапів трансформації даних від вхідного рядка до виконуваного файлу наведено на блок-схемі, що на рисунку 2.5.

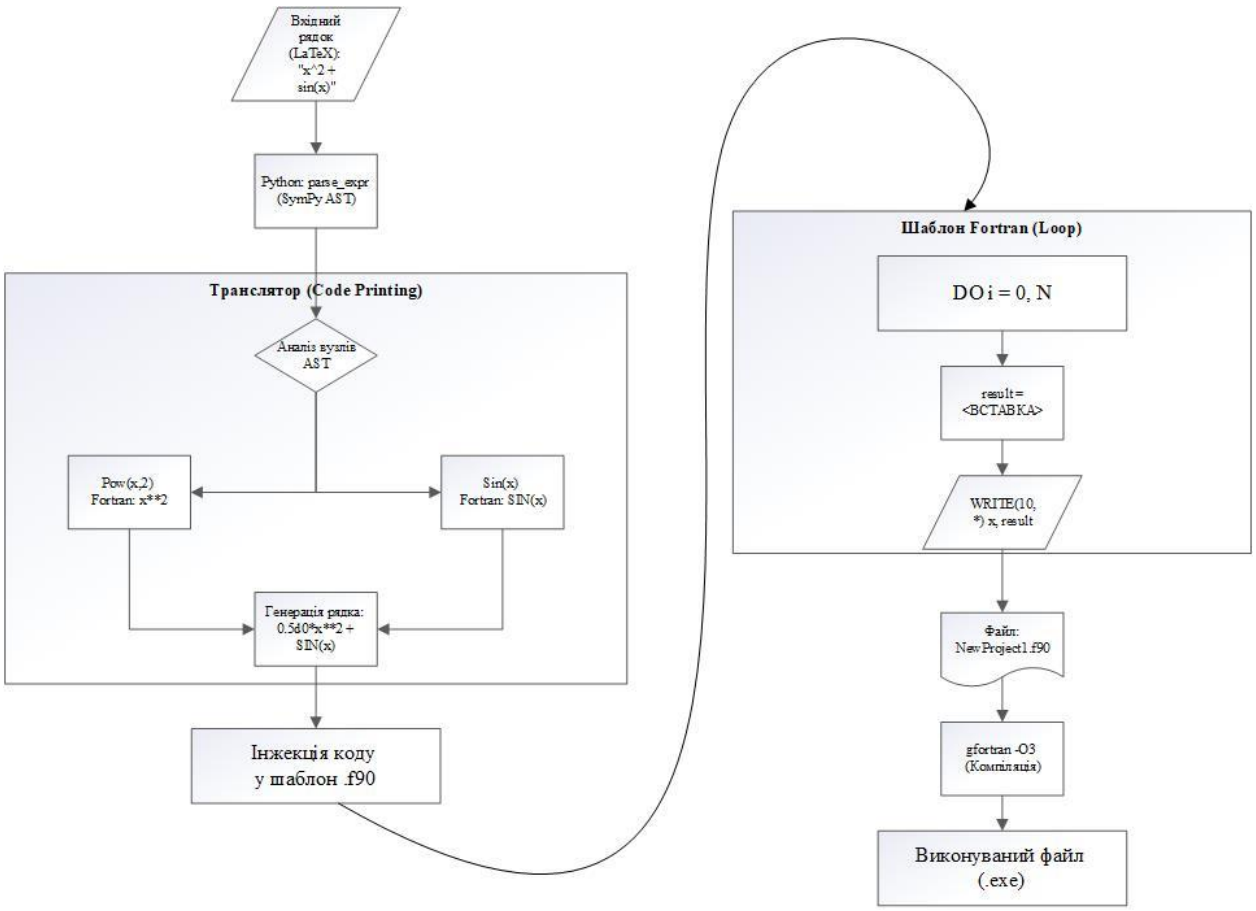

Рисунок 2.5 – Алгоритм генерації коду перетворення абстрактного синтаксичного дерева в код Fortran

Розроблений алгоритм роботи чисельного дискретного табулювання функції на основі гібридного символьно-чисельного підходу базується на генеративному програмуванні і розділений на дві функціональні фази:

– символьна перекомпіляція виконується попередній аналіз вхідного виразу, його алгебраїчне спрощення та аналітичне розв’язання диференціальних або інтегральних операторів;

– чисельне виконання виконується шляхом компіляції згенерованої оптимізованої завдяки -O3 рівню математичної моделі у машинний код, що забезпечує пряму трансляцію формули у векторні інструкції процесора для вискошвидкісних ітераційних розрахунків.

Логіку перетворення математичної моделі та потоків даних на кожному етапі зображено на блок-схемі, що на рисунку 2.6.

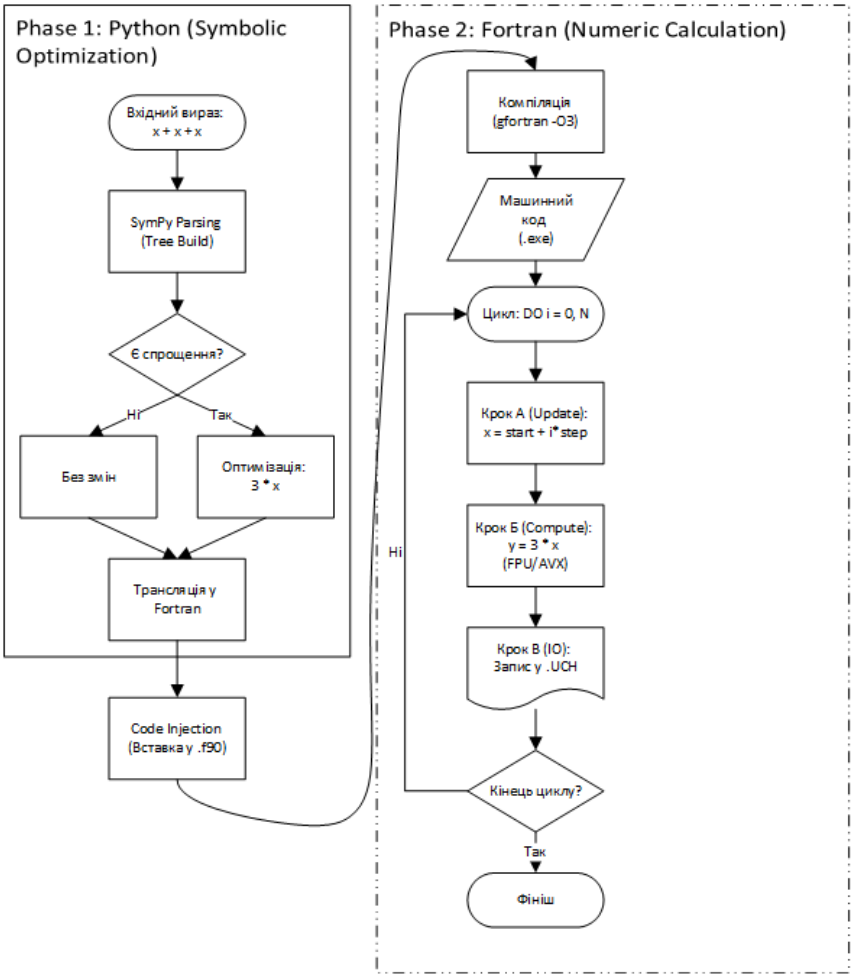

Рисунок 2.6 – Алгоритм роботи обчислювального ядра

Застосування рівня оптимізації -O3 активує агресивні методи трансформації коду, які є критично важливими для чисельного моделювання:

– автоматична векторизація: компілятор аналізує цикли та замінює скалярні інструкції на векторні, що дозволяє процесору обробляти декілька точок даних за один тактовий цикл, використовуючи регістри SSE/AVX сучасних процесорів;

– розгортання циклів: зменшує накладені витрати на перевірку умов закінчення циклу та переходи, дублюючи тіло циклу кілька разів всередині однієї ітерації;

– інбудування функцій: коли математичні функції є короткими, компілятор вставляє їх код безпосередньо в місце виклику, уникаючи витрат на створення стекових кадрів.

Оскільки архітектура нашої програми генерує код, що складається переважно з одного «важкого» циклу без складних залежностей по даних, застосування -O3 дозволяє досягти продуктивності, близької до теоретичного максимуму апаратного забезпечення.

Логіку застосування рівня оптимізації -O3 Fortran зображено на блок-схемі, що на рисунку 2.7.

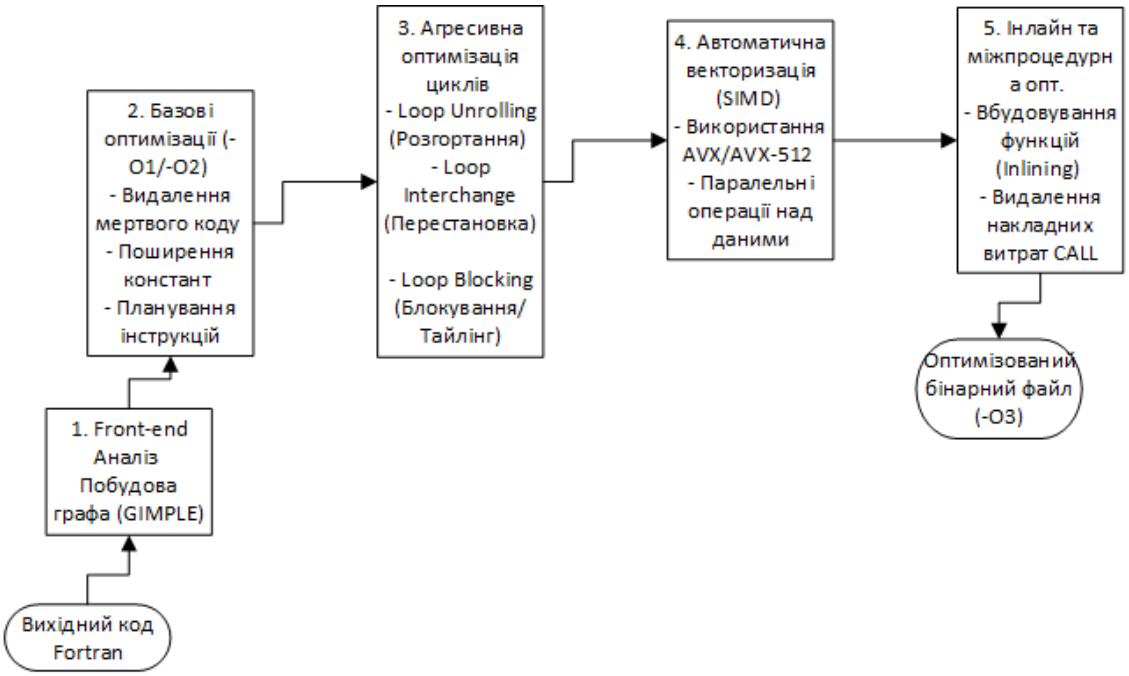

Рисунок 2.7 – Логіка застосування -O3

2.6 Проєктування інтерфейсу

Розроблюваний графічний інтерфейс користувача реалізує гібридну архітектуру, поєднуючи в собі нативні елементи управління завдяки використання класичного набору технологій C# WinForms [32] та сучасні WEB-технології для рендерингу математичних формул HTML/JS [33,34] влаштовані в WebView2.

У верхній частині головного вікна розташовано панель глобальних налаштувань, що дозволяє користувачеві задавати параметри, що є спільними для всього математичного експерименту. Панель поділена на два функціональні блоки:

– блок для введення метаданих візуалізації: заголовку експерименту, мітки осі абсцис, мітки осі ординат, що забезпечують семантичне оформлення результатів. Це дозволяє адаптувати графік під фізичний зміст задачі;

– блок для введення параметрів дискретизації: кількість розрахункових вузлів та крок дискретизації, що дозволяють користувачеві керувати точністю обчислювального процесу. Користувач задає крок дискретизації, що в свою чергу визначає приріст аргументу змінної на кожну ітерацію, та кількість ітерацій, що обмежують тривалість виконання циклу розрахунку у ядрі Fortran.

Взаємодія побудована за принципами реактивного інтерфейсу, де система автоматично адаптує набір полів введення залежно від змін у математичній формулі.

Сценарій взаємодії включає в себе:

– введення моделі: у лівій частині форми розташовано компонент MathField на базі бібліотеки MathLive, інтегрований через локальний файл Latex.html. Користувач вводить формулу у звичному математичному вигляді;

– динамічну генерацію параметрів: при зміні формули система миттєво парсить її структуру у вигляді абстрактного синтаксичного дерева. Для кожної знайденої змінної у панелі властивостей UserControl автоматично створює блок налаштувань текстових полів мінімальної та максимальної межі значень;

– налаштування візуалізації: для кожної кривої користувач може налаштувати візуальні атрибути (тип лінії, маркери, колір)безпосередньо перед розрахунком;

– запуск та результат: натискання кнопки розрахунку ініціює ланцюжок генерації коду. Результати відображатимуться в окремому вікні, що дозволяє порівнювати декілька графіків одночасно, не закриваючи головного вікна з формулою.

Для забезпеченності зручної взаємодії користувача з системою було розроблено структурну схему головного вікна, концептуальну схему, що відображає зонування функціональних блоків наведено на рисунку 2.8.

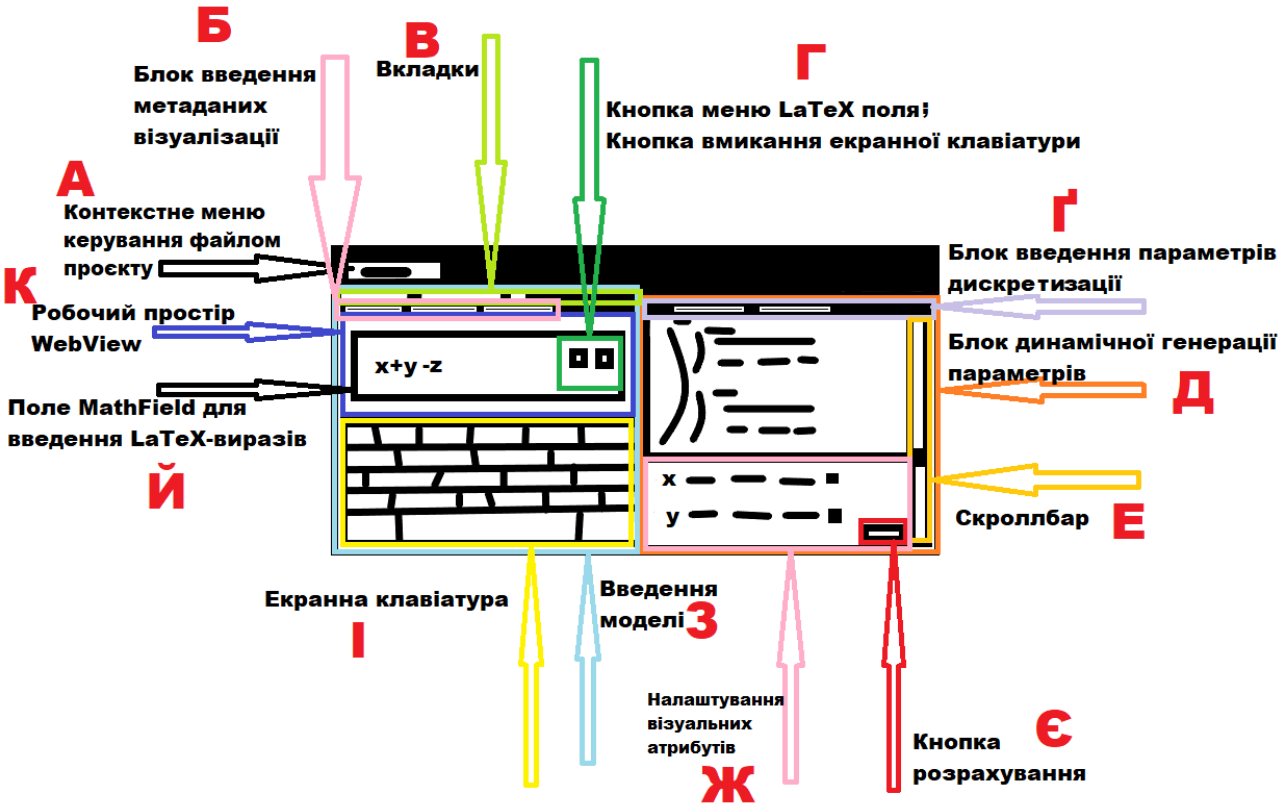

Рисунок 2.8 – Концептуальна схема головного вікна програми

Пояснення функціональних блоків:

– зона А – меню керування файлом проєкту: контекстний елемент інтерфейсу, що забезпечує швидкий доступ до файлових операцій над активною сесією. Викликається натисканням на кнопку «Файл» та надає інструментарій для створення нового проєкту, відкриття вже існуючого, збереження, збереження як окремий файл та безпечного завершення роботи з розрахунком;

– зона Б – блок введення метаданих візуалізації: текстові поля для введення текстових описів осей координат та заголовка графіка. Забезпечує надання контексту абстрактних математичних даних, дозволяючи користувачеві вказати зміст змінних для коректного відображення на фінальній діаграмі;

– зона В – панель керування вкладками: елемент керування робочим простором, де кожна вкладка представляє собою ізольований екземпляр

розрахункової сесії файлу-проєкту з власним набором виразів, змінних та налаштувань, що дозволяє користувачеві працювати з кількома математичними моделями паралельно, виконуючи порівняльний аналіз результатів без необхідності перезавантаження програми;

– зона Г – кнопка меню LaTeX-поля та кнопка активації екранної клавіатури: елементи керування полем Math-Field, що яляють собою допоміжні інтерфейсні кнопки. Кнопка налаштувань редактора містить викликає контекстне меню компонента MathLive, надає доступ до конфігурації поведінки поля введення та допоміжних команд. Кнопка-активатор віртуальної клавіатури керує видимістю екранної панелі, забезпечує швидкий доступ до введення математичних операторів та спецсимволів, відсутніх на стандартній апаратній клавіатурі;

– зона Ґ – блок введення параметрів дискретизації: текстові поля для налаштування точності та діапазону чисельного моделювання. Дозволяє користувачеві визначити загальну кількість ітерацій, що визначає обчислювальну складність та тривалість виконання згенерованого коду та крок приросту аргументу на кожній ітерації циклу, що прямо впливає на відображення графіків та точність визначення;

– зона Д – блок динамічної генерації параметрів: частина інтерфейсу програми реалізації механізму реактивної побудови інтерфейсу користувача на основі синтаксичного аналізу абстрактних синтаксичних дерев. З введеного виразу система автоматично створює індивідуальну панель налаштувань для кожної виявленої змінної та виразу, що забезпечує прямий зв’язок між математичною моделлю та інтерфейсом, дозволяючи користувачеві задавати межі визначення аргументів та атрибути стилізації відповідних кривих без необхідності ручного додавання і налаштування списку змінних користувачем;

– зона Е – скроллбар: навігаційний елемент контейнера динамічної параметризації, що активується автоматично при переповненні видимої області у випадку генерації значної кількості змінних, що забезпечує доступ до прихованих блоків налаштувань без необхідності зміни роздільної здатності головного вікна програми;

– зона Є – кнопка розрахування: інтерактивний елемент, що слугує триггером для початку математичного експерименту, виконує агрегацію всіх введених користувачем даних та передає під управління підсистемі PythonInterop [35] для виконання обчислень;

– зона Ж – налаштування візуальних атрибутів: інструментарій для кастомізації графічного представлення результатів моделювання, що дозволяє користувачеві сформувати унікальний візуальний профіль для кожної змінної або вузла, налаштовуючи параметри лінії, тип штриховки, товщину маркеру даних, форму та розмір символів, що позначають розрахункові вузли, вибір кольору кривої через системний діалог для забезпечення контрастності графіків при накладанні;

– зона З – введення моделі: інтерфейс конструювання виразів, що являє собою інтерактивну панель, яка забезпечує введення математичної моделі. Панель містить структурований набір математичних операторів, функцій, згрупованих за вкладками, що дозволяє користувачеві швидко формувати складні рівняння кліками миші, усуваючи необхідність ручного набору текстових команд LaTeX для спецсимволів;

– зона І – екранна клавіатура: панель введення, вбудована в компонент MathLive, що забезпечує доступ до розширеного набору математичних операторів, грецьких літер та функцій, відсутніх на стандартних апаратних пристроях введення. Клавіатура структурована за категоріями, що дозволяє користувачеві швидко конструювати синтаксично коректні вирази без необхідності ручного набору LaTeX-команд;

– зона Й – поле MathField для введення LaTeX-виразів: інтерактивний редактор формул, реалізований на базі веб-компонента MathLive, що забезпечує візуальне редагування математичних виразів, конвертуючи введені користувачем символи у формат розмітки LaTeX. Підтримує синтаксичну підсвітку, багаторівневу структуру рівнянь та двосторонню синхронізацію даних між графічним інтерфейсом та програмним ядром;

– зона К – робочий простір WebView: контейнер гібридної веб-інтеграції, що інкапсулює вбудований браузерний елемент керування на базі технології

Microsoft Edge WebView2. Слугує середовищем виконання для локального HTML/JS-інтерфейсу редактора MathLive, що забезпечує високоякісний графічний рендеринг математичної верстки та реалізує двосторонній канал комунікації між логікою фронтенду та основним процесом C#-додатку.

2.7 Висновки до розділу 2

Під час виконання роботи було проведено комплексне проєктування програмного забезпечення комп'ютерного моделювання кіберфізичних систем, спрямованого на створення відкритої високопродуктивної альтернативи існуючим пропрієтарним аналогам.

Для досягнення цієї мети було проведено аналіз існуючих рішень, який виявив їхні недоліки для специфічних задач вбудованого моделювання: високу вартість ліцензій, закритість коду ядра та надмірну вимогливість до ресурсів.

Розроблено концепцію і архітектуру системи, що базується на принципах багатомовного програмування.

Було запропоновано гібридну структуру, де інтерфейс та оркестрація реалізовані мовою програмування C#, символьні обчислення та генерація коду – мовою Python з бібліотекою SymPy, а безпосередні чисельні розрахунки виконуються скомпільованим ядром на Fortran.

Цей підхід дозволив поєднати зручність розробки інтерфейсу з максимальною продуктивністю обчислень.

Було виконано декомпозицію процесів функціонування системи в нотації IDEF0, що дозволило чітко визначити потоки даних та функції кожного модуля.

Спроєктовано структури даних, зокрема внутрішнє представлення формул у вигляді абстрактного синтаксичного дерева в форматі JSON та власний формат файлів результатів .UCH, що забезпечило слабку зв'язність модулів та ефективний обмін великими масивами даних без використання складних механізмів міжпроцесної комунікації.

Було розроблено алгоритмічне програмне забезпечення, що включає конвеєр обробки даних: парсинг LaTeX-виразів, генерацію вихідного коду

Fortran 90 та його компіляцію компілятором GNU GFortran із застосуванням прапора оптимізації -O3, що гарантує використання векторних інструкцій процесора та високу швидкість виконання розрахунків.

Спроєктовано графічний інтерфейс користувача, який використовує сучасні веб-технології для зручного введення математичних формул та реалізує механізм динамічної генерації елементів керування на основі аналізу введеної моделі.

Отже, в даному розділі було повністю сформовано технічне завдання, розроблено структуру та алгоритми, необхідні для програмної реалізації системи комп’ютерного моделювання кіберфізичної системи з використанням колекції компіляторів GNU.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОМП'ЮТЕРНОГО МОДЕЛЮВАННЯЯ КІБЕРФІЗИЧНИХ СИСТЕМ З ВИКОРИСТАННЯМ КОЛЕКЦІЇ GNU

3.1 Аналіз та обґрунтування вибору технологій розробки системи

Розробка програмного комплексу для комп'ютерного моделювання КФС як високоорганізованої, гнучкої системи може бути поданою як типовий процес, що починається з визначення головної мети функціонування системи та передбачає її чітку архітектурну диспозицію.

Таку диспозицію зображено на рисунку 3.1.

Головна мета

Розробка ефективного та гнучкого програмного забезпечення для комп'ютерного моделювання кіберфізичних систем із використанням колекції компіляторів GNU, що забезпечує підтримку процесів автоматизації, цифрової та зеленої трансформації відповідно до принципів Industry 5.0 [36].

Цілі другого рівня

Мінімізація витрат часу та обчислювальних ресурсів на моделювання, налагодження та оптимізацію кіберфізичних систем. Забезпечення:		
підвищення точності, швидкодії та надійності процесів моделювання	← →	створення єдиного середовища інтеграції апаратної та програмної частин моделі з використанням інструментів GNU
покращення ефективності роботи дослідників і розробників	← →	підвищення рівня автоматизації процесів моделювання кіберфізичних систем

Цілі третього рівня

Підвищення організаційності (ефективності) процесів комп'ютерного моделювання – реалізація основних принципів побудови кіберфізичних систем і автоматизованих технологій:		
Принципи моделювання та автоматизації	↔	Принципи інноваційності та сталого розвитку
Модульність	↔	Інтегрованість
Стандартизація	↔	Універсальність
Гнучкість	↔	Адаптивність
Надійність	↔	Відмовостойкість
Паралельність	↔	Оптимальність
Прозорість	↔	Верифікованість
Автоматичність	↔	Самоналаштування
Пропорційність	↔	Енергоефективність
Відтворюваність	↔	Циркулярність
Керованість	↔	Самоорганізація

Рисунок 3.1 – Схема узагальненого алгоритму визначення мети та цілей створення програмного забезпечення для комп'ютерного моделювання КФС із використанням колекції компіляторів GNU

Процес імплементації ПЗ ґрунтується на необхідності створення відкритої та високопродуктивної системи. Вибір технологічного стеку обумовлений вимогами до швидкодії чисельних методів та зручності інтерфейсу користувача.

Розробка моделі програмного забезпечення ґрунтується на теоретичних засадах гібридних обчислювальних систем: поєднанні високопродуктивного компільованого ядра GFortran для чисельних розрахунків та гнучкої скриптової мови для аналізу даних та візуалізації.

Організаційна модель розробленої системи включає два основні, функціонально незалежні компоненти: зовнішнє обчислювальне ядро та модуль візуалізації. Реалізація за принципом слабкої зв'язаності компонентів дозволяє незалежно оновлювати модуль інтерфейсу та обчислювальне ядро.

Для реалізації було обрано наступний стек технологій:

– C# .NET 9.0 [37] та WinForms використовуються для побудови класичного інтерфейсу головного вікна програми та оркестрації процесів керування системою. Вибір ґрунтується на можливості інтероперабельності зв'язки мов програмування Python-C# завдяки використанню бібліотеки pythonnet, наявності інструментів для роботи з системними процесами та розвиненою моделлю подій;

– Microsoft Edge WebView2 – інтегрований браузерний рушій на базі Chromium, який використовується для рендерингу математичних формул за допомогою бібліотеки MathLive, що дозволяє відображати складну математичну розмітку LaTeX без необхідності створення власного графічного рушія;

– Python використовується як проміжний інструмент символьних обчислень. Завдяки бібліотеці SymPy стає можливим проведення аналітичного диференціювання, спрощення виразів та трансляцію математичних формул в код Fortran, що значно спрощує розробку програмного забезпечення порівняно з написанням власного парсера;

– GNU GFortran – компілятор, що забезпечує перетворення згенерованого коду у високооптимізований бінарний файл, під час перетворення яких використовуються прапор оптимізації -O3, що дозволяє досягти максимальної продуктивності виконання ітераційних обчислень.

3.2 Розробка інтерфейсу програми

Інтерфейс програми реалізовано як гібридний застосунок, де основна взаємодія користувача з математичною моделлю відбувається через веб-компонент WebView2, що вбудовано у головне вікно програми через WinForms-оболонку з метою коректного відображення LaTeX-виразів.

Головне вікно програми, що зображено на рисунку 3.2, розділене на логічні зони: редактор формул, динамічна панель змінних та вузлів з налаштуваннями симуляції та візуалізації, що зображено на рисунку 3.2.

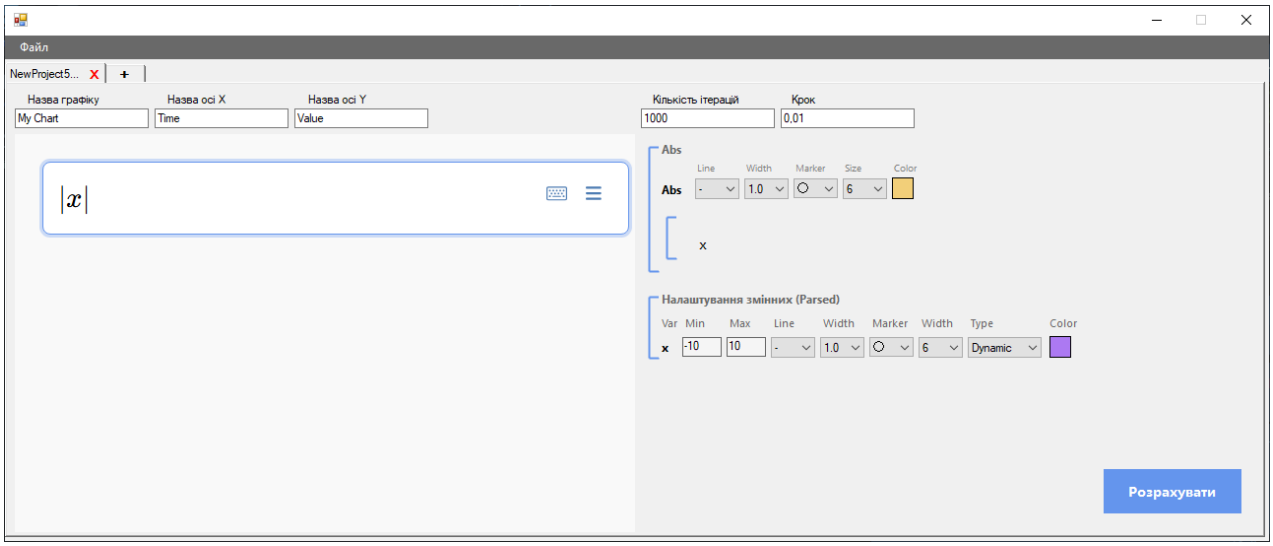

Рисунок 3.2 – Інтерфейс програми

Ключовим технічним рішенням стала реалізація двобічного зв’язку між JavaScript у WebView2 та C# Backend’у. При введенні формули компонент MathLive генерує JSON-структуру, яка описує абстрактне дерево виразу.

Процес обробки введених даних проходить наступні етапи:

– перехоплення події введення, коли C# підписується на подію WebMessageReceived від WebView2;

– десеріалізація отриманого JSON-рядку в дерево об’єктів C#;

– рекурсивний обхід дерева шляхом програмного аналізу структури виразу для ідентифікації операторів, констант та змінних.

Для полегшення взаємодії користувача з системою та усунення необхідності ручного введення складних команд LaTeX, інтерфейс оснащено спеціалізованим меню вставки математичних примітивів.

На рисунку 3.3 продемонстровано розгорнуте контекстне меню, що надає швидкий доступ до шаблонів вищої математики.

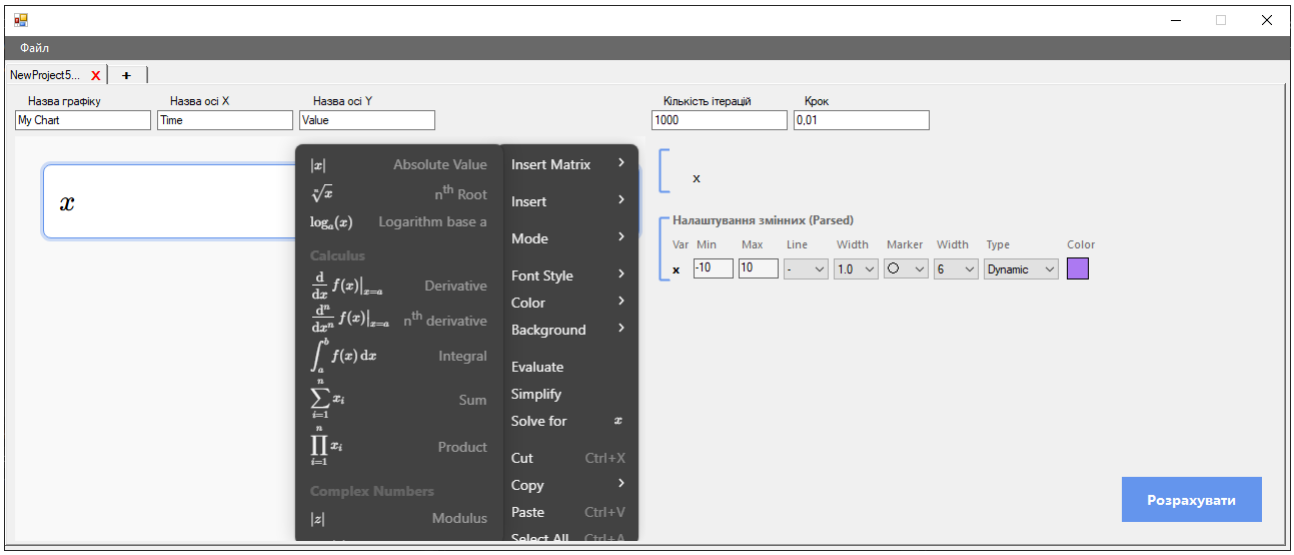

Рисунок 3.3 – Меню вибору математичних операторів та спеціальних функцій

Це меню дозволяє користувачеві оперувати готовими структурними блоками: диференціальними операторами, інтегралами, сумами, добутками та коренями n-го ступеня.

При виборі елемента з меню компонент MathLive автоматично генерує коректну LaTeX-розмітку та оновлює візуальне подання формули, що мінімізує ймовірність синтаксичних помилок на етапі введення моделі.

Структура Chain автоматично ідентифікує оператори як частину адитивного ланцюга, що дозволяє програмі обробляти вирази типу в рамках єдиної логіки, не створюючи окремих класів для операції віднімання, що оптимізує структуру абстрактного синтаксичного дерева.

На рисунку 3.4 зображено результат обробки лінійного арифметичного виразу операції додавання. Програма ідентифікує послідовність доданків як єдиний адитивний ланцюг, автоматично виділяючи змінні для параметризації.

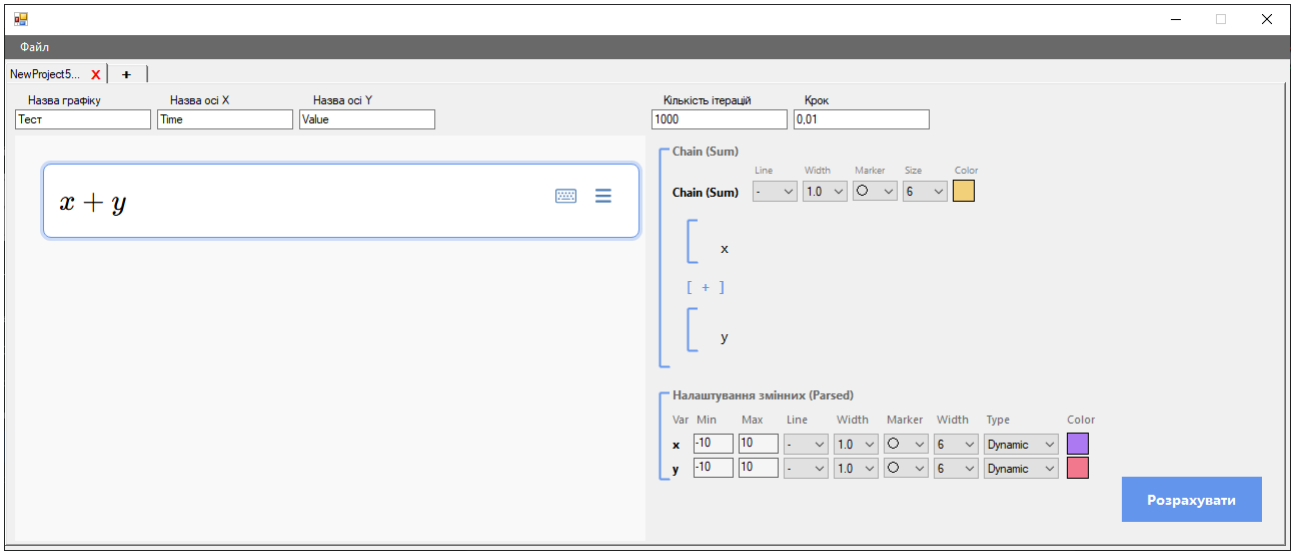

Рисунок 3.4 – Синтаксичний аналіз операції додавання

На рисунку 3.5 зображено результат обробки лінійного арифметичного виразу операції віднімання. Програма ідентифікує послідовність доданків як єдиний адитивний ланцюг, автоматично виділяючи змінні для параметризації.

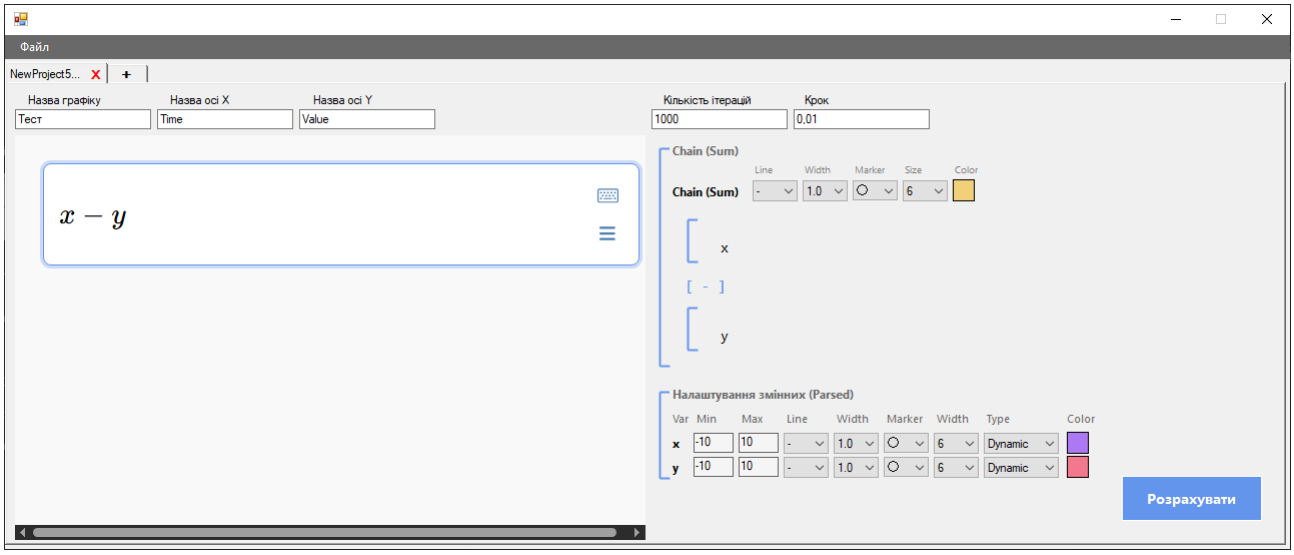

Рисунок 3.5 – Синтаксичний аналіз операції віднімання

На рисунку 3.6 зображено результат обробки лінійного арифметичного виразу операції множення. Програма ідентифікує послідовність доданків як єдиний адитивний ланцюг, автоматично виділяючи змінні для параметризації.

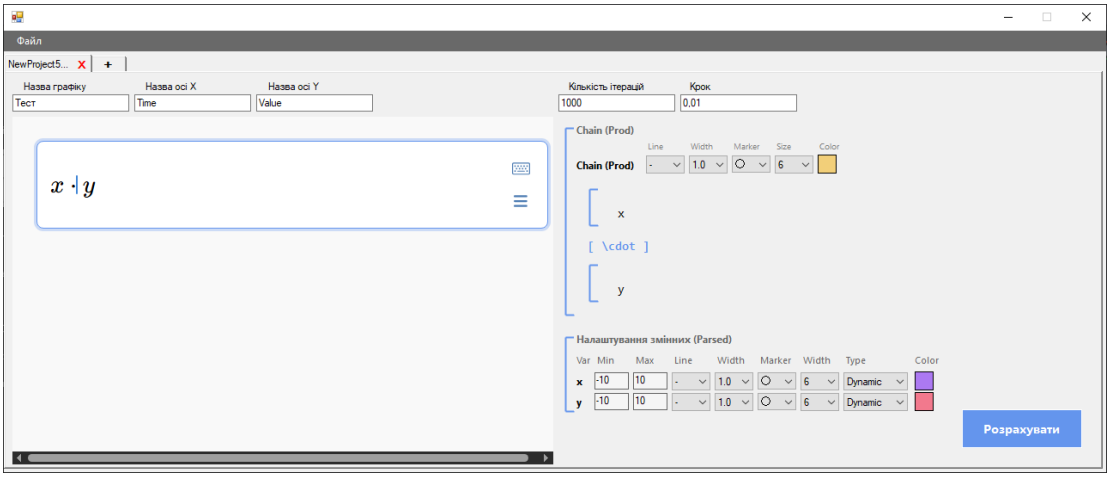

Рисунок 3.6 – Синтаксичний аналіз операції множення

На рисунку 3.7 зображено результат обробки лінійного арифметичного виразу операції ділення. Програма ідентифікує послідовність доданків як єдиний адитивний ланцюг, автоматично виділяючи змінні для параметризації.

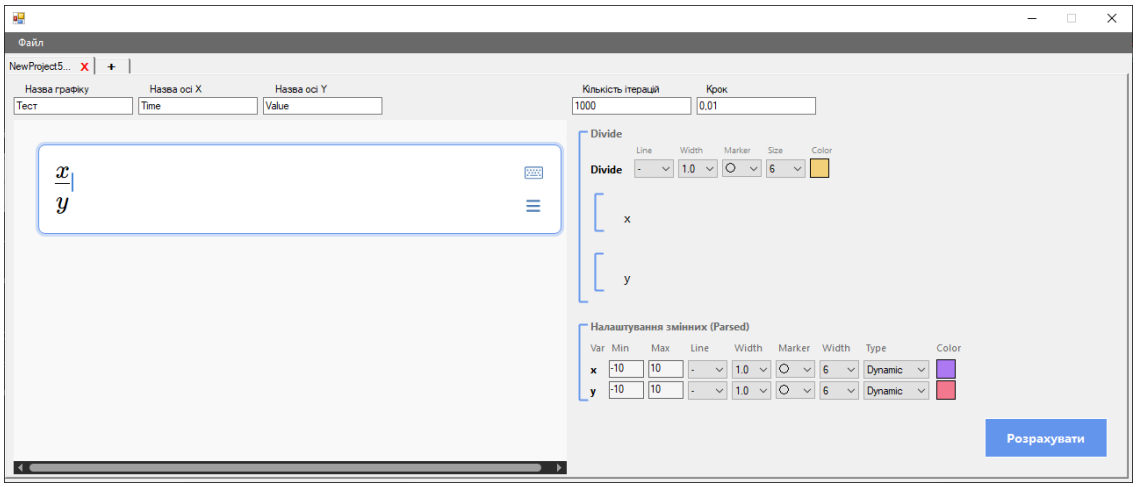

Рисунок 3.7 – Синтаксичний аналіз операції ділення

На рисунку 3.8 наведено результати роботи алгоритму для інтегральних виразів. Програма розпізнає структуру визначеного інтеграла, автоматично ідентифікуючи підінтегральну функцію, змінну інтегрування та межі інтегрування, що відображається у відповідних вузлах синтаксичного дерева.

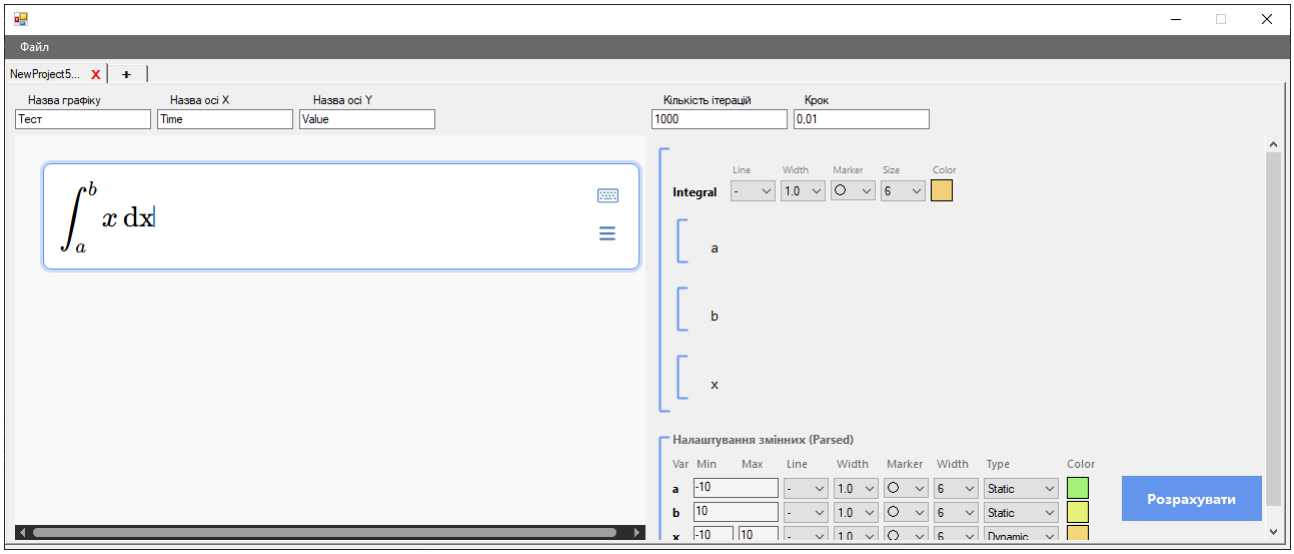

Рисунок 3.8 – Синтаксичний аналіз оператора інтегралу

На рисунку 3.9 зображено результати роботи алгоритму розпізнавання, парсингу та побудови абстрактного синтаксичного дерева для виразів похідної.

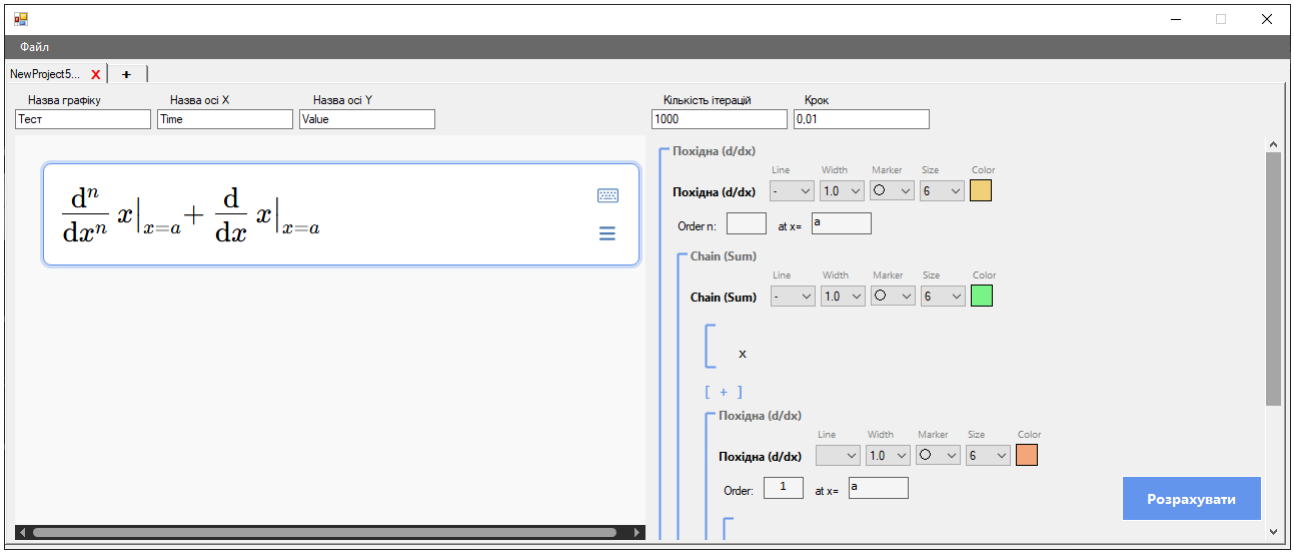

Рисунок 3.9 – Синтаксичний аналіз оператора похідних

На рисунку 3.10 наведено результати роботи алгоритму для операції обчислення суми послідовності. Програма розпізнає символ ∑, ідентифікує змінну-лічильник, межі ітерування та підіндексний вираз, формуючи відповідний вузол у синтаксичному дереві.

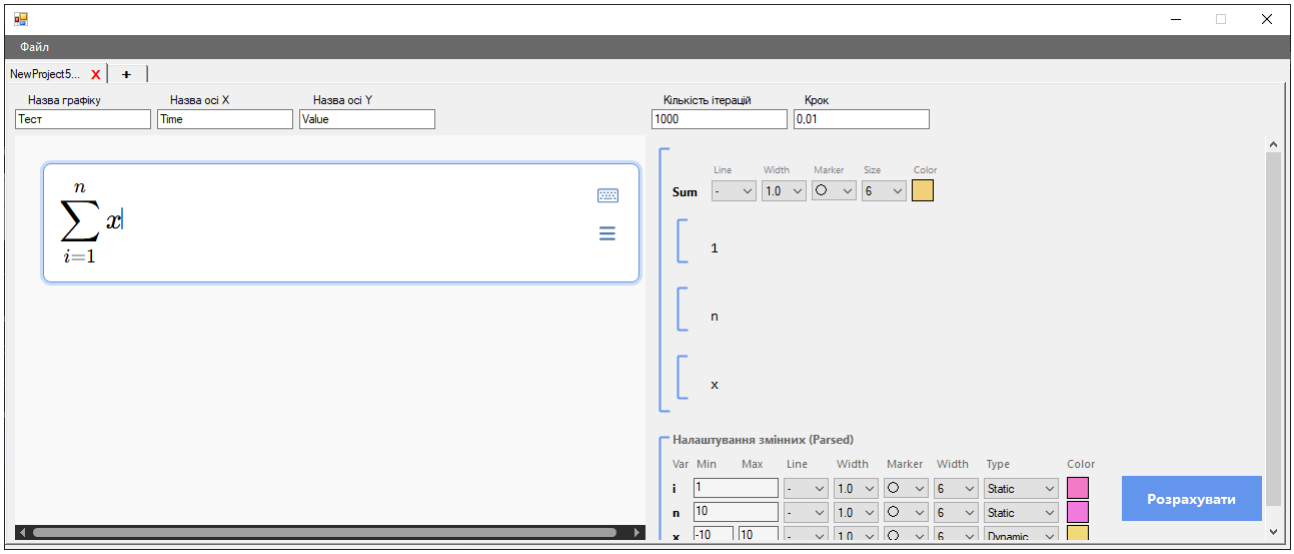

Рисунок 3.10 – Синтаксичний аналіз операції суми послідовності

На рисунку 3.11 наведено результати роботи алгоритму для операції обчислення добутку послідовності. Програма розпізнає символ П, ідентифікує змінну-лічильник, межі ітерування та підіндексний вираз, формуючи відповідний вузол у синтаксичному дереві.

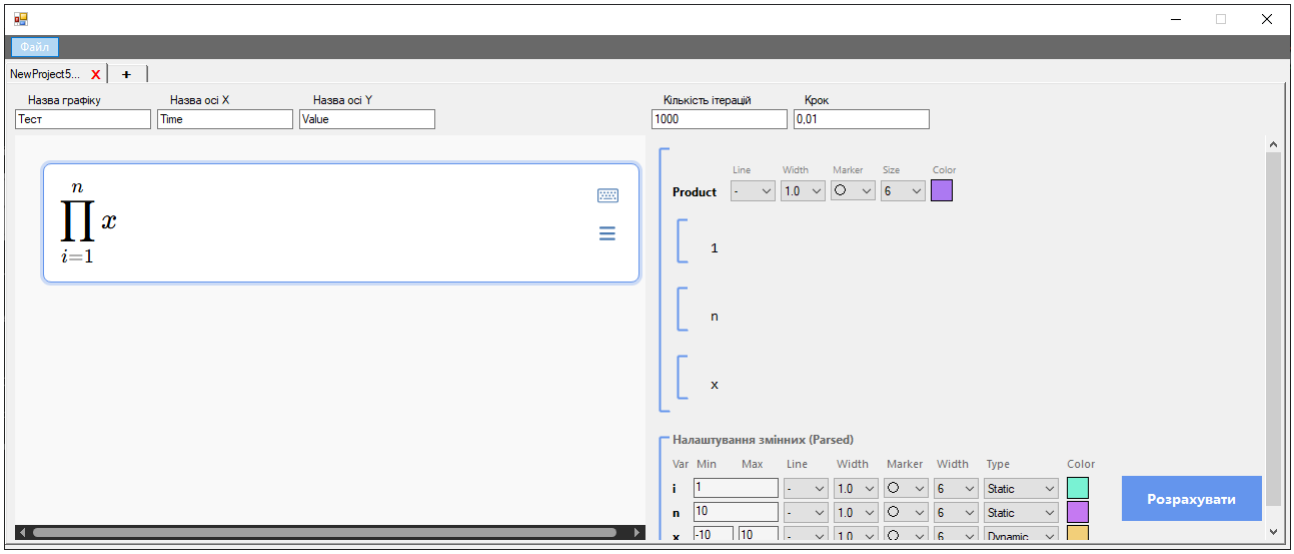

Рисунок 3.11 – Синтаксичний аналіз операції добутку послідовності

На рисунку 3.12 наведено результат обробки квадратної матриці. Інтерфейс автоматично адаптує список доступних математичних операцій, виключаючи ті, що неможливі для даної розмірності, та відображає сформоване синтаксичне дерево.

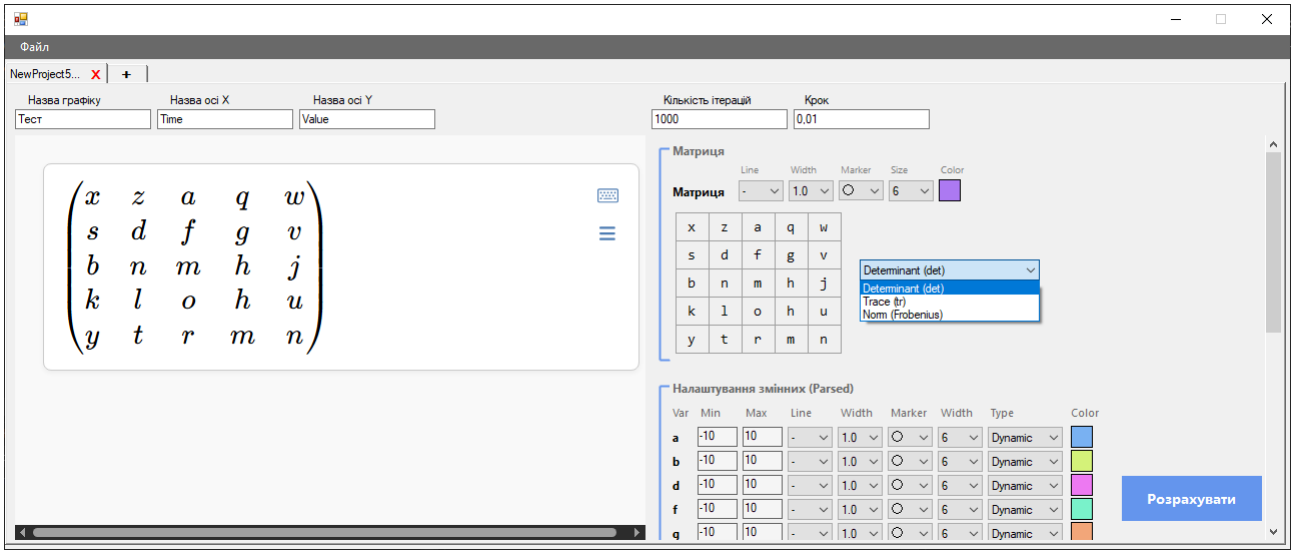

Рисунок 3.12 – Контекстні операції та результат парсингу квадратної матриці

На рисунку 3.13 наведено результат обробки неквадратної матриці. Інтерфейс автоматично адаптує список доступних математичних операцій, виключаючи ті, що неможливі для даної розмірності та відображає сформоване синтаксичне дерево.

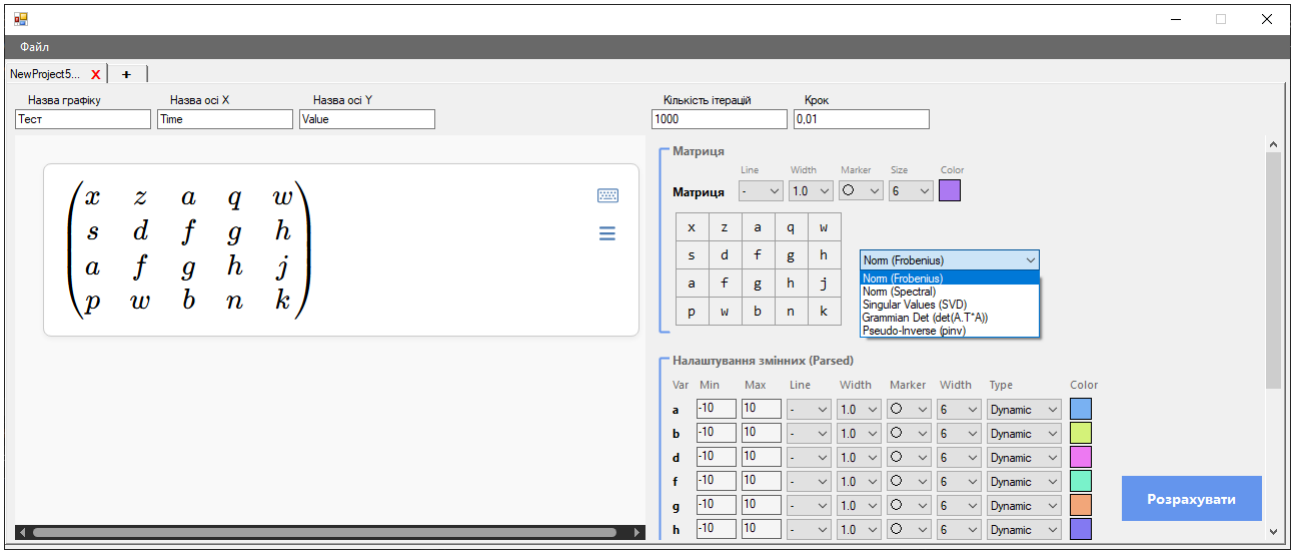

Рисунок 3.13 – Контекстні операції та результат парсингу неквадратної матриці

На рисунку 3.14 наведено результат роботи алгоритму для логарифмічних функцій. Система ідентифікує структуру логарифма, коректно розпізнаючи основу, задану у вигляді підрядкового індексу, та аргумент функції, формуючи відповідний вузол обчислювального дерева.

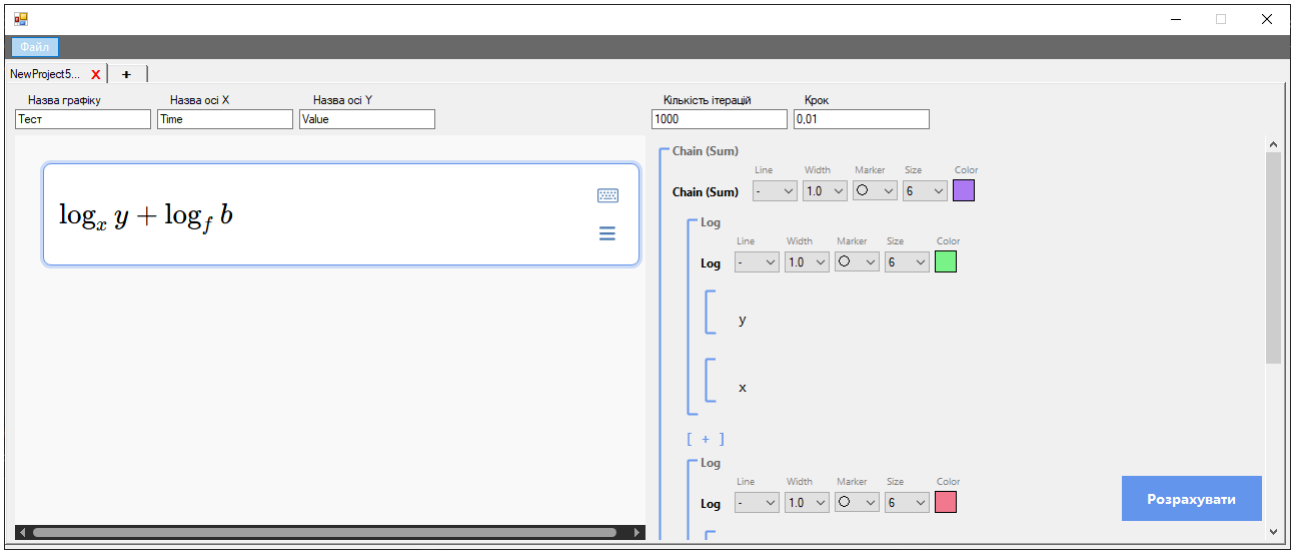

Рисунок 3.14 – Синтаксичний аналіз оператора логарифму

На рисунку 3.15 зображено результат роботи алгоритму з коренями. Система реалізує обробку загального випадку кореня n-го ступеня, автоматично ідентифікуючи показник кореня та складний підкореневий вираз, що дозволяє коректно будувати дерево обчислень для будь-яких арифметичних коренів.

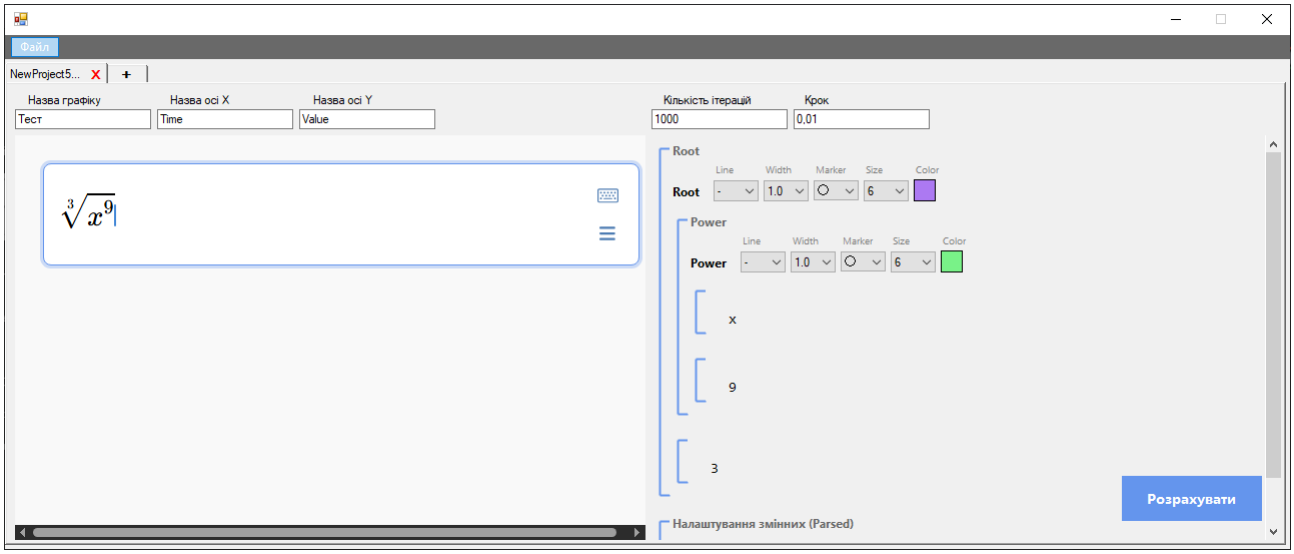

Рисунок 3.15 – Синтаксичний аналіз оператора кореня n-го ступеня

На рисунку 3.16 зображено роботу алгоритму з функцією абсолютної величини . Система автоматично розпізнає синтаксис вертикальних обмежувачів, формуючи вузол Abs у дереві об’єктів, що забезпечує коректну трансляцію виразу у відповідну інструкцію Fortran (ABS(x)) для подальших чисельних розрахунків.

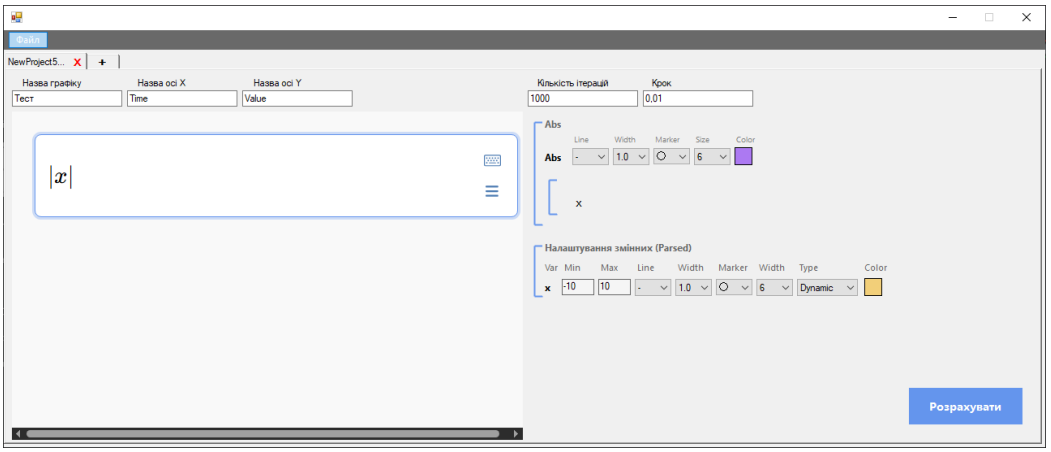

Рисунок 3.16 – Синтаксичний аналіз оператора абсолютної величини

Важливою особливістю реалізації програмного комплексу є інтегрована підтримка файлових операцій у контексті проєкту. Кожній розрахункової сесії система автоматично створює ізольоване файлове оточення, що забезпечує структуроване зберігання даних та унеможливлює конфлікти імен файлів при паралельних розрахунках. На рисунку 3.17 наведено контекстне меню вкладки, через яке отримуєтьсяшвидкий доступ до управління поточним розрахунком.

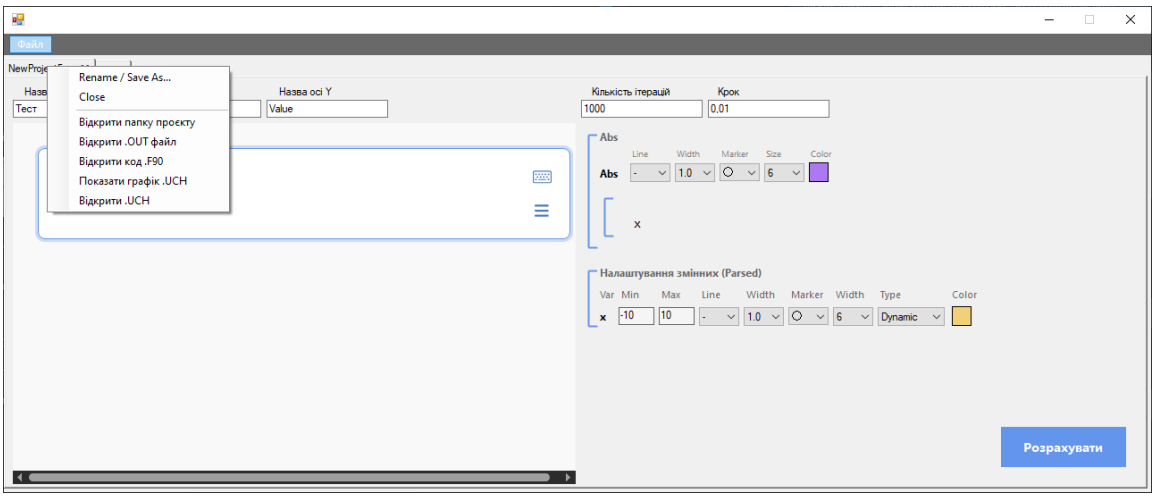

Рисунок 3.17 – Контекстне меню керування файлами проєкту

Функціонал меню дозволяє користувачеві миттєво відкривати згенеровані артефакти: вихідний код мовою Fortran, файли даних для візуалізації та текстові звіти, а також переходити до робочої директорії у провіднику Windows.

На рисунку 3.18 зображено результуючу структуру файлів проєкту, яка генерується автоматично після запуску симуляції.

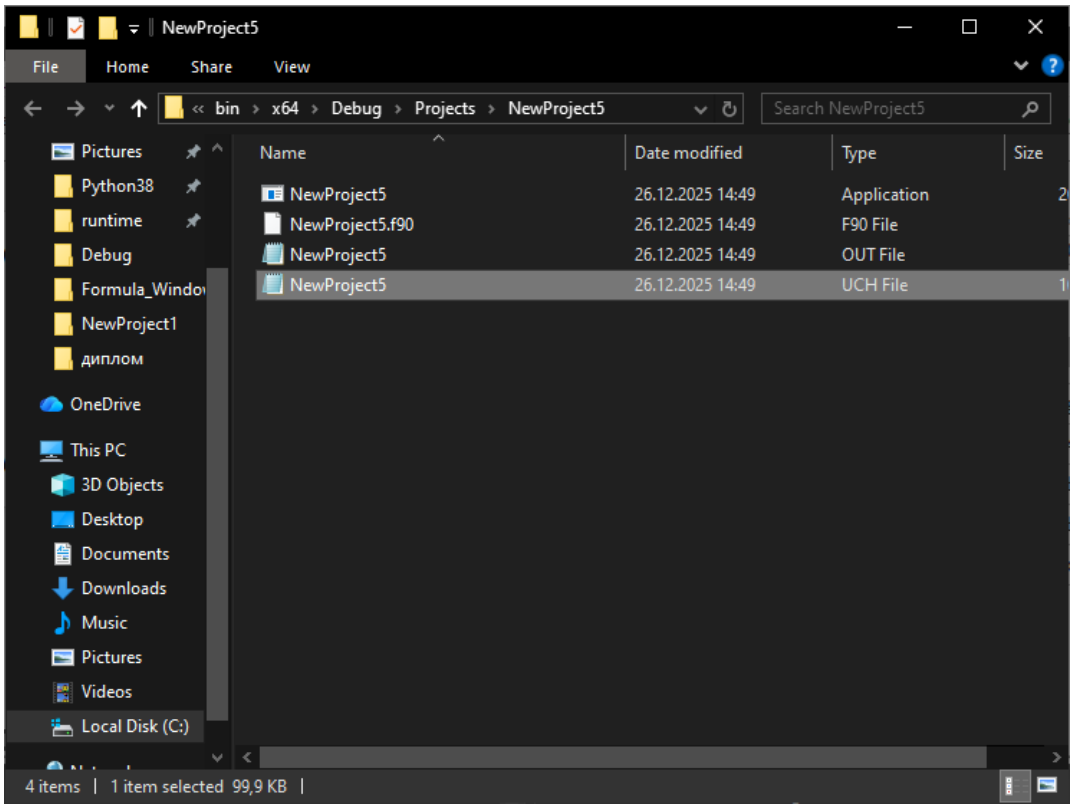

Рисунок 3.18 – Структура файлів проєкту

Програма створює окрему папку для кожного експерименту, де зберігаються всі компоненти моделювання:

– f90 згенерований вихідний код математичної моделі;
– exe скомпільований виконуваний файл обчислювального ядра;
– UCH файл з результатами розрахунку для побудови графіків;
– OUT файл-звіт із метаданими та параметрами симуляції.

3.3 Розробка Транспілятора та обчислювального ядра

Підсистема транспіляції відповідає за перетворення розпарсеного абстрактного синтаксичного дерева у виконуваний код. Цей процес автоматизовано і приховано від користувача.

Алгоритм генерації коду працює наступним чином:

– символьна обробка являє собою C#-передачу математичного рядку у скрипт Python, який використовує sympy.fcode для конвертації виразу у синтаксис Fortran 90;

– ін’єкція шаблону являє собою вставку отриманого фрагменту коду, що вставляється у заздалегідь підготовлений шаблон програми Fortran (template.f90), який містить оголошення змінних, цикли ітерацій за часом та процедури запису файлів;

– компіляція являє собою системний виклик ініціації процесу gfortran.exe, який компілює отриманий файл.

На рисунку 3.19 зображено кнопку розрахунку, що ініціює цей ланцюжок.

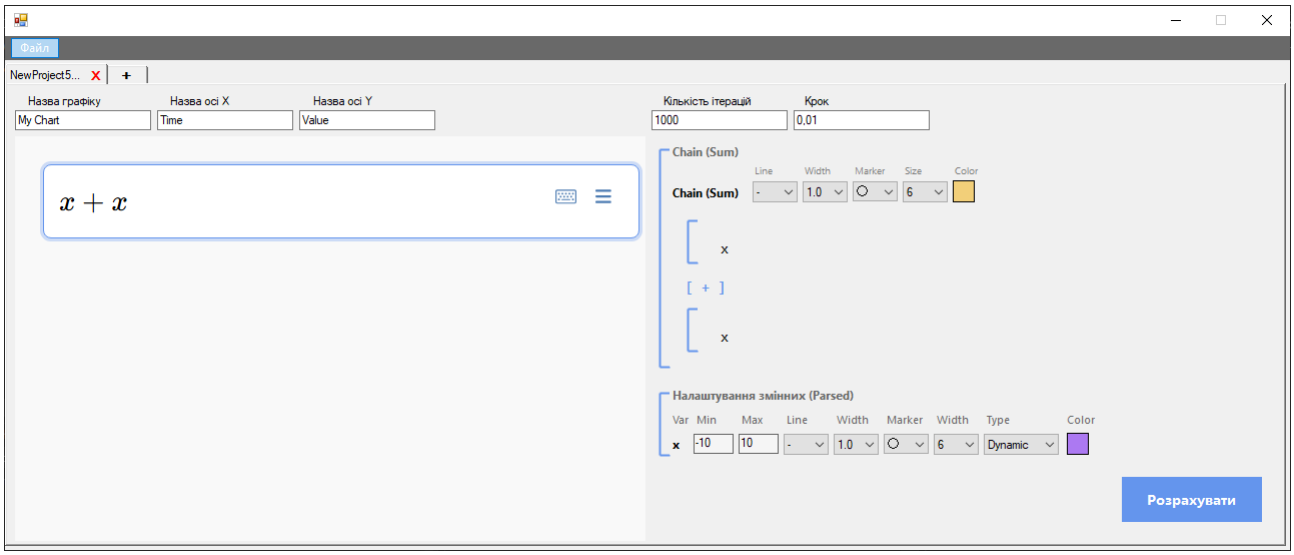

Рисунок 3.19 – Кнопка розрахунку

Успішне завершення компіляції та розрахунку підтверджуєтсья системним повідомленням, що зображено на рисунку 3.20.

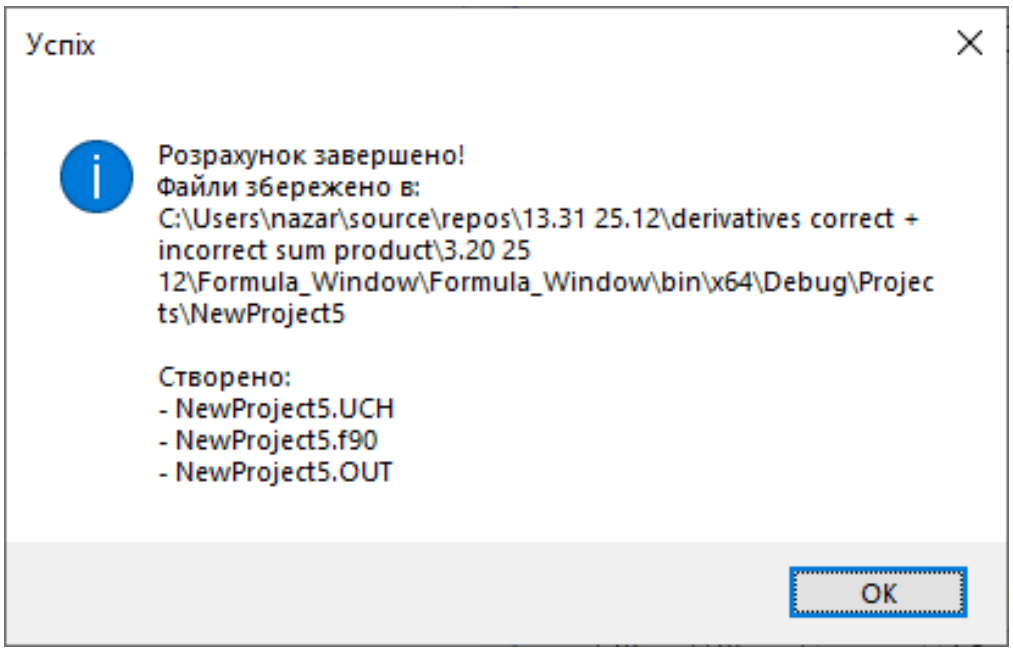

Рисунок 3.20 – Успішний розрахунок

На рисунку 3.21 зображено повідомлення MessageBox-результат пропозиція перейти до етапи візуалізації.

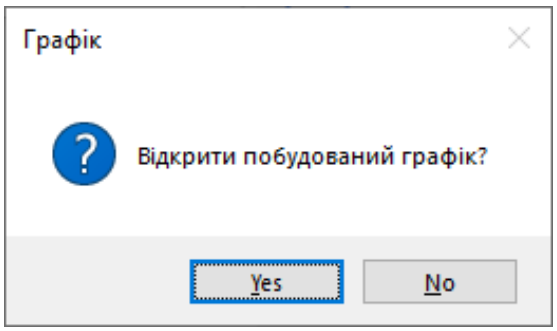

Рисунок 3.21–Підтвердження переходу до етапу візуалізації

Результатом роботи обчислювального ядра є набір файлів та графік, які забезпечують повну відтворюваність експерименту:

– файл звіту .OUT: містить детальний опис вхідних параметрів, знайдених змінних та метаданих симуляції. Цей файл є людино-читаним і слугує для верифікації параметрів моделі;

– файл даних .UCH: спеціалізований формат для зберігання результатів чисельного моделювання (рис. 3.18). Він містить масиви розрахованих точок, оптимізовані для швидкого зчитування модулем візуалізації.

– вихідний код (.f90): Згенерований код мовою Fortran (рис. 3.19), що дозволяє користувачеві за необхідності перевірити логіку розрахунків або використати код в інших проєктах.

На рисунку 3.22 зображено вміст файлу .OUT, що слугує для рекурсивної логіки моделювання.

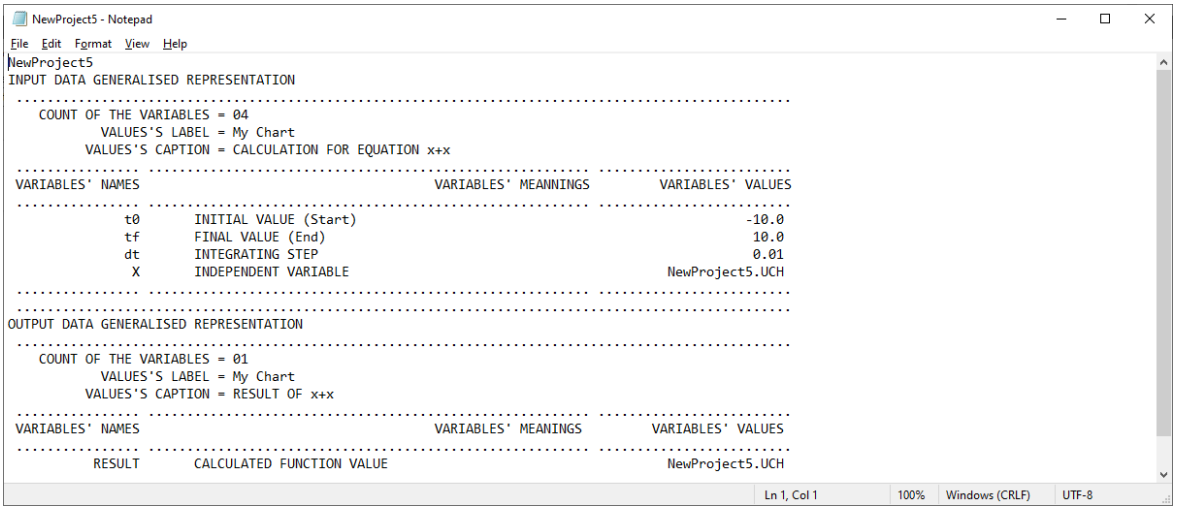

Рисунок 3.22 – Вміст файлу .OUT

На рисунку 3.23 зображено вміст UCH-файлу.

Рисунок 3.23 – Вміст .UCH-файлу

На рисунку 3.24 зображено вміст f90-файлу.

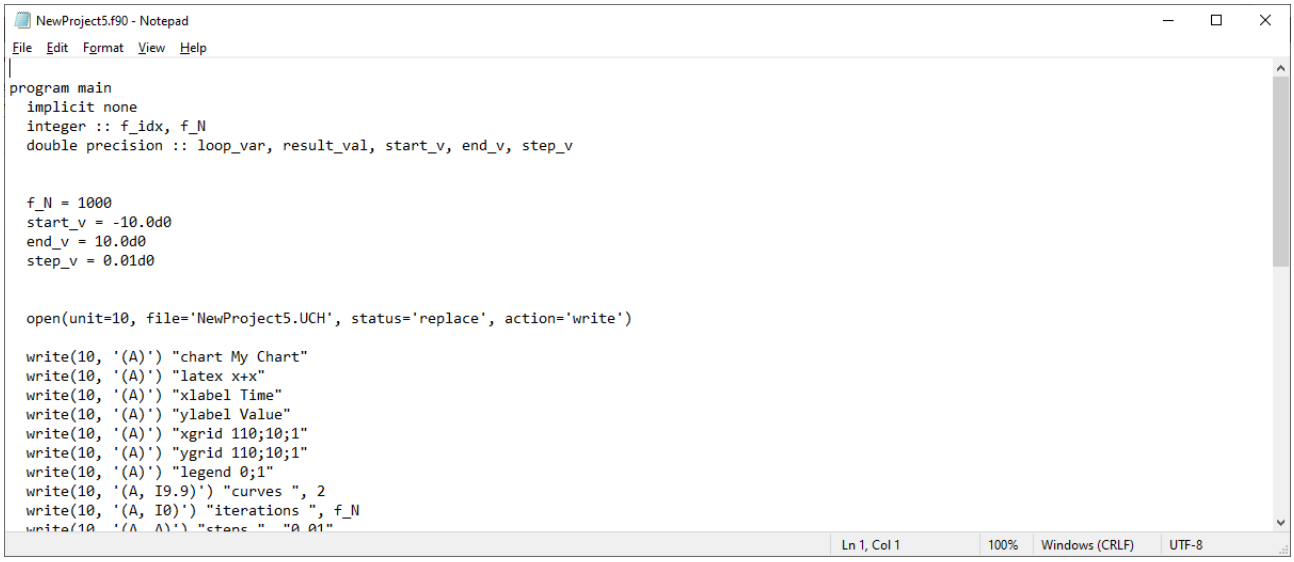

Рисунок 3.24 – Вміст F90-файлу

На рисунку 3.25 зображено результат обчислень функції x+x в вигляді графіка.

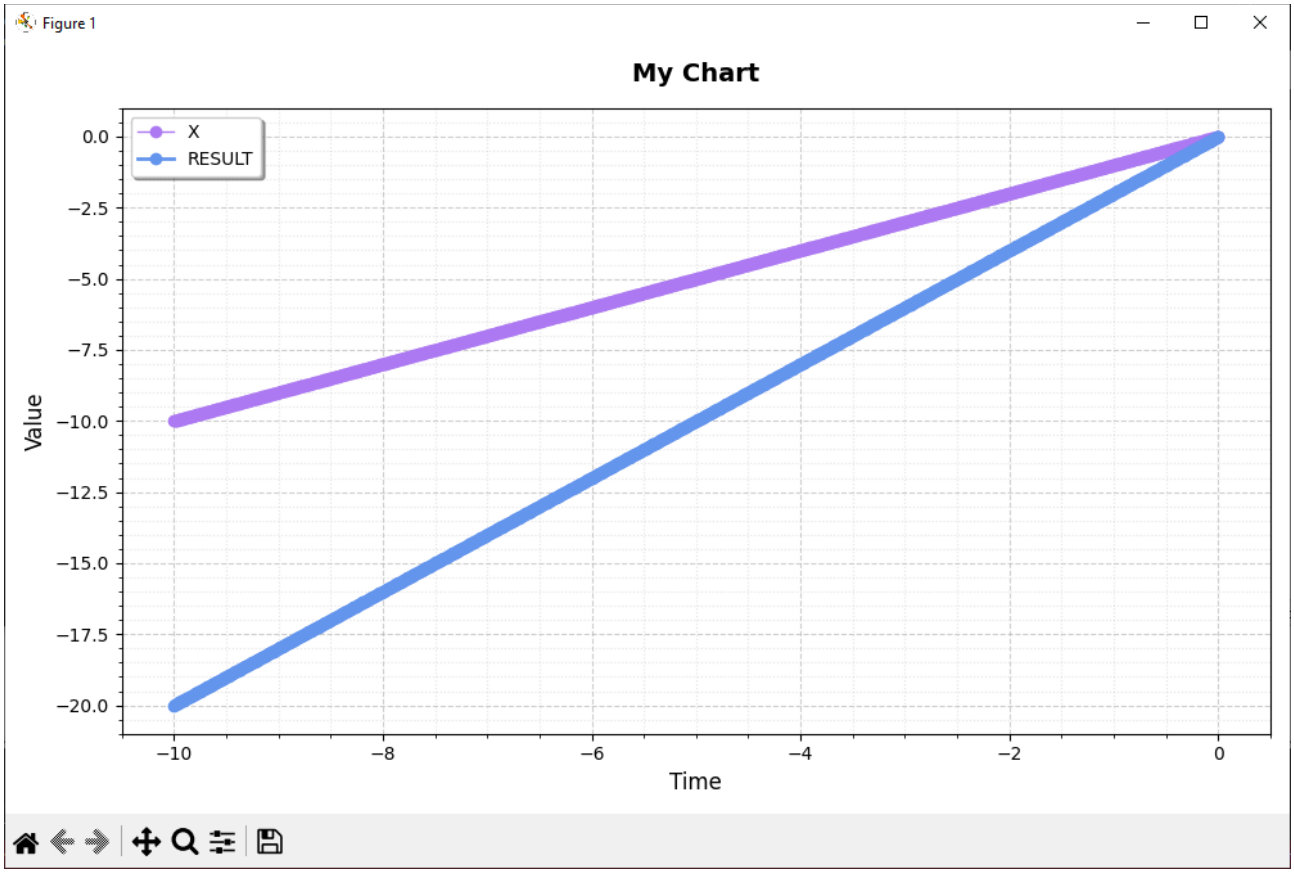

Рисунок 3.25 – Результат обчислень функції x+x в вигляді графіка

3.4 Експериментальна перевірка та аналіз результатів

3.4.1 Затихаючі коливання

Для перевірки коректності роботи розробленого програмного забезпечення обчислювального ядра та модуля візуалізації було отримано задачу моделювання перехідного процесу в динамічній системі другого порядку

В якості тестової моделі було використано функцію імітації затухаючих коливань в механічній або електричній підсистемі кіберфізичної системи:

$$e^{-0.2x}\cos(3x). \tag{3.1}$$

Метою експерименту була перевірка коректності обробки функцій експоненти та тригонометрії, точності відображень амплітуди, що експоненційно зменшується.

Параметри експерименту:

– діапазон інтегрування: xЄ[0;20];

– кількість ітерацій: n=1000;

– крок дискретизації: h=0,02.

Хід експерименту.

У поле редактора введено формулу у форматі LaTeX як зображено на рисунку 3.26.

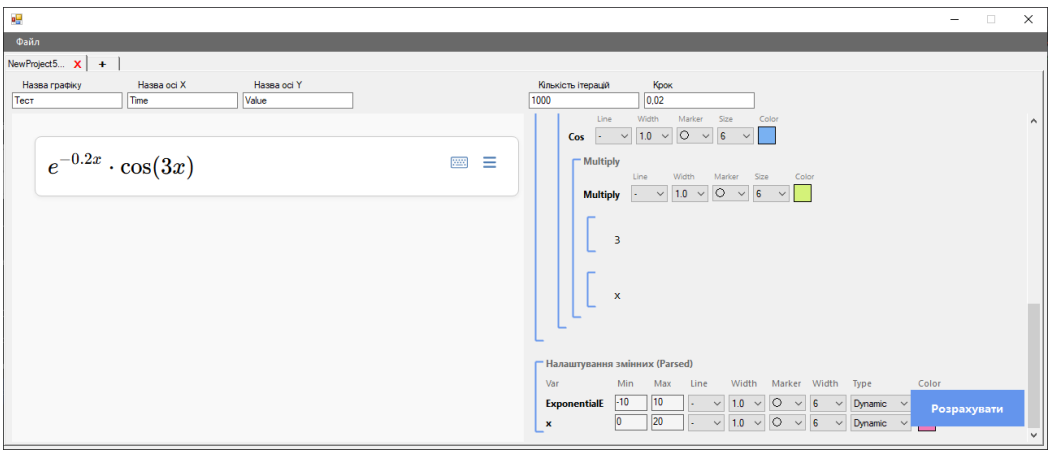

Рисунок 3.26 – Введена експериментальна формула

Система автоматично визначила змінну x та згенерувала поле для введення меж, було введено межі як зображено на рисунку 3.26 та по натисканню на кнопку розрахунку було автоматично згенеровано код Fortran, проведено компіляцію та виконання файлу і миттєво з’явилось повідомлення про успішне завершення розрахунку як зображено на рисунку 3.27.

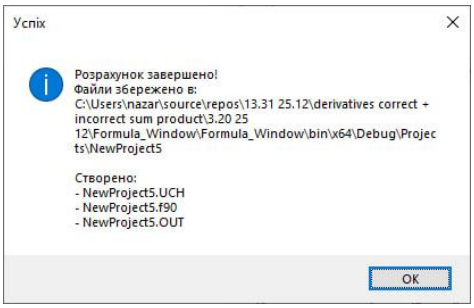

Рисунок 3.27 – Повідомлення про успіше завершення розрахунку

Далі з’явилось пропозиція показати графік як зображено на рисунку 3.28.

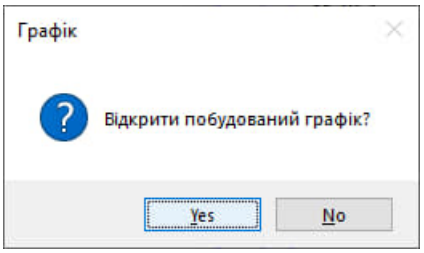

Рисунок 3.28 – Пропозиція показати графік

Натиснуто «так» і отримано графік, що зображено на рисунку 3.29 та переглянемо .UCH-файл через контекстне меню вкладки «Відкрити файл .UCH» та оглянемо зміст.

```
chart Тест
latex e^{-0.2x}\cos\left(3x\right)
ylabel Value
xgrid auto
ygrid auto
legend 0;1
curves 000000008
```

```
iterations 1000
steps 0.02
curve EXPONENTIALE
min -10
max 10
color 242207121
line 11.0
markers 016
data 000001001
  0.000000000000000E+000; -1.000000000000000E+001…
curve X
min 0
max 20
color 212242121
line 11.0
markers 016
data 000001001
  0.000000000000000E+000;  0.000000000000000E+000…
curve COS((3*X))
color 121242136
line 11.0
markers 016
data 000001001
  0.000000000000000E+000;  1.000000000000000E+000…
curve (EXPONENTIALE**NEGATE((0.2*X)))
color 121177242
line 11.0
markers 016
data 000001001
  0.000000000000000E+000;  0.000000000000000E+000
curve NEGATE((0.2*X))
color 237121242
line 11.0
markers 016
data 000001001
  0.000000000000000E+000;  0.000000000000000E+000…
```

curve (0.2*X)
color 121242202
line 11.0
markers 016
data 000001001
0.000000000000000E+000; 0.000000000000000E+000…
curve RESULT
color 100149237
line 12
markers 016
data 000001001
0.000000000000000E+000; 0.000000000000000E+000…

В результаті графік, що зображено на рисунку 3.29, демонструє динаміку затухання процесу. Система коректно відпрацювала як високочастотну складову коливання, так і експоненційне спадання, що підтверджує придатність ПЗ до моделювання систем автоматичного керування.

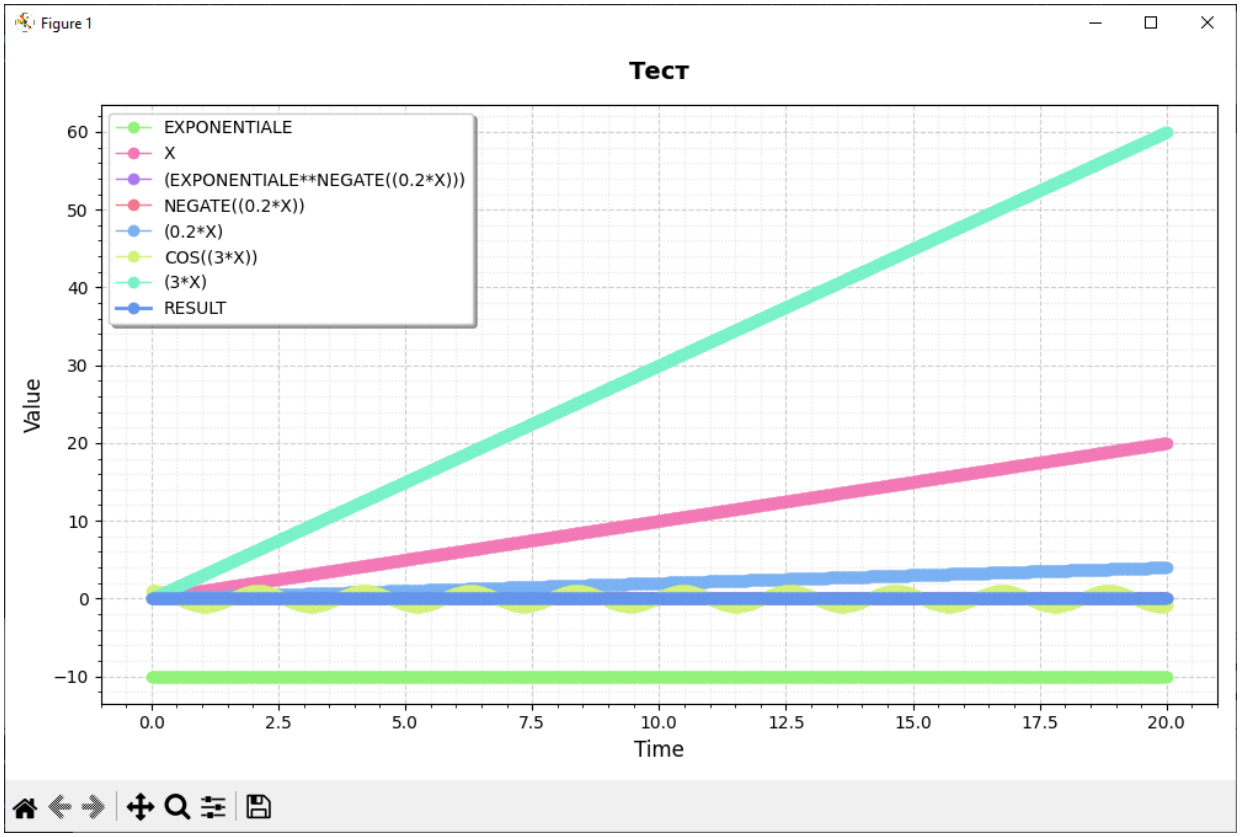

Рисунок 3.29 – Результат обчислень в вигляді графіка

3.4.2 Биття частот

Для перевірки точності дискретизації та візуалізації високочастотних сигналів було проведено моделювання явища фізичного биття (інтерференції двох хвиль з близькими частотами.

Математична модель описується сумою двох гармонік:

$$\sin(10x) + \sin(11x). \tag{3.2}$$

Метою експерименту було виявити здатність системи будувати складні інтерференційні картини без виникнення ефекту спотворення сигналу.

Параметри експерименту:

– діапазон інтегрування: xЄ[0;50];

– кількість ітерацій: n=5000;

– крок дискретизації: h=0,01.

Хід експерименту.

Системі було задано суму синусів. Транспілятор успішно розпізнав адитивний ланцюг та згенерував цикл обчислень як зображено на рисунку 3.30.

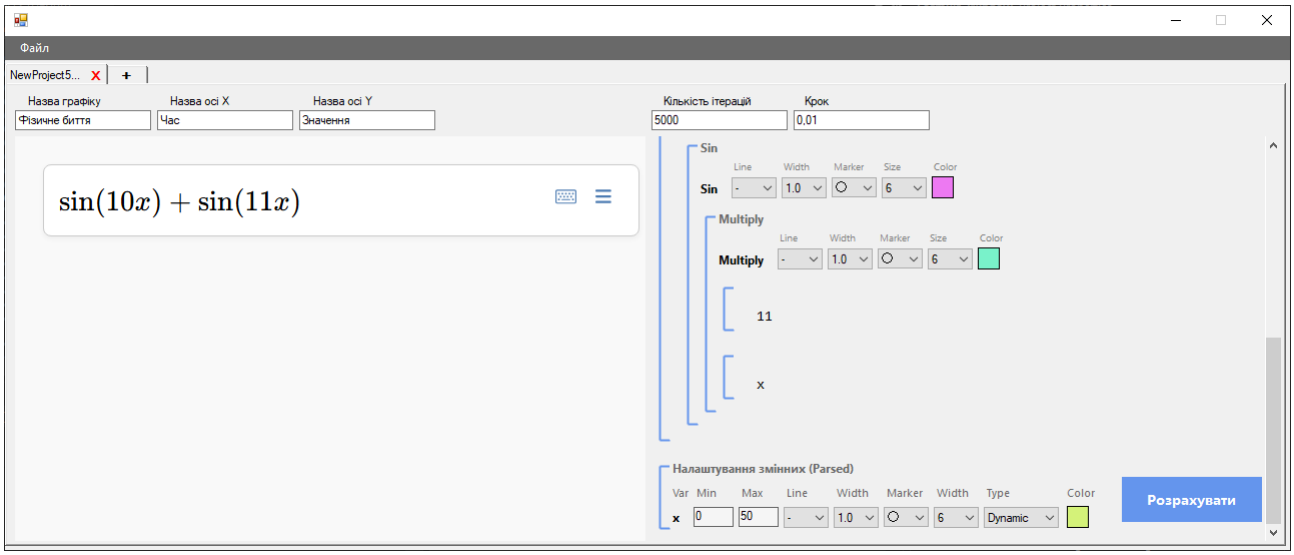

Рисунок 3.30 – Введена експериментальна формула

Система автоматично визначила змінну x та згенерувала поле для введення меж, було введено межі як зображено на рисунку 3.30 та по натисканню на кнопку розрахунку було автоматично згенеровано код Fortran, проведено компіляцію та виконання файлу і миттєво з’явилось повідомлення про успішне завершення розрахунку як зображено на рисунку 3.31.

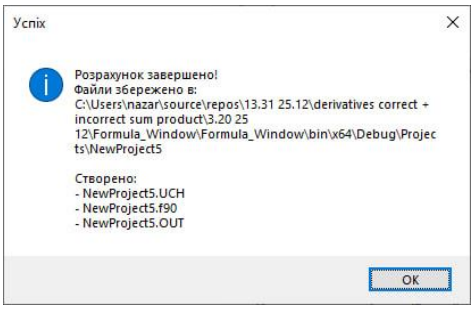

Рисунок 3.31 – Повідомлення про успішне завершення розрахунку

Далі з’явилось пропозиція показати графік як зображено на рисунку 3.32.

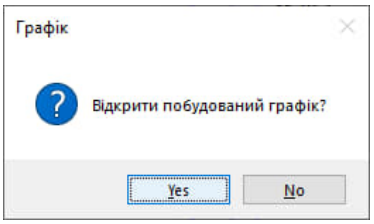

Рисунок 3.32 – Пропозиція показати графік

Натиснуто «так» і отримано графік, що зображено на рисунку 3.33 та переглянемо .UCH-файл через контекстне меню вкладки «Відкрити файл .UCH» та оглянемо зміст.

chart Фізичне биття

latex \sin\left(10x\right)+\sin\left(11x\right)

xlabel Час

ylabel Значення

xgrid 110;10;1

ygrid 110;10;1

legend 0;1

curves 000000006

iterations 5000

steps 0.01

curve X

min 0

max 50

color 212242121

line 11.0

markers 016

data 000005001

0.000000000000000E+000; 0.000000000000000E+000…

curve SIN((10*X))

color 121242136

line 11.0

markers 016

data 000005001

0.000000000000000E+000; 0.000000000000000E+000…

curve (10*X)

color 121177242

line 11.0

markers 016

data 000005001

0.000000000000000E+000; 0.000000000000000E+000…

curve SIN((11*X))

color 237121242

line 11.0

markers 016

data 000005001

0.000000000000000E+000; 0.000000000000000E+000…

curve (11*X)

color 121242202

line 11.0

markers 016

data 000005001

0.000000000000000E+000; 0.000000000000000E+000…

curve RESULT

color 100149237

line 12

markers 016

data 000005001

0.000000000000000E+000; 0.000000000000000E+000

В результаті графік, що зображено на рисунку 3.33 демонструє характерну періодичну зміну амплітуди результуючого коливання групи хвиль. Візуалізація є плавною та точною, що свідчить про високу швидкодію ядра GFortran при обробці великих масивів даних.

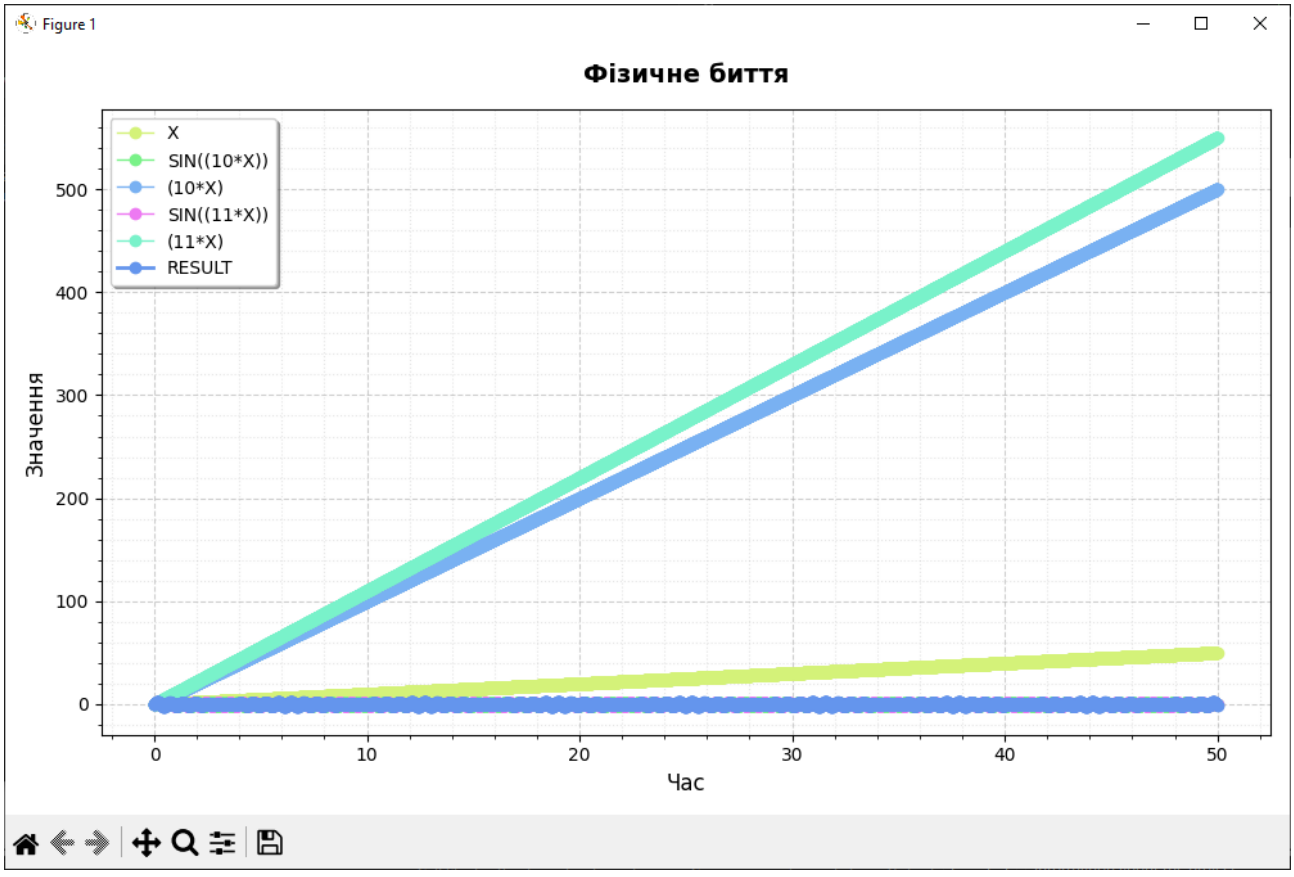

Рисунок 3.33 – Результат обчислень в вигляді графіка

3.5 Висновок до розділу 3

У цьому розділі кваліфікаційної роботи було розроблено програмну реалізацію системи комп’ютерного моделювання кіберфізичних систем на основі відкритого інструментарію та гібридної архітектури. Створене середовище поєднує графічний інтерфейс на базі C# .NET та WinForms із технологією Microsoft Edge WebView2, що разом із бібліотекою MathLive забезпечує нативну підтримку та коректне відображення LaTeX-виразів.

Впроваджено механізм автоматичної трансляції, який трансформує введені користувачем формули в абстрактне синтаксичне дерево. Завдяки інтеграції з Python та бібліотекою SymPy система забезпечує генерацію оптимізованого вихідного коду мовою Fortran 90, що дозволяє автоматизувати процес програмування чисельних методів без ручного втручання.

Високу продуктивність обчислень забезпечено інтеграцією колекції компіляторів GNU GFortran, що дозволило реалізувати конвеєр автоматичної компіляції моделей.

Спроєктовано систему керування даними, яка включає генерацію ізольованих робочих директорій та використання спеціалізованих форматів файлів. Впровадження форматів .UCH для масивів результатів та .OUT для метаданих забезпечує надійність зберігання даних та повну відтворюваність результатів кожного експерименту.

Працездатність комплексу підтверджено серією чисельних експериментів, зокрема моделюванням затухаючих коливань та явища биття частот. Результати тестування засвідчили коректність роботи синтаксичного аналізатора при обробці складних функцій, а отримані графіки підтверджують високу точність дискретизації без спотворень сигналу.

Розроблене програмне забезпечення повністю відповідає вимогам технічного завдання, забезпечуючи автоматизацію повного циклу моделювання кіберфізичних систем: від введення формули до візуалізації результатів з використанням відкритих технологій GNU.

4 ВИВЕДЕННЯ ЦІЛЬОВОЇ ФУНКЦІЇ. ОБЧИСЛЕННЯ ПАРАМЕТРІВ МОДЕЛІ СИСТЕМИ

В розробленій системі комп'ютерного моделювання є набір параметрів, які являють собою вихідні дані програми .

$$\mathrm{v_k}, \mathrm{k} = 1,2,..\mathrm{N_v}$$
$$0 \le \mathrm{v_k} \le 1,$$

де, $v_k -$ вихідні дані програми;

$\mathrm{N_v}$ – завдання користувача дорівнює 2, бо у користувача наявно 2 завдання.

Та набір параметрів, що являє собою налаштування програми.

$$u_k, k = 1,2 \dots N_U,$$

де u_k – налаштування програми;

$\mathrm{N_U}$ – параметри управління дорівнює 9, бо наявно 9 параметрів управління.

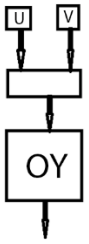

Рисунок 4.1 – Схематична модель програми як об'єкта автоматизації

Формула 4.1 – Модель програми як об'єкта автоматизації

$$\mathrm{x} = \sum_{\mathrm{k}=1}^{\mathrm{N_v}} \mathrm{a_k v_k} + \sum_{\mathrm{k}=1}^{\mathrm{N_u}} \mathrm{b_k u_k} = \sum_{\mathrm{k}=1}^{2} \mathrm{a_k v_k} + \sum_{\mathrm{k}=1}^{9} \mathrm{b_k u_k} \qquad (4.1)$$

де, $\mathrm{a_k}, \mathrm{b_k}$ –параметри моделі, x – час виконання завдання програми

Основна ідея полягає в підборі параметрів u_k таким чином, щоб час виконання завдання час (x) досягав заданого значення. Для цього необхідно визначити цільову функцію і умови задоволення задачі оптимізації.

Для побудови математичної моделі програми як об'єкта автоматизації використовуємо статичну ідентифікацію на основі даних тестувань програми

$$x^{(j)} \begin{cases} V_k^{(j)}\ k = 1,2,\dots N_v \\ U_k^{(j)} k = 1,2 \dots, N_u, \\ \quad j = 1,2,\dots n \end{cases} \tag{4.2}$$

$$x^{(j)} = \sum_{k=1}^{2} a_k v_k{}^{(j)} + \sum_{k=1}^{9} b_k u_k{}^{9}, \tag{4.3}$$

де n – число випробувань;

$V_k^{(j)}$ k, $U_k^{(j)}$k – умови випробування j;

$x^{(j)}$ – час випробування j.

Для знаходження параметрів моделі використаємо метод найменших квадратів

$$s = \sum_{j=1}^{n} (\sum_{n=1}^{9} a_k\, u_k^j + \sum_{k=1}^{2} b_k v_k^j - x^j)^2, \tag{4.3}$$

$$\frac{ds}{da_i} = 0, i\ = 1,2\dots 9, \tag{4.4}$$

$$\frac{ds}{db_i} = 0, i\ = 1,2, \tag{4.5}$$

$$\frac{ds}{da_i}=\sum_{j=1}^{n}(\sum_{i=1}^{n_u} a_i\, u_i^j + +\sum_{i=1}^{nv} b_i v_i^{(j)} - x^j) * u_i^j, i=1,2,\ldots N_u, \qquad (4.6)$$

$$\frac{ds}{db_i}=\sum_{j=1}^{n}(\sum_{i=1}^{n_u} a_i\, u_i^j + +\sum_{i=1}^{nv} b_i v_i^{(j)} - x^j) * v_i^j, i=1,2,\ldots N_v, \qquad (4.7)$$

$$\frac{d_s}{d_{ak}}=\sum_{k=1}^{nu}(\sum_{j=1}^{n} u_i^j \cdot u_k^j)\, a_k + \sum_{k=1}^{Nv}(\sum_{j=1}^{n} u_i^j V_k^j) b_k - \sum_{j=1}^{n} u_i^j x^j, \qquad (4.8)$$

$$i=1, 2..N_u$$

$$\frac{ds}{db_i}=\sum_{k=1}^{Nu}(\sum_{j=1}^{n} v_i^j * u_k^j)a_k + \sum_{k=1}^{n_v}\left(\sum_{j=1}^{n} v_i^j v_k^j\right) b_k - \sum_{j=1}^{n} v_i^j x^j, \qquad (4.9)$$

$$i=1, 2\ldots Nv$$

Введемо умовні позначення:

$$A_{ik}=\sum_{j=1}^{n} u_i^j u_k^j, \qquad (4.10)$$

$$; B_{ik}=\sum_{j=1}^{n} v_i^j v_k^j, \qquad (4.11)$$

$$; C_{ik}=\sum_{j=1}^{n} u_i^j v_k^j, \qquad (4.12)$$

$$f_i^u=\sum_{j=1}^{n} u_i^j x^j , \qquad (4.13)$$

$$f_i^v=\sum_{j=1}^{n} v_i^j x^j . \qquad (4.14)$$

З урахуванням позначень (4.8) та (4.9) отримані рівняння (4.4) та (4.5) дорівнюють нулю. Таким чином отримуємо систему рівнянь

$$\sum_{k=1}^{9} A_{ik}a_k + \sum_{k=1}^{2} c_{ik}b_k = f_i^u;\ i = 1,2\ldots9,$$

$$\sum_{k=1}^{9} C_{ik}a_k + \sum_{n=1}^{2} B_{ik}b_k = f_i^v;\ i=1,2. \tag{4.15}$$

Дану систему лінійних рівнянь необхідно вирішити методом Гауса.

$$\begin{pmatrix} A & C_u \\ C_v & B \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} f_u \\ f_v \end{pmatrix}.$$

Заповнюємо кожну матрицю для розв’язання.

A=

2	-1	0	0	0	0	0	0	0
-1	2	-1	0	0	0	0	0	0
0	-1	2	-1	0	0	0	0	0
0	0	-1	2	-1	0	0	0	0
0	0	0	-1	2	-1	0	0	0
0	0	0	0	-1	2	-1	0	0
0	0	0	0	0	-1	2	-1	0
0	0	0	0	0	0	-1	2	-1
0	0	0	0	0	0	0	-1	2

C_u=

1	*0*
0	*1*
1	*0*
0	*1*
1	*0*
0	*1*
1	*0*
0	*1*
1	*0*

C_v=

1	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*
0	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*

B=

2	-1
-1	2

f_u =

1
0
1
0
1
0
1
0
1

f_v =

1
0

Для формування розширеної матриці об’єднуємо всі матриці у одну розширену матрицю:

M=

2	-1	0	0	0	0	0	0	0	1	0
-1	2	-1	0	0	0	0	0	0	0	1
0	-1	2	-1	0	0	0	0	0	1	0
0	0	-1	2	-1	0	0	0	0	0	1
0	0	0	-1	2	-1	0	0	0	1	0
0	0	0	0	-1	2	-1	0	0	0	1
0	0	0	0	0	-1	2	-1	0	1	0
0	0	0	0	0	0	-1	2	-1	0	1
0	0	0	0	0	0	0	-1	2	1	0
1	0	1	0	1	0	1	0	1	2	-1
0	1	0	1	0	1	0	1	0	-1	2

І вектор правих частин f:

f =

1
0
1
0
1
0
1
0
1
1
0

Розв’язання системи методом Гауса.

1. Прямий хід (елімінація Гауса).

Використовуємо часткове вибирання головного елемента для уникнення числової нестабільності.

2. Зворотний хід (підстановка).

Код для рішення систем рівнянь, написаний мовою Python наведено в додатку Г.

Розв’язуємо систему для отримання векторів a, b.

Виведення результатів.

Отже, було отримано результати, зображені на рисунку 4.2.

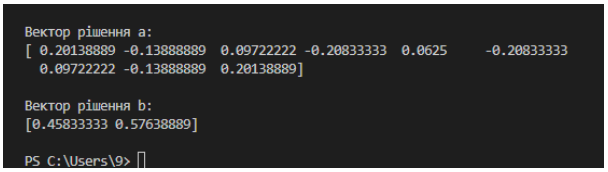

Рисунок 4.2 – Результат обчислення параметрів a_k та b_k

a =

0.20138889
−0.13888889
0.09722222
−0.20833333
0.0625
-0.2083333
0.9722222
-0.13888889
0.20138889

b=

0.45833333
0.57638889

Тепер, коли ми маємо масив a_k та b_k елементів – можна шукати параметри для управління u_k через v_k , щоб час виконання завдання x дорівнював x-цільовому.

Цільова функція матиме наступний вигляд:

$$f(u) = |x_{target} - (\sum_{k=1}^{N_v} a_k v_k + \sum_{k=1}^{N_u} b_k u_k)|.$$

5 ОХОРОНА ПРАЦІ

Інформаційні технології є невід'ємною частиною нашого повсякденного життя. Правила роботи з ними впроваджуються в наше життя завдяки збільшення ергономічності систем з точки зору користувацького досвіду як наслідку постійного розвитку технологій.

Адміністратори – відповідальні та дуже зосереджені на вирішенні проблем люди. Навіть під час вільного часу від роботи такі люди зазвичай думають про вирішення проблеми підприємства.

У ролі адміністратора на підприємстві, особливо при роботі з комп'ютерною системою моделювання, надзвичайно важливо не лише забезпечувати ефективну роботу систем та підлеглих, але й правильно організовувати власний робочий час та відпочинок. Постійне перебування у стані мозкового штурму може призвести до вигорання, зниження продуктивності та негативно вплинути на загальну ефективність роботи.

5.1 Важливість організації робочого часу

Планування завдань:

– планування робочого дня дозволяє краще розподілити час та ресурси, уникнути перевантаження та зменшити стрес. Важливо визначити пріоритетні завдання і розподілити час на їх виконання;

– тайм-менеджмент: використання методів тайм-менеджменту, таких як техніка Pomodoro (робота інтервалами по 25 хвилин з короткими перервами) або метод ABC (розподіл завдань за важливістю та терміновістю), допомагає підвищити продуктивність та зосередженість;

– делегування обов'язків: розподіл завдань між підлеглими не лише знижує навантаження на адміністратора, але й сприяє розвитку команди. Важливо навчитися довіряти підлеглим та ефективно розподіляти обов'язки;

– регулярні перерви: перерви під час роботи допомагають зняти напругу та відновити працездатність. Рекомендується робити короткі перерви кожні 60 хвилин, під час яких можна виконати прості фізичні вправи або пройтися.

При роботі з системою моделювання важливо забезпечити відповідність умов праці державним стандартам України, зокрема ДСТУ ISO 45001:2019 «Системи менеджменту охорони здоров'я та безпеки праці» [38].

Основні вимоги охорони праці включають забезпечення безпечних умов роботи на робочому місці, організацію належного освітлення, вентиляції, ергономіки робочих місць та мінімізації шкідливих факторів, пов'язаних з тривалим перебуванням за комп'ютером.

5.2 Аналіз небезпек і шкідливих факторів

Основні небезпеки та шкідливі фактори при роботі з автоматизованою комп'ютерною системою моделювання включають:

– фізичні фактори: тривале сидіння за комп'ютером, неправильна постава, неадекватне освітлення робочого місця;

– психофізіологічні фактори: стрес, пов'язаний з великою кількістю інформації та швидкими термінами виконання завдань;

– електромагнітне випромінювання: випромінювання від комп'ютерної техніки, яке може впливати на здоров'я працівників;

– втома очей: навантаження на зір через тривале використання моніторів.

5.3 Заходи з охорони праці

Для забезпечення безпеки працівників необхідно впровадити наступні заходи з охорони праці:

– організація робочого місця: відповідно до ДСанПіН 3.3.2-007-98 [39], робоче місце повинно бути організоване з урахуванням ергономіки, забезпечено регульованими меблями та правильним розташуванням обладнання;

– освітлення: відповідно до ДБН В.2.5-28-2018 [40], освітлення робочого місця повинно бути не менше 300 люкс, щоб зменшити напруження очей;

– вентиляція: відповідно до ДБН В.2.5-67:2013 [41], необхідно забезпечити належну вентиляцію приміщень для підтримки свіжого повітря;

– перерви та вправи: введення регулярних перерв кожні 60 хвилин для виконання вправ для очей та спини;

– навчання: проведення регулярних інструктажів з охорони праці та навчання працівників правилам безпечного використання комп'ютерної техніки.

5.4 Пожежна безпека

Для забезпечення пожежної безпеки в приміщеннях, де використовується комп'ютерна система моделювання, необхідно дотримуватись наступних вимог:

– **з**асоби пожежогасіння**:** наявність первинних засобів пожежогасіння (вогнегасників) відповідно до ДСТУ 3675-98 [42];

– **с**истема оповіщення**:** встановлення системи пожежної сигналізації та оповіщення про пожежу відповідно до ДСТУ EN 54-2:2003 [43];

– **е**вакуаційні шляхи**:** забезпечення вільного доступу до евакуаційних шляхів та виходів, які повинні бути позначені та освітлені згідно з ДБН В.1.1-7:2016 [44];

5.5 Висновки до п’ятого розділу

Організація робочого часу та відпочинку є критично важливою на підприємстві. Ефективне планування завдань, використання методів тайм-менеджменту, делегування обов’язків, а також регулярні перерви, фізична активність, релаксаційні техніки, збалансоване харчування та якісний сон допомагають підтримувати високу продуктивність, знижувати стрес та запобігати професійному вигоранню. Забезпечення власного здоров’я та добробуту є ключем до успішного виконання професійних обов’язків та загального процвітання підприємства.

Забезпечення охорони праці при роботі з комп’ютерною системою моделювання є критично важливим для підтримання здоров’я та безпеки працівників.

Дотримання державних стандартів України, таких як ДСТУ ISO 45001:2019, ДСанПіН 3.3.2-007-98, ДБН В.2.5-28-2018, ДБН В.2.5-67:2013, ДСТУ 3675-98 та ДБН В.1.1-7:2016, дозволяє мінімізувати вплив шкідливих факторів, запобігти травмам та аварійним ситуаціям.

Впровадження відповідних заходів охорони праці та пожежної безпеки забезпечить комфортні та безпечні умови роботи для всіх працівників, що сприятиме підвищенню продуктивності та ефективності роботи організації.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи виконано комплексне дослідження та розробка програмного забезпечення для автоматизації процесу комп'ютерного моделювання кіберфізичних систем з використанням відкритих програмних технологій та колекції компіляторів GNU.

Проведена робота охоплює повний цикл створення системи автоматизованого проєктування – від аналізу предметної області та недоліків існуючих пропрієтарних рішень, через детальне проєктування гібридної архітектури та алгоритмів трансляції, до практичної реалізації десктопного застосунку та експериментальної верифікації отриманих результатів.

Актуальність дослідження обумовлена необхідністю забезпечення технологічної незалежності, зниження витрат на ліцензування програмного забезпечення у сфері інжинірингу та освіти, а також потребою у інструментах, що відповідають концепціям Індустрії 5.0.

Обраний шлях поєднання зручного інтерфейсу введення математичних моделей з високопродуктивним обчислювальним ядром дозволив створити доступну альтернативу комерційним пакетам для задач динамічного моделювання.

В першому розділі проведено аналіз предметної області та поставлено задачі до виконання. Порівняння існуючих інструментальних засобів на прикладі MATLAB/Simulink та ANSYS показало, що їхній вартісний профіль та закритість програмного коду є бар'єром для широкого впровадження у наукових дослідженнях та стартапах, тоді як існуючі відкриті рішення часто мають високий поріг входження через складність ручного кодування.

Критичним результатом аналізу стало виявлення можливості застосування підходу модельно-орієнтованого проєктування на базі відкритих технологій GNU GFortran, Python, C# та JS MathLive, що відкриває шлях до створення систем моделювання, що забезпечують швидкість виконання коду на рівні низькорівневих мов при збереженні зручності високорівневого опису моделей.

В другому розділі розроблено повну специфікацію системи, що включає в себе: функціональні вимоги, гібридну архітектуру програмного комплексу та алгоритми обробки даних.

Розроблений алгоритм автоматичної трансляції базується на побудові та аналізі абстрактних синтаксичних дерев AST, що дозволяє конвертувати математичні вирази у форматі розмітки LaTeX безпосередньо у валідний програмний код. Запропонована структура файлів даних формату .UCH та .OUT забезпечує ефективну взаємодію між різнорідними модулями системи та гарантує відтворюваність експериментів.

В третьому розділі було виконано програмну імплементацію системи та проведено чисельні експерименти на типових задачах динаміки.

Незважаючи на складність організації взаємодії між керованим кодом C# та нативним кодом Fortran, завдяки використанню Python було отримано показники, що підтверджують ефективність рішення. Таким чином експериментальна перевірка на моделях затухаючих коливань та биття частот підтвердила високу точність дискретизації та відсутність спотворень сигналу.

Завдяки використанню компілятора GFortran з прапорами оптимізації -O3, час виконання розрахунків для 1000–5000 ітерацій складає частки секунди, що значно перевищує показники інтерпретованих мов.

Не менш важливим є те, що система продемонструвала стабільну роботу при обробці складних трансцендентних функцій та вкладених операцій, забезпечуючи повну відповідність візуалізованих графіків аналітичним розв’язкам.

Наукова новизна роботи полягає у розробці унікального методу автоматизованої генерації високопродуктивного коду для моделювання КФС, який поєднує символьні обчислення Python і бібліотеку SymPy з чисельною потужністю Fortran у прозорому для користувача режимі.

Запропонована архітектурна взаємодія показала, що бар’єр складності низькорівневого програмування може бути знятий завдяки автоматичній трансляції AST-дерев.

Практична цінність рішення полягає у створенні інструментарію, який дозволяє інженерам зосередитися на фізичній суті процесів, делегуючи рутинні задачі кодування та компіляції програмному комплексу.

Водночас робота фіксує межі виконаної реалізації: тестування проводилося на задачах, що описуються системами диференціальних та алгебраїчних рівнянь, без підключення апаратних контролерів у контурі (Hardware-in-the-Loop). Не розглядалися сценарії розподілених обчислень на кластерах.

Ці аспекти визначають логічні напрями продовження: впровадження підтримки зовнішніх інтерфейсів для підключення реальних датчиків, розширення бібліотеки чисельних методів інтегрування, а також розробки веб-версії для хмарного моделювання.

Загалом мета роботи досягнута, адже було спроєктовано, реалізовано і експериментально підтверджено працездатність програмного забезпечення для комп’ютерного моделювання КФС з використанням колекції компіляторів GNU.

Отримані результати підтверджують, що використання відкритого стеку технологій дозволяє створити ефективне середовище моделювання, яке не поступається комерційним аналогам у швидкодії розрахунків та забезпечує технологічну незалежність користувача.

Запропоноване рішення є готовим фундаментом для подальшого розвитку вітчизняних засобів інженерного аналізу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки з підготовки кваліфікаційної роботи магістра спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» / Упоряд.: І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, С.П. Новоселов, О.В Сичова. Харків: ХНУРЕ, 2025. – 55 с.

2. Положення про організацію освітнього процесу у ХНУРЕ [електронний ресурс]: Наказ ХНУРЕ від 27 листопада 2020 р. No 400. – Режим доступу: https://nure.ua/wp- content / uploads / Main _ Docs _ NURE/polozhennja-proorganiza-ciju-osvitnogo-procesu-v-hnure.pdf.

3. ДСТУ 3008: 2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. К.: ДП "УкрНДНЦ". 2016. 30 с.

4. CyberPhysical Systems (CPS). NSF - U.S. National Science Foundation. URL: https://www.nsf.gov/funding/opportunities/cps-cyberphysical-systems/503286/nsf24-581 (дата звернення: 28.12.2025).

5. Modeling Cyber-Physical Systems - MATLAB & Simulink. MathWorks - MATLAB and Simulink Conferences. URL: https://ch.mathworks.com/help/simevents/ug/modeling-cyber-physical-systems.html (дата звернення: 28.12.2025).

6. MathWorks - Solutions. MathWorks - Maker of MATLAB and Simulink. URL: https://www.mathworks.com/solutions.html (дата звернення: 28.12.2025).

7. Simulink Documentation. MathWorks - Maker of MATLAB and Simulink. URL: https://www.mathworks.com/help/simulink/index.html (date of access: 28.12.2025).

8. User Documentation. Introduction. URL: https://openmodelica.org/useresresources/userdocumentation/ (дата звернення: 28.12.2025).

9. GitHub - modelica/ModelicaStandardLibrary: Free (standard conforming) library to model mechanical (1D/3D), electrical (analog, digital, machines), magnetic, thermal, fluid, control systems and hierarchical state machines. Also numerical

functions and functions for strings, files and streams are included. GitHub. URL: https://github.com/modelica/ModelicaStandardLibrary (date of access: 28.12.2025).

10. URL: https://www.ansys.com/advantage-magazine (дата звернення: 28.12.2025).

11. A Security Analysis of Cyber-Physical Systems Architecture for Healthcare. MDPI. URL: https://doi.org/10.3390/computers5040027 (дата звернення: 28.12.2025).

12. GCC, the GNU Compiler Collection- GNU Project. GCC, the GNU Compiler Collection- GNU Project. URL: https://gcc.gnu.org/ (date of access: 28.12.2025).

13. GitHub - pythonnet/pythonnet: Python for .NET is a package that gives Python programmers nearly seamless integration with the .NET Common Language Runtime (CLR) and provides a powerful application scripting tool for .NET developers. GitHub. URL: https://github.com/pythonnet/pythonnet (дата звернення: 28.12.2025).

14. Getting Started - MATLAB. Access Denied. URL: https://ch.mathworks.com/products/matlab/getting-started.html (date of access: 28.12.2025).

15. The US Navy. Wayback Machine. URL: https://web.archive.org/web/20060402215910/https://www.navy.mil/navydata/fact_display.asp?cid=1100&tid=1100&ct=1 (date of access: 28.12.2025).

16. LaTeX Documentation. LaTeX - A document preparation system. URL: https://www.latex-project.org/help/documentation/ (дата звернення: 28.12.2025).

17. L'approche Model-BasedÂ Design avec Simulink - MATLAB & Simulink. https://ch.mathworks.com/help/simulink/gs/model-based-design.html. URL: https://fr.mathworks.com/help/simulink/gs/model-based-design.html (дата звернення: 28.12.2025).

18. CAD Software | 2D and 3D Computer-Aided Design | Autodesk. Access Denied. URL: https://www.autodesk.com/solutions/cad-software (дата звернення: 28.12.2025).

19. Piven N. Free Software as a tool for technological advantage and independence in the digital environment / N. Piven // Матеріали I International Conference «Sustainable smart cities and communities: business and innovation solutions» (Сталі розумні міста та спільноти: бізнес та інноваційні рішення) SSC&C2025, м. Харків, квітень 2025 р. – Харків: ХНУРЕ, 2025. – с 13-15. [Електронний ресурс]. – Режим доступу: https://tapr.nure.ua/wp-content/uploads/2025/04/zbirnik_sscc2025_compressed.pdf (дата звернення: 28.12.2025).

20. Mathfield API Reference · MathLive. Scientific Computing for Everyone · MathLive. URL: https://mathlive.io/mathfield/api/ (date of access: 28.12.2025).

21. Insert Equations into the Live Editor - MATLAB & Simulink. MathWorks - MATLAB and Simulink Conferences. URL: https://ch.mathworks.com/help/matlab/matlab_prog/insert-equations.html (date of access: 28.12.2025).

22. C# - a modern, open-source programming language | .NET. Microsoft. URL: https://dotnet.microsoft.com/en-us/languages/csharp (date of access: 28.12.2025).

23. Get started with WebView2 in WinForms apps - Microsoft Edge Developer documentation. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/microsoft-edge/webview2/get-started/winforms (date of access: 28.12.2025).

24. JSON. JSON. URL: https://www.json.org/json-en.html (дата звернення: 28.12.2025).

25. Introduction · MathLive. Scientific Computing for Everyone · MathLive. URL: https://mathlive.io/mathfield/ (дата звернення: 28.12.2025).

26. Welcome to Python.org. Python.org. URL: https://www.python.org/doc/ (дата звернення: 28.12.2025).

27. GCC, the GNU Compiler Collection- GNU Project. URL: https://gcc.gnu.org/onlinedocs/gcc-15.2.0/gcc.pdf (дата звернення: 28.12.2025).

28. IDEFØ – Function Modeling Method – IDEF. IDEF – Integrated DEFinition Methods (IDEF). URL: https://www.idef.com/idefo-function_modeling_method/ (дата звернення: 28.12.2025).

29. Introduction - Compute Engine · MathLive. Scientific Computing for Everyone · MathLive. URL: https://mathlive.io/compute-engine/ (дата звернення: 28.12.2025).

30. SymPy. URL: https://www.sympy.org/en/index.html (дата звернення: 28.12.2025).

31. Codegen - SymPy 1.14.0 documentation. URL: https://docs.sympy.org/latest/modules/utilities/codegen.html#sympy.utilities.codegen.FCodeGen (дата звернення: 28.12.2025).

32. What is Windows Forms - Windows Forms. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/ (дата звернення: 28.12.2025).

33. HTML: HyperText Markup Language | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 28.12.2025).

34. JavaScript | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 28.12.2025).

35. Calling Python from C#: an introduction to PythonNET. https://.medium.com/.
URL: http://somegenericdev.medium.com/calling-python-from-c-an-introduction-to-pythonnet-c3d45f7d5232 (дата звернення: 30.09.2021).

36. Industry 5.0. Research and innovation. URL: https://research-and-innovation.ec.europa.eu/research-area/industrial-research-and-innovation/industry-50_en (дата звернення: 28.12.2025).

37. What's new in .NET 9. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-9/overview (дата звернення: 28.12.2025).

38. ДСТУ ISO 45001:2019. Системи управління охороною здоров’я та безпекою праці. Вимоги та настанови щодо застосовування (ISO 45001:2018, IDT). – [Чинний від 2021-01-01]. – Київ : ДП «УкрНДНЦ», 2020. - 50 с. – (Національний стандарт України).

39. ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин : затвердж. Постановою Головного державного санітарного лікаря України від 10.12.1998 р. № 7. – Київ : МОЗ України, 1998.

40. ДБН В.2.5-28:2018. Природне і штучне освітлення. – [Чинний від 2019-03-01]. – Київ : Мінрегіон України, 2018. – 135 с. – (Державні будівельні норми України).

41. ДБН В.2.5-67:2013. Опалення, вентиляція та кондиціонування. – [Чинний від 2014–01–01]. – Київ : Мінрегіон України, 2013. – 149 с. – (Державні будівельні норми України).

42. ДСТУ 3675-98. Пожежна техніка. Вогнегасники переносні. Загальні технічні вимоги та методи випробувань. – [Чинний від 1999-01-01]. – Київ : Держстандарт України, 1998. – 26 с. – (Національний стандарт України).

43. ДСТУ EN 54-2:2003. Системи пожежної сигналізації. Частина 2. Прилади приймально-контрольні пожежні (EN 54-2:1997, IDT). – [Чинний від 2004-07-01]. – Київ : Держспоживстандарт України, 2004. – (Національний стандарт України).

44. ДБН В.1.1-7:2016. Пожежна безпека об’єктів будівництва. Загальні вимоги. – [Чинний від 2017-05-01]. – Київ: Мінрегіон України, 2017. – 36 с. – (Державні будівельні норми України).