МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Литвиненко О.Є.

«___»_____2021р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО–КВАЛІФІКАЦІЙНОГО РІВНЯ "БАКАЛАВР"

Тема: Програмний модуль аналізу динамічних властивостей математичних моделей керованих об'єктів

Виконавець: Водолазкін Є.О.

Керівник: Кучеров Д.П.

Нормоконтролер: Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління

Освітньо-кваліфікаційний рівень бакалавр
Спеціалізація 6.050102 "Системне програмування"
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Литвиненко О. Є.
«_____»_____ 2021р.

ЗАВДАННЯ
на виконання дипломної роботи (проекту)
Водолазкіна Євгенія Олександровича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема проекту (роботи): Програмний модуль аналізу динамічних властивостей математичних моделей керованих об'єктів
затверджена наказом ректора від "4" лютого 2021 року №135/ст.

2. Термін виконання проекту (роботи): з 24.05.2021 до 20.06.2021

3. Вихідні дані до проекту (роботи): інформація про математичні моделі, методи аналізу динамічних властивостей, мова програмування високого рівня, *Microsoft Word.*

4. Зміст пояснювальної записки (перелік питань, що належать розробці):
1) доцільність розробки програмного модуля;
2) проектування програмного модуля;
3) опис та перевірка;

5. Перелік обов'язкового графічного матеріалу:
1) структура програмного модуля;
2) діаграма моделі;
3) екранна форма інтерфейсу користувача;
4) розрахунок матриці переходу

6. Календарний план

№ п/п	Етапи виконання дипломного проекту	Термін виконання етапів	Примітка
1	Ознайомлення з постановкою задачі на дипломний проект	17.05.21-18.05.21	
2	вивчення спеціальної літератури і технічної документації	19.05.21-20.05.21	
3	Аналіз основних завдань та цілей проектування	21.05.21-22.05.21	
4	Підготування розділу 1.	23.05.21-24.05.21	
5	Проаналізувати сучасні методи і алгоритми реалізації програмних засобів	26.05.21-27.05.21	
6	Підготувати розділ 2.	27.05.21-28.05.21	
7	Провести моделювання програмного модулю, оцінити його ефективність	29.05.21-30.05.21	
8	Підготувати розділ 3	31.05.21-01.06.21	
9	Оформити пояснювальну записку та пройти нормоконтроль	01.06.21-05.06.21	
10	Підготовити графічний демонстраційний матеріал	06.06.21-12.06.21	

7. Дата видачі завдання «_17_» травня 2021 р.

Керівник дипломного проекту _____ Кучеров Д.П.
(підпис) (П.І.Б.)

Завдання прийняв до виконання _____ Водолазкін Є.О.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Програмний модуль для аналізу динамічних властивостей математичних моделей керованих об’єктів: 50с., 22 рис., 15 літературних джерел., 3 додатки.

Ключові слова: ПРОГРАМНИЙ МОДУЛЬ, МОДЕЛІ, АНАЛІЗ, АЛГОРИТМ МАТРИЦІ ПЕРЕХОДУ, *WINDOWSFORMS, C#.*

Об'єкт дослідження – алгоритм дослідження моделі для аналізу поведінки об’єкта.

Предмет дослідження – програмний модуль аналізу динамічних властивостей математичних моделей керованих об’єктів.

Мета дипломного проекту – реалізувати додаток, що описує динамічні властивості за заданими користувачем параметрами.

Прогнозні припущення щодо подальшого розвитку предмету дослідження – можливе доопрацювання у виведенні результатів в 3д графіці. Додавання нових моделей та алгоритмів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

СУБД - система управління базами даних

САПР – система автоматизованого проектування та розрахунку

WPF – *Windows Presentation Foundation*(графічна підсистема)

COM – розширення файлу

API – прикладний програмний інтерфейс

SADT – структурована методика аналізу та проектування

ERD – модель «сутність-зв’язок»

DFD – діаграма потоків даних

ВСТУП

Актуальність теми. Моделювання в широкому розумінні - це особливий когнітивний процес, теоретичний та практичний непрямий когнітивний метод. Коли суб’єкт замість безпосереднього об’єкта пізнання вибирає або створює подібний допоміжний замінник об’єкта (моделі), інформація про реальні навчальні теми при проведенні досліджень та передача отриманих результатів.

Однією з характеристик цього методу є те, що він може відтворювати модель відповідно до деяких основних атрибутів, структури, взаємозв'язку між елементами та цілі дослідження взаємозв'язку об'єкта дослідження. У процесі пізнання модель слідує за об’єктом і є своєрідною його копією; тоді як у процесі відтворення та побудови об’єкт слідує за моделлю та копіює її.

Математичне моделювання-моделювання, де модель - це система математичних взаємозв’язків, що описують певні технологічні, економічні чи інші процеси.

На сьогоднішній день багато обсягів інформації необхідні у деталізації та розумінні. Особисто у сфері ІТ. Одним із напрямків розвитку є моделювання.

Комп'ютерні моделі стали загальним інструментом математичного моделювання і використовуються у різних галузях, таких як медицина, машинобудування, проектування та інших. Такі моделі використовуються для отримання нових знань про об’єкти або для наближення поведінки систем, занадто складних для аналізу та дослідження. Масштаб подій, що моделюються за допомогою комп'ютерного моделювання, набагато перевищує всі можливості, яких можна досягти за допомогою традиційних ручок та паперу.

Воно є дуже ефективним як для студентів оскільки можуть допомогти зрозуміти та дослідити рівняння або функціональні методи, так і для більшості сфер людської діяльності, адже моделювання та дослідження моделей є компонентом остаточної повної моделі системи або задачі яку необхідно розв’язати.

Зараз є такі комп’ютерні гіганти у сфері математичного моделювання як *MatLab* або *Maple* з широким спектром функціоналу та можливостей. Є і більш

дешеві та прості пакети які дають змогу робити необхідні операції, але наприклад робота з *MatLab* є простою і водночас складною. Маючи величезний набір інструментів та бібліотек , проте їх використання як і програмування на мові *MatLab* є дещо заплутаним і потребує довгого навчання для якісної роботи.

Як конкуренція цим пакетам створити проект звісно неможливо на даний момент, але має місце реалізування подібного продукту для більш систематизованого напрямку у моделюванні. Так, такий програмний модуль буде досить примітивним, в порівнянні з іншими, але дуже простим у використанні та цілеспрямованим у необхідній галузі.

Тому саме такі програми як дана мають місце у навчанні або експлуатації для малої кількості фахівців певної фірми або закладу. Метою є якраз розроблення базового модуля для аналізу властивостей об’єктів під час впливу на параметри системи користувачем. Необхідно реалізувати зручний інтерфейс з можливістю відображення схеми або моделі елементу об’єкта чи самого об’єкта для розуміння, відповідно список параметрів та вивід точних результатів. Результати вважаються будуть графічними.

Щодо галузі застосування то це можуть бути школярі або студенти, що навчаються за даним напрямком, наприклад факультет аеронавігації чи телекомунікацій, або механіко-математичному.

Об'єкт дослідження – алгоритм дослідження моделі для аналізу поведінки об’єкта

Предмет дослідження – програмний модуль аналізу динамічних властивостей математичних моделей керованих об’єктів

Мета виконання дипломного проекту – реалізувати додаток, що описує динамічні властивості за заданими користувачем параметрами.

Практичне значення та наукова новизна отриманих результатів – В результаті отримуємо програмний модуль, що забезпечить аналіз моделей у звуженому напрямку для людей, що працюють у цій області.

РОЗДІЛ 1
ДОЦІЛЬНІСТЬ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ

1.1. Математичне моделювання і його властивості

Поки що математичне моделювання є одним з основних методів дослідження. Суть цього методу полягає у розкритті атрибутів та функцій об’єкта, його сутності та процесу за допомогою математичної моделі об’єкта.

Поняття моделі та самого моделювання має безліч визначень, але в більшості випадків ми можемо описати їх наступним чином: [1] Моделі (від латинських коефіцієнтів-метрик, зразків, норм) створюються замінниками для використання в За певних умов відтворюють основні властивості та налаштування вихідного об’єкта. Моделі можуть бути представлені фізичними об’єктами, подібними до оригіналу, або вони можуть бути представлені математичними формулами, описом об’єктів у вигляді тексту або комп’ютерних програм.

Моделювання передбачає створення, дослідження та використання об’єктних моделей. Методи моделювання широко використовуються в різних сферах людської діяльності, особливо у сферах проектування та управління, де основним процесом є прийняття ефективних рішень на основі отриманої інформації.

Метою моделювання є отримання, аналіз та використання інформації про об'єкти, що взаємодіють між собою та із зовнішнім середовищем; модель тут є засобом для поняття властивостей та закономірностей поведінки об'єкта, що продемонстровано на рис. 1.1.

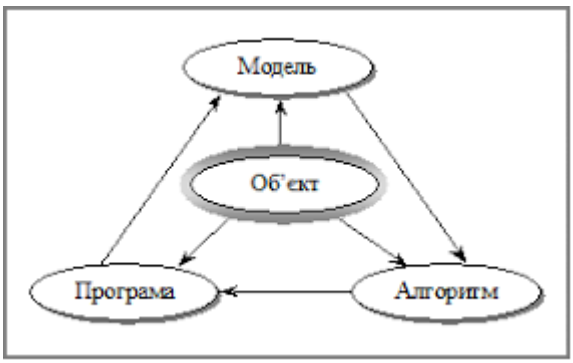

Рис. 1.1. Принцип моделювання

Моделі також можна класифікувати відповідно до математичних методів, що використовуються для вирішення проблеми [2]. Є моделі:

– Лінійна або нелінійна модель [3];

– Централізована або розподілена система [4];

– визначеність чи випадковість [5];

– Статичний або динамічний.

Відповідно до методу, це буде модель. Можливі також змішані типи. Для цього ви також можете додати атрибути моделі [6]. Головна вимогами, що пред'являються до математичних моделей, є вимоги адекватності, універсальності і економічності, що відображені на малюнку рис. 1.2.

Рис. 1.2. Властивості моделі

Достатність. Якщо модель може відображати зазначені атрибути з прийнятною точністю, модель вважається достатньою.

Точність визначається як ступінь збігу початкових значень параметрів моделі та об'єкта. За різних умов експлуатації об’єкта точність моделі різна.

Універсальність. В основному це визначається кількістю та змістом зовнішніх та початкових параметрів, що розглядаються в моделі. економічний. Ця модель характеризується витратами обчислювальних ресурсів, необхідних для реалізації моделі-машини часу і вартості пам'яті. простий. Модель, яка враховує меншу кількість факторів, може отримати очікувані результати одночасно і з однаковою точністю.

Потенціал. За допомогою моделей можна отримати нові знання про предмети. Можливість розробляти моделі без принципових змін. За формою представлення моделей можна виділити наступні:

- використовувати інваріант традиційною математичною мовою для запису взаємозв'язку моделі без розгляду методу розв'язання рівняння моделі;
- аналітичне - початкове рівняння моделі використовується як результат аналітичного рішення для запису моделі;
- Алгоритм - записати взаємозв'язок між моделлю та обраним методом чисельного рішення у вигляді алгоритму;
- Схематична (графічна) - виражає модель графічною мовою (наприклад, графічною мовою, еквівалентною ціновою схемою, схемами тощо);
- Фізичне представлення моделі як скороченої копії реального обладнання та процесу;
- Імітаційні моделі, засновані на подібності явищ, мають різні фізичні властивості, але описуються тими самими математичними рівняннями.

Що стосується моделювання у нашому випадку, то ми розглядаємо такий об’єкт керування як БПЛА. У такій системі відповідно об’єктом управління буде двигун. Динамічні властивості такого елементу будуть представлятися математичною моделлю частини всієї системи і буде характеризуватися нашим

алгоритмом. Двигун буде представлений у вигляді елементарної динамічної ланки, а саме аперіодичною ланкою, що описується рівнянням першого порядку. В результаті отримаємо реакцію на вплив, тобто перехідну функцію при початково заданих умовах.

Досліджуючи саме тип моделювання можна виділити різні види:

– Матеріальне (фізичне) моделювання називається моделюванням. У цьому моделюванні реальний об'єкт порівнюється з його збільшеною або зменшеною копією, а теорія подібності використовується для передачі дослідницьких характеристик об'єкта.
– Ідеальним моделюванням є моделювання, яке порівнює реальні об’єкти з їх описом у вигляді мови, графіки, таблиць та математичних виразів.
– Імітаційне моделювання засноване на аналогії процесів і явищ з різними фізичними властивостями, але всі вони мають однаковий опис за формою.
– Інтуїтивне моделювання, засноване на інтуїтивному мисленні досліджуваного об'єктного моделювання, не може бути формалізованим або непотрібним.
– Наукове моделювання - це завжди логічне моделювання, яке в якості гіпотези використовує найменш прийняте припущення, засноване на спостереженні за імітованим об’єктом.

1.2. Прикладні програми для математичного моделювання

Додаток - це програма, призначена для виконання певних завдань користувача і призначена для безпосередньої взаємодії з користувачем [7]. У більшості операційних систем програми не можуть безпосередньо отримувати доступ до комп'ютерних ресурсів, але можуть взаємодіяти з апаратним забезпеченням та іншими. Використовуйте операційну систему. Комунальні послуги також надаються простою мовою.

Прикладне програмне забезпечення - це програма, яку користувачі можуть використовувати для вирішення своїх інформаційних завдань, не вдаючись до

програмування. Серед них є додатки загального та спеціального призначення. Майже кожному користувачеві потрібен універсальний додаток. Поширені програми включають: текстові та графічні редактори, за допомогою яких можна готувати різні тексти, створювати малюнки, будувати креслення, схеми:

- Система управління базами даних (СУБД), яка дозволяє перетворити ваш комп'ютер на путівник з будь-якої теми;
- Електронна форма, що дозволяє впорядковувати розрахунки, які дуже поширені на практиці;
- Комунікаційна (мережева) програма, призначена для обміну інформацією з іншими комп’ютерами та об’єднання даних у комп’ютерну мережу;
- Програми, що дозволяють створювати комп’ютерні презентації, які використовуються в рекламних цілях, навчанні, використанні з лекціями та доповідями тощо.

Сучасний процес математичного моделювання важко реалізувати без комп'ютерів або вбудованих контролерів [8]. Для цього використовують різноманітні інструментальні програмні засоби та середовища (*MathCad, MatLab, Mathematica, Maple, Derive, VisSim, Genius* й інші), що суттєво спрощують моделювання.

1.2.1. *MathCad*

Поява комп’ютерів не спростило математичних задач, а лише значно покращило швидкість їх виконання та складність вирішення задач. Перш ніж розпочинати такі обчислення, користувачі ПК повинні вивчити комп'ютери, мови програмування та складні методи обчислення та використовувати їх для їх вирішення. Система інтеграції та автоматичного програмного забезпечення, подібна до *MathCAD*[9], широко відома та популярна.

Поки що вони досі є єдиними математичними системами, які використовують загальновживані математичні формули та символи для опису математичних задач. Результат сторінки має однаковий вигляд. Тому система

MathCAD повністю доводить значення САПР (автоматизованого проектування), що є найскладнішою та вдосконаленою автоматичною системою автоматизованого проектування. Можна сказати, що *MathCAD* - це метод САРР в математиці. Функції програми *MathCAD*:

– Розв'язування диференційних рівнянь;

– Будування різновимірних рівнянь;

– Використовувати грецькі літери як у рівняннях, так і в тексті;

– Використовувати систему САПР.

Система *MathCAD* - це потужний, практичний та інтуїтивно зрозумілий інструмент для опису алгоритмів вирішення математичних задач. Система *MathCAD* дуже гнучка та універсальна і може надати цінну допомогу для вирішення математичних задач.

Текстовий редактор використовується для введення та редагування тексту. Текст є коментарем, і математичні вирази в ньому не будуть реалізовані. Текст може складатися зі слів, математичних символів, виразів і формул, як на рис. 1.3.

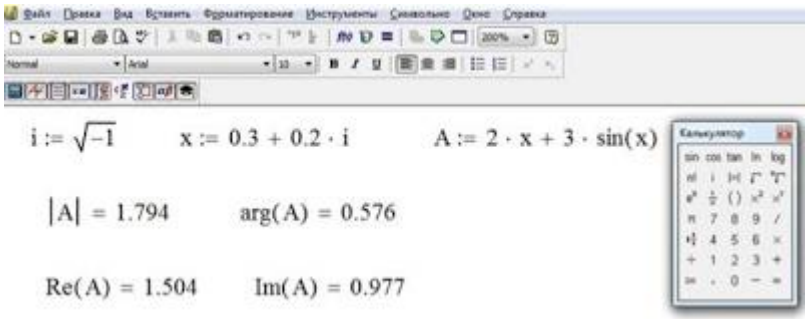

Рис. 1.3. Приклад роботи *MathCad*

Процес формули забезпечує відповідну «багатокрокову» формулу, побудовану із звичайними математичними позначеннями (ділення, множення, квадратний корінь, ціле число, ціле число). Решта версії *Mathcad* повністю підтримує кириличні літери в примітках, формулах та графіці.

Комп’ютери забезпечують функції обчислення за допомогою складних математичних формул.

Зазвичай порівнюють *Mathcad* з *Maple*, *Mathematica*, *MATLAB* та іншими програмними пакетами та відповідними програмними пакетами *MuPAD*, *Scilab*, *Maxima* та іншими. Однак через використання пізніх цілей та пізніх ідеологій об'єктивне порівняння стає все важчим. Функції *Mathematica* орієнтовані на професійних математиків. Ми можемо говорити лише про кленові дерева. *Maple* спочатку був створений для числового зв’язку математичних задач.

Коли ви хочете отримати результати без вивчення математичних задач, він зосереджується на вирішенні задач з прикладної математики, а не теоретичної математики. Однак інші версії наближають обчислювальну потужність *Mathcad* до *Maple*.

Основною відмінністю між *Mathcad* та подібними програмами є графічний спосіб введення виразів, а не текстовий спосіб. Для набору команд, функцій та формул ви можете одночасно використовувати клавіатури та кнопки на багатьох спеціальних панелях інструментів. У будь-якому випадку формула має книжковий вигляд.

1.2.2. *Matlab*

Це програмний пакет для чисельного аналізу та мови програмування, що використовується в програмному пакеті. [10] Система була створена *The MathWorks* і є загальним інструментом для створення користувацьких інтерфейсів із використанням матриць, графічних функцій та графіків як на рис. 1.4. Незважаючи на те, що виріб має знання в багатьох сферах, за допомогою програмного забезпечення *Maple* ви можете використовувати спеціальні інструменти, щоб максимізувати його ефект.

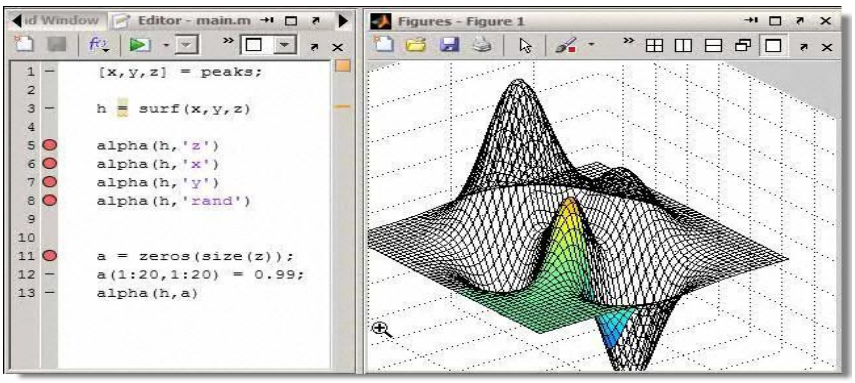

Рис. 1.4. Приклад роботи *MatLab*

MATLAB надає користувачам велику кількість функцій аналізу даних, які охоплюють майже всі галузі математики, особливо [11]:

– Матрична та лінійна алгебра-матрична алгебра, лінійні рівняння, власні значення та вектори, особливості, склад матриці тощо;
– Інтерполяція багаточленів та поліноміальних коренів, функціонування та диференціювання багаточленів, інтерполяція та екстраполяція кривих ...;
– Математична статистика та аналіз даних - статистичні функції; статистична регресія, цифрова фільтрація, швидке перетворення за Файєром тощо;
– Обробка даних - набір спеціальних функцій, включаючи фіксацію, оптимізацію, нульовий пошук, числову інтеграцію тощо;
– Диференціальні рівняння - написати диференціальні та диференціально-алгебраїчні рівняння, диференціальні рівняння із запізненням, рівняння з обмеженнями, рівняння з частковими похідними тощо;
– Розширений матричний спеціальний пакет даних *MATLAB* для спеціальних програм [12].

MATLAB пропонує загальні інструменти алгоритмів для включення алгоритмів, включаючи вдосконалені алгоритми, які використовують об'єктно-орієнтоване поняття програмування. Він має всі інструменти, необхідні для інтегрованого середовища розробки, включаючи налагоджувачі та аналізатори. Можливість обробки всіх типів даних допомагає створювати алгоритми для мікроконтролерів та інших програм, коли це необхідно.

Програмний пакет *MATLAB* містить велику кількість функцій візуалізації графіки, включаючи тривимірний аналіз даних візуалізації та створення анімаційного відео. Вбудоване середовище дозволяє створювати графічний інтерфейс користувача з різними елементами управління (наприклад, кнопками, полями введення тощо).

Програмний пакет *MATLAB* надає доступ до функцій, які дозволяють створювати, маніпулювати та видаляти *COM*-об'єкти (клієнт та сервер). Він також підтримує технологію *ActiveX*. Ці *COM*-об'єкти належать до спеціального *COM*-класу пакету *MATLAB*.

Програмний пакет *MATLAB* надає доступ до функцій, які дозволяють створювати, маніпулювати та видаляти *COM*-об'єкти (клієнт та сервер). Він також підтримує технологію *ActiveX*. Всі *COM*-об'єкти належать до спеціального *COM*-класу пакету *MATLAB*. Ви можете завантажити версію *.NET* із середовища *MATLAB* та використовувати об'єкти *.NET*. У *MATLAB* - це можливість користуватися веб-сервісами. Спеціальна функція створює клас, який містить методи *API* веб-сервісу, що дозволяє отримувати доступ до веб-служб, викликаючи методи класу. Через послідовний інтерфейс *MATLAB* ви можете безпосередньо отримати доступ до периферійних пристроїв, підключених до комп'ютера через послідовний порт (*COM*). Інтерфейс *MATLAB* відноситься до загальнодоступної бібліотеки *DLL*, що дозволяє вам викликати функції у звичайній бібліотеці динамічних посилань безпосередньо з *MATLAB*. Ці функції повинні мати інтерфейс *C*.

1.2.3. *Mathematica*

Це алгебраїчна комп’ютерна система від *Wolfram Research*. [13] Він містить багато функцій для аналітичного перетворення та чисельного обчислення. Крім того, програма також підтримує графіку та обробку звуку, включаючи двовимірну та тривимірну графіку як на рис. 1.5.

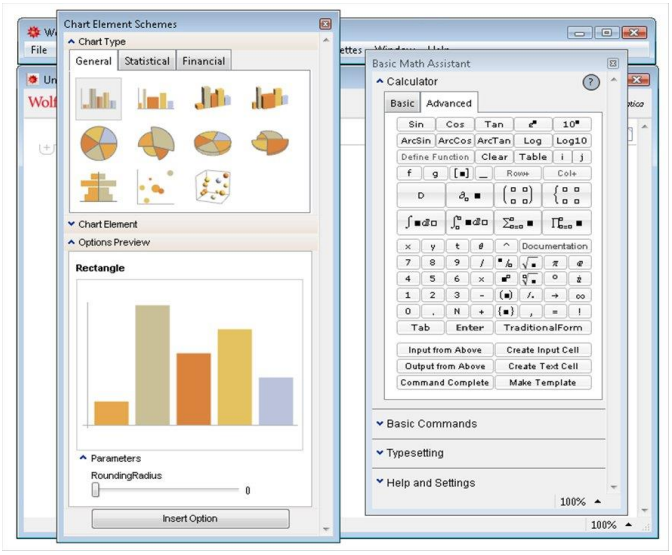

Рис. 1.5. Приклад роботи *Mathematica*

Аналіз перетворень:

– Розуміння систем поліноміальних та тригонометричних рівнянь, нерівності та задачі з попередніми рівняннями;
– Стенограми посилань;
– Зображення;
– Інтеграція та диференціація функцій;
– Знаходження кінцевих та інфінітивних формул;
– Написання диференціальних рівнянь та рівнянь з частковими похідними;
– Перетворення Файп'є та Лапласа та Z-перетворення;
– Перетворення рядів Тейлора, щоб виконувати такі операції як додавання, множення, об'єднання, віднімання обернених функцій тощо.

Можливість цифрового розташування:

– Обчислювати значення функції з довільною точністю, включаючи спеціальні значення;
– Рівняння посилань;
– Виконання інтерполяції поліноміальної функції на основі будь-якої кількості параметрів та відомих наборів значень;
– Перетворення Файп'є та Лапласа та Z-перетворення.

Теорія чисел:

– Визначення чисел за формулами та рядами;
– Дискретний переклад Файп'є;
– Переміщення цифри в основних фактах і знаходження *PNB* та *NSC*.

Лінійна алгебра:

– Операції з матрицями;
– Пошук власних значень та власних векторів.

Графіка та звук:

– Функції малювання, включаючи параметричні криві та криволінійні поверхні;
– Побудова геометричних фігур: ламаної, круглої, прямокутної тощо;
– Відтворювати звук, а його діаграма встановлюється функцією аналізу або точкою набору;
– Імпортувати та експортувати різні пакети даних та векторні формати та аудіо графіку;
– Графічний дизайн та обробка.

Є також функціональні мови програмування, які дуже цікаві. *Mathematica* використовує визначення ": =" за допомогою операторів для додавання розрахунків затримки. [14] Можна сказати, що система *Mathematica* написана на *Mathematica*, хоча деякі функції (особливо ті, що стосуються лінійної алгебри) написані на C для оптимізації.

1.2.4. *Maple*

Один із найпотужніших математичних програмних пакетів. [15] Його функції охоплюють багато галузей математики і можуть використовуватися на різних рівнях, включаючи рівень ретельного вивчення. Він може використовуватися в інтерактивному діалозі, або шляхом написання та коригування програм спеціальною мовою *Maple* (фокус на компіляції).

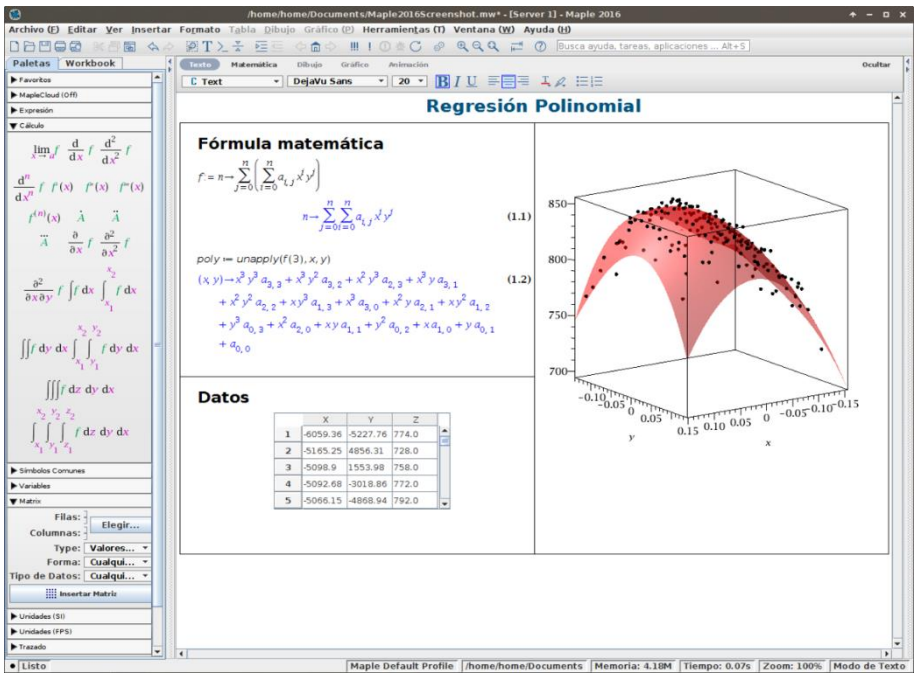

Рис. 1.6. Приклад роботи *Maple*

Основний пакет містить спеціальну символічну програму пісень. Крім того, у пакетах та бібліотеках, завантажених пакетом, зберігаються тисячі спеціальних функцій. Звичайне захоплення символічним перетворенням (комп'ютерна алгебра) для програмних пакетів не означає, що ви можете писати завдання за допомогою *Maple* як це зображено на рис. 1.6.

Вікно *Maple* містить багато атрибутів, знайомих користувачам інших доповнень *Windows*: основна частина, порядок випадаючих меню, панель управління, лінійка тощо. Основну частину основного вікна займає інше вікно, і в нього, як правило, поміщають один або кілька допоміжних документів (аркуші *Maple*). У цьому ж вікні ви можете знайти вікно довідки. Стан випадаючих меню та контекстного меню та кнопок на панелі керування залежить від того, яке вікно активно зберігає та відображає кнопки у вікні робочого аркуша *Maple*.

Введіть виконану команду після спеціального підказки ">" (червоний) і повинен закінчуватися комою (комою або двокрапкою). Спеціальні команди вводяться у послідовному форматі, але користувачі можуть вибрати перехід на спеціальний математичний формат (відсортований за замовчуванням). На лінію можна поставити кілька команд. Інформація про джерело приписується синьому

кольору за замовчуванням. Для початку виконання команди *Maple*, потрібно натиснути клавішу *enter*.

Основні можливості[16]:

– Розв'язання чисельних функцій та алгоритмів;
– Робота з лінійною та булевою алгеброю;
– Компоненти для двовимірної та трьохвимірної графіки;
– Побудова моделей та діаграм;
– Програмування;
– Підтримка коду Матлаб;
– Шаблони та методи вирішення типових задач.

1.2.5. *VisSim*

Це візуальна мова програмування для динамічного моделювання системи та модельного проектування вбудованих мікропроцесорів. *VisSim* має інтуїтивно зрозумілий інтерфейс для *Windows* для створення блок-схем та потужне ядро моделювання. Мовою користується американська компанія *Visual Solutions*, яка перебуває у Вестфорді в Маккаше. Мови та програмне середовище широко використовуються при моделюванні та проектуванні систем управління та цифрової обробки сигналів. Він містить модулі для арифметичних, розширювальних та трансцендентальних функцій, а також цифрові фільтри, передавальні функції, числову інтеграцію та модулі взаємодії. Основними напрямками моделювання є аерокосмічна промисловість, медична сфера, машинобудування та інше.

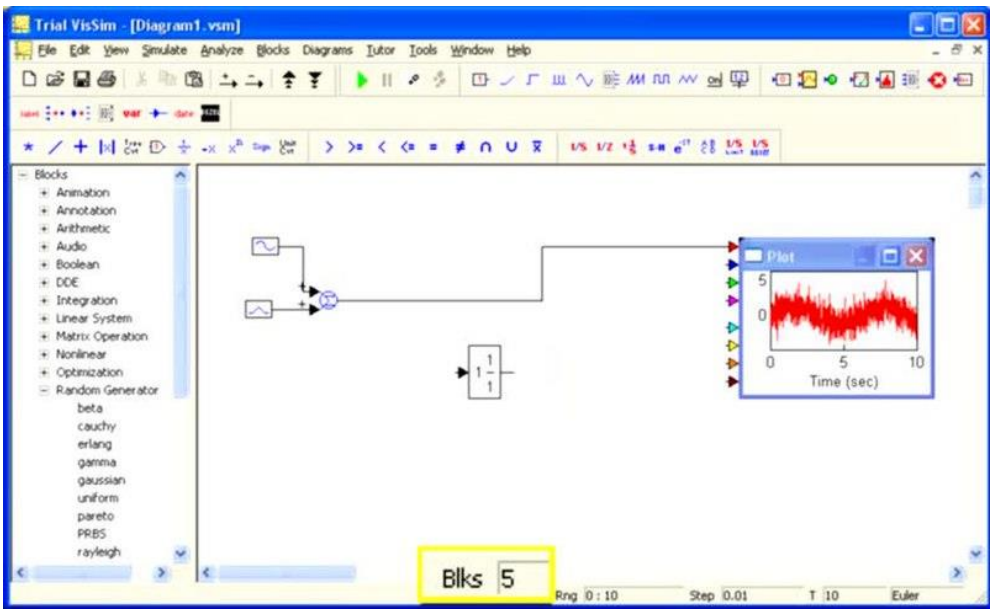

Рис. 1.7. Приклад роботи *VisSim*

VisSim використовується для проектування систем управління та цифрової обробки сигналів для багатодоменного моделювання та проектування. [17] Він включає блоки для біоніки, функцій виступу та трансценденції, а також цифрові фільтри, функції передачі, чисельну інтеграцію та інтерактивні функції як зображено на рис. 1.7.[18] Найчастіше модельовані системи - це авіація, біологічна / медична, цифрова енергія, електродвигуни, електричні, гідравлічні, механічні та технологічні.

1.2.6. *Derive*

Є нащадком *muMATH*, успадкованого *Soft Warehouse* в Гонолулу, Гаваї, що належить *Texas Instruments* (*TI*). [19] Це комп'ютерна система алгебри. Виведення реалізовано в *muLISP [de]* (також сховище програмного забезпечення). Першим випуском стала версія *DOS* у 1988 році. *CAS TI-Nspire* було припинено 29 червня 2007 року. Поточна та остаточна версія - *Derive 6.1* для *Windows*

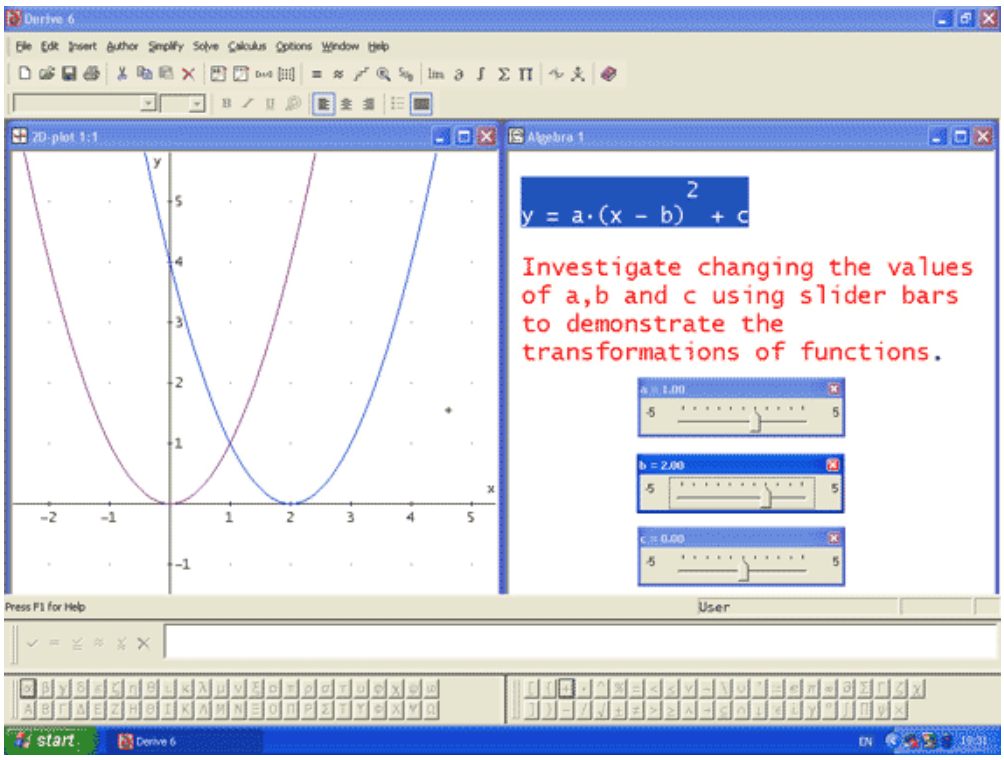

Рис. 1.8. Приклад роботи *Derive*

Коли поточний калькулятор обчислює числа, пакет охоплює алгебраїчні змінні, вирази, рівняння, функції, вектори, матриці та булеві вирази. Також в області арифметики ви можете писати алгебру, тригонометричні функції, числення, лінійну алгебру та позиційне числення одним клацанням миші. За допомогою різних систем координат графіки математичних виразів можуть бути сформовані в два і три вимипа. Завдяки безпечній інтеграції числа, алгебри та графічних можливостей, *Derive* є чудовим інструментом для навчання, викладання та математики, як це відображено на рис. 1.8.

1.3. Аналіз можливостей прикладних програм

Після ознайомлення з вище зазначеними програмами ми можемо робити певні висновки. Щодо *Matlab* то на сьогоднішній день у сфері моделювання це все ж таки лідер. По-перше, це графічні можливості як у 2д так і в 3д, але для всього. Тобто візуалізація можлива як для методів, так і для певних схем, алгоритмів, систем. Додатково хочеться виділити такий вбудований пакет як *Simulink*. Він є

інструментом для моделювання схем з усіх відомих, так і для створення власних компонентів, роботу яких також можна відображати графіками або візуальними моделями. *MathCAD* в свою чергу є менш розширеною програмою, але на мою думку з більш зручним інтерфейсом для математичних обрахувань. І відповідно є більш дешеві, простіші програми як *MuPad* з чудовим графічним наповненням, але звуженим функціоналом. Як результат можна побачити, що кожна програма є більш кращою від іншої за певних умов та поставлених задач.

1.4. Завдання на розроблення програмного модуля

Завданням являється розробка програмного модуля аналізу динамічних властивостей певних математичних моделей деяких керованих об’єктів. Метою завдання є створення цього модуля при дотриманих вимогах та з необхідними результатами на виході.

Основою для створення даного продукту є дослідження та аналіз математичних моделей вузьковизначених об’єктів, оскільки більшість програм для подібних досліджень несуть таку можливість в собі як один з елементів функціонування ПЗ. Дане ПЗ матиме набагато зрозуміліший, простіший функціонал, менший об’єм пам’яті, який займається та конкретний напрямок роботи.

Відповідно за визначені терміни необхідно пройти всі етапи створення проекту:

- Визначення;
- Планування;
- Проектування;
- Реалізація;
- Тестування;
- Завершення.

Проектування реалізовується на операційній системі *Windows* у середовищі розробки *Visual Studio* на мові *C#* з допомоги технології інтелектуальних клієнтів для *NET Framework* під назвою *Windows Forms*.

Створюється програма на такій системі:

– Процесор: *AMD Quad-Core A12-9720P* (2.7 - 3.6 *ГГц*);

– Оперативна пам’ять: 8 *ГБ*;

– Тип оперативної пам’яті: *DDR4* 1866*МГц*;

– Накопичувач: *SSD* 256 *ГБ*;

– Відеокарта: *AMD Radeon* 530, 4 *ГБ*.

1.5. Висновки до розділу

В ході опрацювання даного розділу описано поняття моделювання та його необхідність на сьогоднішній час. Отже проект для аналізу моделей є актуальним. Після аналізу компонентів, що входять у дану область проект має чітке визначення у типі моделювання, а саме роботу з змішаними моделями, які оброблюються певним алгоритмом по відношенню до моделі і має схематичний характер, тобто її опис та результати досліджень будуть відображатися графічно(структурна схема, графік зміни поведінки об’єкту від часу).

При описанні прикладних програм, що існують, можна означити приорітети у розробці аналізатора динамічних властивостей моделей керованих об’єктів. Акцент буде робитися саме на правильність і точність роботи з моделлю, ніж на багатофункціональність та наповненість. Також для правильної роботи необхідно, щоб проект мав певні графічні можливості.

Аналіз дає змогу знайти відношення для розділення роботи над проектом порівну, як на логічну його частину(близько 60% часу):

– Описання зв’язку між значеннями;

– Знаходження перехідних значень;

– Зв’язок між компонентами проекту.

Так і графічну частину, таку як інтерфейс та графічні елементи.

РОЗДІЛ 2
ПРОЕКТУВАННЯ

2.1. Проектування програмного модулю

Розробка програмного забезпечення - це процес концептуалізації вимог до програмного забезпечення для реалізації програмного забезпечення. Розробка програмного забезпечення сприймає вимоги користувача як складні завдання і намагається знайти оптимальне рішення. Поки розробляється програмне забезпечення, намічається план, щоб знайти найкращий можливий дизайн для реалізації передбачуваного рішення.

Існує кілька варіантів дизайну програмного забезпечення. І одним з таких варіантів є структурне проектування. На цьому етапі відбувається саме проектування рішення. Щодо самого проектування, то тут також є декілька шляхів для вирішення даної проблеми. В основу кожного з них входить метод декомпозиції – процес розбиття системи або задачі на підсистеми, під задачі, під функції. Перший принцип полягає у розкладанні проблеми на множину задач, а другий принцип полягає у будуванні ієрархічної структури з частин, методів, процедур.

Основні моделі такі:

– *SADT* (Структурний аналіз і розробка дизайну) ;

– *ERD* (Діаграма відносин і сутностей) ;

– *DFD* (Діаграма даних).

У нашій розробці ми використовуємо методологію *SADT*. Модель є у вигляді графічного блоку, що приймає вхідні та видає вихідні значення. Також блок керується певними умовами та механізмом взаємодії. Модель до нашого програмного модуля буде виглядати так як на рис. 2.1.

Рис. 2.1. Діаграма моделі

Даних підхід був винайдений Дугласом Россом, на основі якої також створена методологія *IDEF0*. Дана методологія являє собою набір правил, методів і процедур для побудови моделі нашої програми. Важливо пам’ятати що блокова частина буде поділена. *SADT* використовує традиційний підхід "зверху вниз" для ієрархічного моделювання функцій. Процес починається із зазначення найвищого спрощеного рівня деталізації на діаграмі верхнього рівня, який потім розкладаються на подальші деталі на кожному кроці, поки не буде досягнутий необхідний рівень деталізації.

Також структуру програми чудово описують вище зазначені *UML* діаграми. Основними є діаграми класів, станів, кооперацій та послідовності.

Діаграма класу - це сукупність визначених компонентів моделі. Частинками діаграми класів можуть бути інтерфейси, пакети, взаємозв'язки, об'єкти та з'єднання як зображено на рис. 2.2.

Клас на малюнку зображений у вигляді прямокутника, який розділений на три частини горизонтальною лінією. Перша частина містить назву класу. Зазвичай назва класу складається з одного і не більше двох слів. Другий містить список атрибутів класу, які описують характеристики таких об’єктів у певній сфері моделі. Третя є списком дій, що відображають її поведінку в моделі предметної області.

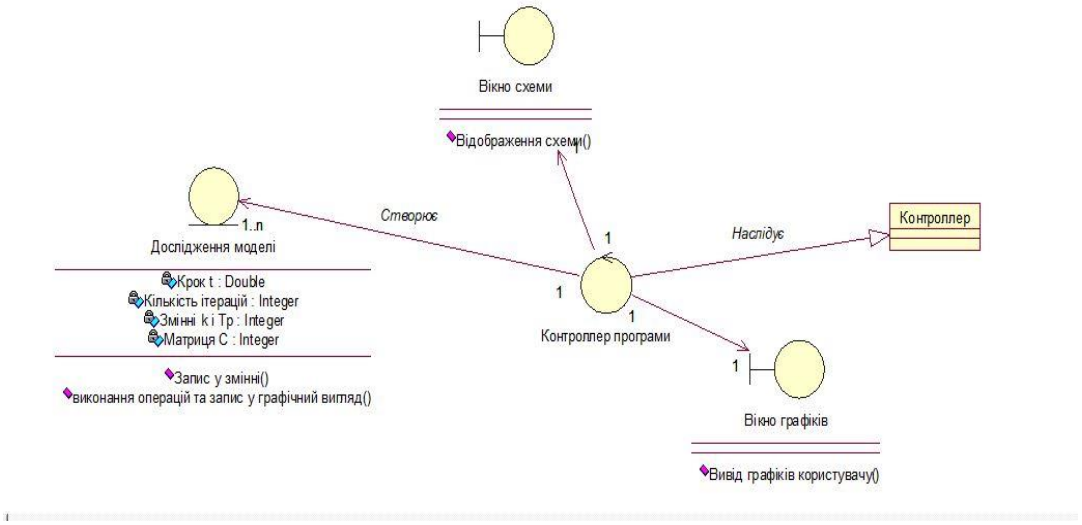

Рис. 2.2. Діаграма класів

Атрибут – це одна з властивостей класів, яка описує набір значень, які може приймати один об’єкт у класі.

Операція - це послуга, що надається кожним екземпляром або об'єктом класу на запит його клієнта. Службою можуть бути інші об'єкти, включаючи екземпляри цього класу.

Інтерфейс - іменований набір операцій, що використовується для опису поведінки окремого елемента в моделі. Інтерфейс у контексті *UML* - це особливий випадок класу, який має операції, але не має атрибутів. Для представлення інтерфейсу використовується спеціальний графічний коло символів або стандартний метод - прямокутник класу з прототипом "інтерфейс".

Асоціації - це зв’язки між об’єктами. Асоціація може мати назву, що вказує на характер взаємозв’язку між об’єктами, а назва може вказувати напрямок, у якому зв’язок читається за допомогою трикутної позначки. Односторонні асоціації можуть бути представлені стрілками. Окрім напрямку, асоціація на діаграмі також вказує на роль, яку кожен клас відіграє у відносинах, і різноманітність, тобто кількість об’єктів, пов’язаних із відносинами. Якщо асоціація групує три або більше категорії разом, вона називається n-масивом і представлена ромбом на перетині прямих. Асоціація з агрегацією - це складний взаємозв'язок між "цілими" класами.

Діаграми стану використовуються для опису поведінки різних систем та підсистем, а також можуть бути використані для моделювання всіх можливих змін стану конкретного об'єкта. Діаграма стану - це особливий тип графіки, що використовується для представлення кінцевого автомата.

Стан на графіку представлений прямокутником із закругленими вершинами. Прямокутник можна розділити на дві частини горизонтальною лінією. Якщо вказана лише одна частина, в ній пишеться лише назва штату. Якщо ні, напишіть ім'я штату в першому штаті та напишіть список деяких внутрішніх дій або переходів штату у другому.

Ця операція змінює стан системи і може бути досягнута шляхом передачі повідомлення об'єкту, модифікації значення з'єднання або значення атрибута.

Кожна дія записується окремим рядком у форматі "мітка дії" / "вираз дії". Тег дії вказує середовище або умови для виконання діяльності, визначеної виразом дії. Вирази дій можуть використовувати будь-які атрибути та відносини, що належать до простору імен або контексту об'єкта моделювання.

Псевдо-стан - це вершина кінцевого автомата, що має форму стану, але не має поведінки. Прикладами псевдо-станів, визначених в *UML*, є початковий і кінцевий.

Перехід - зв’язок між двома станами, що означає, що об’єкт у першому стані повинен виконати певні операції і перейти в інший стан.

Операція переходу залежить не тільки від настання події, але і від реалізації конкретних умов, які називаються охоронцями. Якщо зазначена подія відбувається і значення умови захисту має значення "*true*", об'єкт переходить з одного стану в інший. На діаграмі стану перехід представлений суцільною лінією зі стрілкою, яка відображається із вихідного стану і вказує на цільовий стан.

Відповідно до типів подій, що відбуваються, *UML* розрізняє два типи переходів: триггерний та нетриггерний.

Якщо перетворення задається подією тригера, пов’язаною із зовнішнім станом, пов’язаним із станом, це називається тригером. У цьому випадку потрібно

вказати назву події у вигляді рядка тексту, що починається з малої літери поруч зі стрілкою тригера.

Якщо перехід відбувається після завершення діяльності в цьому стані, це називається не ініційованим. Для них поруч зі стрілкою переходу не вказано жодної назви події, а у початковому стані повинні бути описані внутрішні дії, після чого буде виконаний той чи інший не ініційований перехід, як на рис. 2.3.

Умова сторожового пса - логічна умова, вкладена в дужки. Це логічний вираз, значення якого взаємно виключається як " *true* " або " *false* ".

Рис. 2.3. Діаграма станів

Діаграма співпраці- взаємодії, де основна увага приділяється структурній організації об’єктів, що надсилають і отримують повідомлення.

Графік співпраці містить об'єкти, ці об'єкти є екземплярами класів, взаємозв'язки між ними, а об'єкти - екземплярами асоціацій та повідомлень. Стрілка повідомлення доповнює посилання і показує лише ті об’єкти, які беруть участь у реалізації симуляційного співробітництва. Далі, як і на схемі класів, структурні зв’язки між об’єктами відображаються у вигляді різних сполучних ліній. Відносини можуть бути доповнені назвою ролі, яку об’єкт відіграє у

відносинах. Він також описує динамічні взаємозв'язки - потік повідомлення у вигляді стрілки вказує напрямок уздовж сполучної лінії між об'єктами, а ім'я повідомлення та його порядковий номер вказуються у всій послідовності повідомлення, що видно на рис. 2.4.

У контексті *UML* усі об'єкти поділяються на дві категорії: пасивні та активні. Пасивні об'єкти працюють лише з даними і не можуть ініціювати діяльність з управління іншими об'єктами. Однак пасивні об'єкти можуть надсилати сигнали під час виконання запитів, якими вони обробляються. На схемі співпраці пасивні об’єкти зображуються звичним способом, без додаткових стереотипів.

Активний об'єкт має власний процес управління і може ініціювати діяльність з управління іншими об'єктами.

Мультиоб’єкт - це група анонімних об’єктів, яка може бути сформована на основі одного класу.

На діаграмі співпраці кілька об'єктів використовуються для відображення операцій та сигналів, адресованих усьому набору анонімних об'єктів.

Повідомлення на графіку співпраці представлені іншими стрілками поруч із відповідним з'єднанням або пов'язаною роллю. Напрямок стрілки вказує одержувача повідомлення. Поява стрілки повідомлення є значущим. Схема співпраці може використовувати один із трьох типів стрілок для позначення повідомлення

- Суцільна лінія з трикутною стрілкою вказує на виклик процедури (операції) або управління передачею потоку.
- Суцільна лінія з V-подібною стрілкою представляє асинхронне повідомлення в простому потоці управління.
- Пунктирна лінія з V-подібною стрілкою вказує на повернення із виклику процедури.

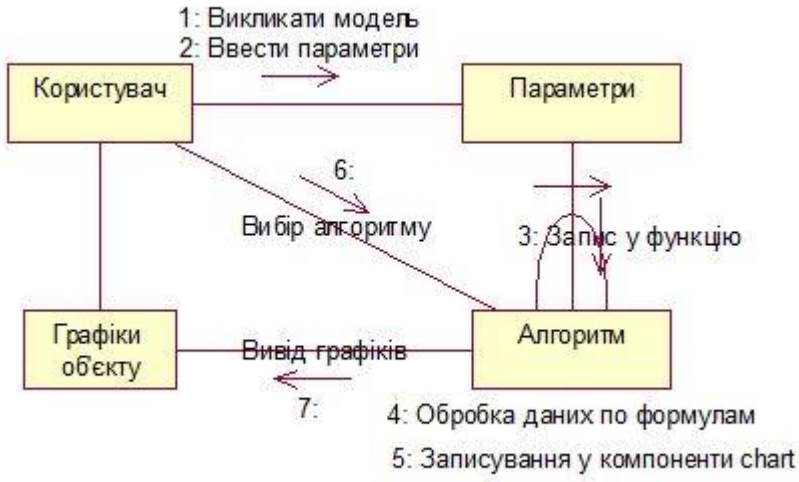

Рис. 2.4. Діаграма співпраці

Діаграма послідовності - показує часові характеристики об'єкта, що передає та приймає повідомлення. Оскільки діаграми робились у програмі *Rational Rose*, аби отримати діаграму послідовностей, необхідно розробити діаграму співпраці та натиснути F5 і отримаємо діаграму, що на рис. 2.5.

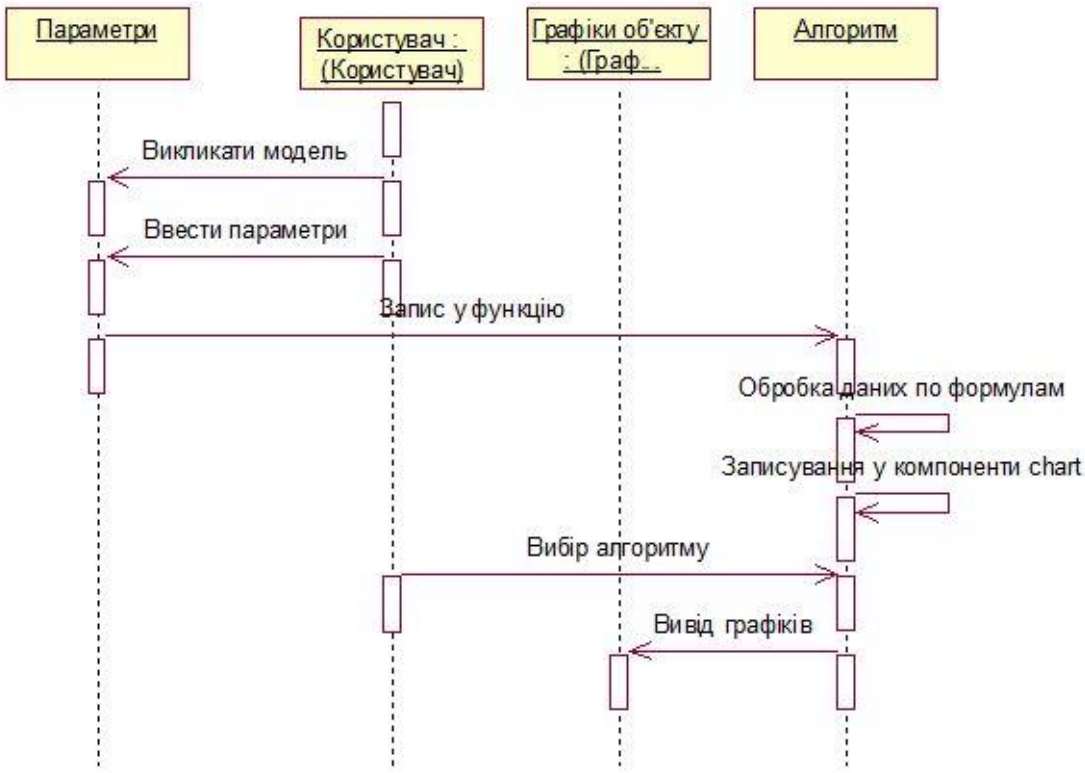

Рис. 2.5. Діаграма послідовності

Архітектура програмного забезпечення системи відображає організацію або структуру системи та надає пояснення того, як вона поводиться. Система представляє набір компонентів, які виконують певну функцію або набір функцій. Іншими словами, архітектура програмного забезпечення забезпечує міцну основу, на якій можна будувати програмне забезпечення.

Існує безліч моделей та принципів архітектури високого рівня, які зазвичай використовуються в сучасних системах.

Архітектура служить проектом системи. Він забезпечує абстракцію для управління складністю системи та встановлення механізму зв'язку та координації між компонентами. Він визначає структуроване рішення, яке відповідає всім технічним та експлуатаційним вимогам, одночасно оптимізуючи загальні ознаки якості, такі як продуктивність та безпека.

Далі, це передбачає набір важливих рішень щодо організації, пов’язаних із розробкою програмного забезпечення, і кожне з цих рішень може мати значний вплив на якість, ремонтопридатність, продуктивність та загальний успіх кінцевого продукту. Ці рішення складаються з:

- Вибір структурних елементів та їх інтерфейсів, за якими складається система;
- Поведінка, як зазначено у співпраці серед цих елементів;
- Склад цих структурних та поведінкових елементів у велику підсистему;
- Архітектурні рішення відповідають цілям бізнесу;
- Архітектурні стилі керують організацією.

Архітектура системи описує її основні компоненти, їх взаємозв'язки (структури) та те, як вони взаємодіють між собою. Архітектура та дизайн програмного забезпечення включає кілька таких факторів, як бізнес-стратегія, показники якості, людська динаміка, дизайн та *IT*-середовище, як на рис. 2.6.

Рис. 2.6. Структура архітектурного проектування

IT-середовище це сукупність апаратних і програмних засобів для реалізації проектів. У нашому випадку це ноутбук *HP 17-ak074ur*, операційна система *Windows*, середовище розробки *Visual Studio* – середовище для розробки з безліччю можливостей для написання, налагодження, тестування і розгортання коду на будь-якій платформі. Модуль написаний на мові програмування *C#* на платформі *Windows Forms* — фреймворк для розробки програм з примітивним, але зручним інтерфейсом на основі *.NET* розробки.

Бізнес-стратегія це план, у якому визначаються цілі, фінансові можливості та можливості проникання на ринок, пріоритети. Щодо цілей то це систематизація та звуження використання моделювання у певній області для більш кращого розуміння. Модуль має більш профільний та навчальний характер, тому розглядання з точки зору вигідності та виходу на ринок не має сенсу.

Показниками якості слугують критерії якості. Оскільки програма повинна показувати достатню точність, усі числові дані мають використовують *double*, а саме значення з плаваючою комою. Окремо присутній чисельний та графічний вивід результатів. Програма має бути зручною, з простим дизайном та інтерфейсом. Має виконувати свій функціонал та бути здатною до подальшого розвитку.

Людська динаміка, а саме людський ресурс вже передбачається суттю програми, оскільки її ціль це дослідження ознак об’єктів від дій людини. Також програма не має знатність до самонавчання, тому необхідна взаємодія «людина-машина». Також для подальшого прогресу, необхідний спеціаліст, що зможе програмно описувати нові моделі та алгоритми для їх дослідження, усувати помилки та інше.

Дизайн є стилем програми. Він визначає основний фундамент побудови додатку. Дизайн архітектури має бути гнучким, надійним і, звичайно, пристосований до використання.

2.2. Розробка алгоритму та аналіз динаміки

Кожне рішення проблеми починається з плану. Цей план називається алгоритмом. Алгоритм – шлях для рішення необхідної для когось задачі . Розробка та аналіз алгоритмів мають вирішальне значення для всіх аспектів інформатики: штучного інтелекту, баз даних, графіки, мереж, операційних систем, безпеки тощо. Розробка алгоритму - це більше, ніж просто програмування. Це вимагає розуміння альтернатив, які можуть бути використані для вирішення обчислювальної проблеми, включаючи апаратне забезпечення, мережу, мову програмування та обмеження продуктивності, які поставляються з будь-яким конкретним рішенням. У сенсі вирішення проблеми повністю та ефективно, йому також потрібно зрозуміти значення «правильного» алгоритму. Ви не можете написати програму, пояснити комп’ютеру, що йому потрібно робити, поки не зрозумієте самі, як виконувати завдання точно, поетапно [12]. Насправді це часто найскладніше в програмуванні.

При розв’язуванні будь-якого завдання та побудови алгоритму її розв’язку звичайно беруть до уваги представленням вхідних даних і мають уявлення про результат, що потрібно отримати. Будь-який алгоритм повинен мати такі основні характеристики:

– Детермінізм (детермінізм)-оскільки правила, встановлені в алгоритмі, абсолютно незрозумілі, застосування алгоритму до вхідного повідомлення повинно привести до результату;
– Дискретність - Процес, визначений алгоритмом, можна розділити (розділити) на окремі основні етапи (етапи), і кожен етап викликається в процесі або алгоритмі алгоритму;
– Якість - Алгоритм повинен бути придатним для вирішення всіх задач певного типу. Наприклад, алгоритм, що використовується для розв'язування лінійних рівнянь, повинен застосовуватися до системи, що складається з будь-якої кількості рівнянь. З цієї причини в якості вхідних даних допускається набір даних, тобто вихідну систему значень можна вибрати з деяких потенційно нескінченний набір;
– Ефективність-вказує на те, що існують такі варіанти вхідних даних. Для цих варіантів процес обчислення, що виконується даним алгоритмом, повинен зупинитися через обмежену кількість етапів (кроків) і дати необхідний результат або сигнал, вказуючи, що дані алгоритми є не підходить для вирішення проблем.

Отже, першим кроком буде розуміння і описання задачі. Необхідно досліджувати модель певного керованого об'єкту, і маючи образ схеми об'єкту ввести параметри які впливають на його властивості. На прикладі береться генератор руху БПЛА та алгоритм повороту, що використовується для нього. Для отримання різних графічних результатів ми задаємо параметри. Після отримуємо графічний і чисельний результат по руху об'єкта.

Другим кроком є аналіз та вимоги до рішення. У даному випадку ми маємо початкову схему з блоками, що підписані відповідними операціями над параметрами. Параметри повинні мати значення з області допустимих значень. Повинна показуватися стала напруги, а також початкова затримка. Окремо набір результуючих значень має бути у чисельному вигляді у відповідних полях. За кожний крок відповідає необхідна вкладка.

Алгоритм високого рівня складається з: обирання моделі, запис параметрів, після відбувається запис значень у масив. Цей масив використовується для визначення матричної експоненти для матриці переходу.

Взагалі, матрична експонента використовується для рішення лінійних систем диференціальних рівнянь зі сталими коефіцієнтами. Розраховується вона шляхом ставлення нашої матриці у ряд Тейлора. Далі для знаходження нашої функції ми перемножуємо матричну експоненту з попереднім значенням функції при умові, що є початкове її значення. Після робимо вивід у графічний компонент та текстовий компонент.

$$e^{At} = 1 + \frac{At}{1!} + \frac{A^2t^2}{2!} + \frac{A^3t^3}{3!} + \cdots, \ -\infty < x < \infty .$$

Для отримання графіка динаміки обираємо початкове значення функції, аргументом якої буде значення зміни у часі, а значенням функції буде рівняння вигляду:

$$Y[i+1] = e^{At}Y[i]$$

Важливо розуміти, що у цьому пункті розглядається модель генератору руху БПЛА, яка може бути представленою структурною схемою зі своїми ланками кожна з яких має свої параметри і інтеграторами. У даному випадку при подачі напруги ми маємо прохід першого значення для функції. Після проходу операцій з параметрами у інтеграторі ми отримуємо коефіцієнти для нашої матриці переходу. У даному випадку використовувалось дві різні матриці і відповідно результати по алгоритму, що застосовувався до моделі будуть різними.

2.3. Проектування інтерфейсу

Windows Forms - це технологія користувальницького інтерфейсу для *.NET*, що являє собою набір керованих бібліотек, що спрощує загальні завдання, такі як

читання та запис із файлової системи. Завдяки такому вдосконаленому середовищу, як *Visual Studio*, ви можете створювати інтелектуальні клієнтські програми *Windows Forms*, які відображають інформацію, легко вводяться користувачем та взаємодіють із віддаленими комп'ютерами в мережі.

Спочатку створюємо проект *C# Windows Forms*. Відкривається головна форма *Form.cs*. У вкладці властивості ставимо певні налаштування. У полі *Name* ставимо назву *Main, BackgroundImage* -> Завантажити і підкріплюємо фон для форми. Розміщення фону контролюємо через *BackgroundImageLayout* і обираємо положення *Stretch*. Також створюємо значок для нашого додатку. Для цього у полі *Icon* підкріплюємо значок типу *.ico*. Назва додатку буде *Analizer*, тому у полі *Text* пишемо *Analizer*. Програма буде на весь екран тому у полі *WindowState* обираємо *Maximized*.

Тепер розміщуємо компоненти нашої програми. Основна взаємодія користувача з програмою буде виконуватись через компоненти, які будуть розміщені у трьох компонентах *Panel*, панель, компонент управління *Windows Forms* для компонування груп інших компонентів і для розділу форми на функції.

На першу панель ставимо такі налаштування: у полі *Name* пишемо *panel1*. У властивості *Anchor*, що визначає до яких граней буде прикріплена дана панель, ставимо *Top, Left*. Окрема для більш зручнішого виділення утворимо рамки через властивість *BorderStyle*, обираючи фіксовану 3Д рамку. Окремо для панелі ставимо розміщення зліва. Для цього у властивості *Dock* обираємо *Left*. Також для панелі у властивості *Visible* робимо елемент прихованим через значення *False*. Ця панель буде для параметрів моделі. Розміщаємо у панелі компоненти *Label* та *TextBox* для відображення назв та запису значень параметрів.

Друга панель має подібні налаштування, але *Dock* має значення *Top* та *Name* з назвою *panel2*. У цій панелі буде розміщено компонент *PictureBox*. Цей компонент слугує для завантаження зображення у програму. Для нього ставимо *BackgroundImageLayout* значення *Zoom*. Властивість *Location* для розміщення *PictureBox* у панелі 20; 20 по *X* та *Y* відповідно. Властивість *Visible* також ставимо *False*.

Остання панель зроблена для відображення результатів. Налаштування аналогічні попереднім, крім *Dock*, що має значення *Bottom*. На ній розміщуємо три компоненти типу *Chart*. У властивостях *Serias* налаштовуємо тип графіків у редакторі колекції. У кожному *Chart* створюємо 2 члени, де *BorderWidth* дорівнює 2, та задаємо для графіків різні кольори.

Тепер для інтерфейсу створимо навігаційне меню. Для цього в панелі елементів знаходимо компонент *MenuStrip*. Ставимо властивість *Dock* у значення *Top*. Цей компонент є контейнером для компонентів *ToolStripMenu.* Для них робимо необхідні налаштування. Також у редакторі колекції можна створити підменю з кнопками, які будуть необхідні для навігації меню. Меню складається з таких кнопок як «Математична модель», «Алгоритм», «Справка», «Про Автора», «Вихід». Також «Математична модель» та «Алгоритм» мають підменю для обирання моделі та алгоритму, а також для закриття панелей. Повний інтерфейс відображено на рис. 2.7.

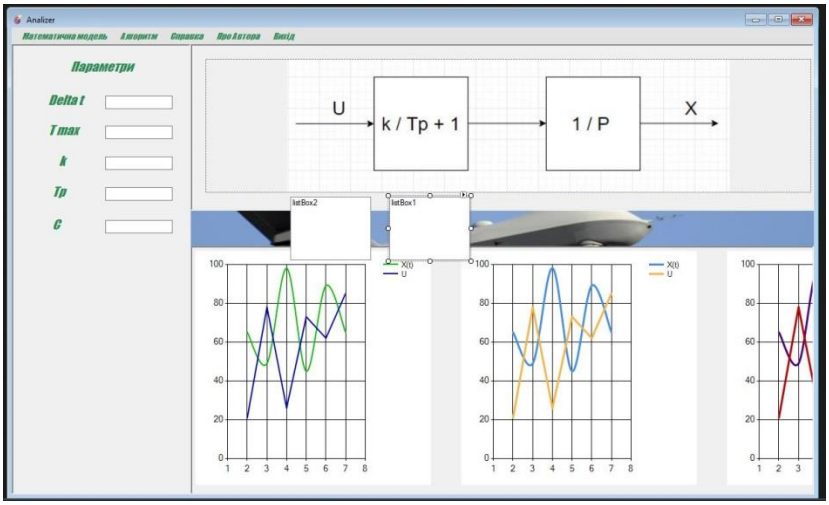

Рис. 2.7. Вигляд інтерфейсу

Окремо слід виділити вкладку «Довідка» , що на рис. 2.8. Вона відкриває нове вікно 700 на 700. У цю форму заносимо компонент *RichTextBox* у яке завантажується файл типу *RTF* з довідкою для користувача, а також кнопки «Вихід» для завершення роботи і «Про Автора» з даними про розробника.

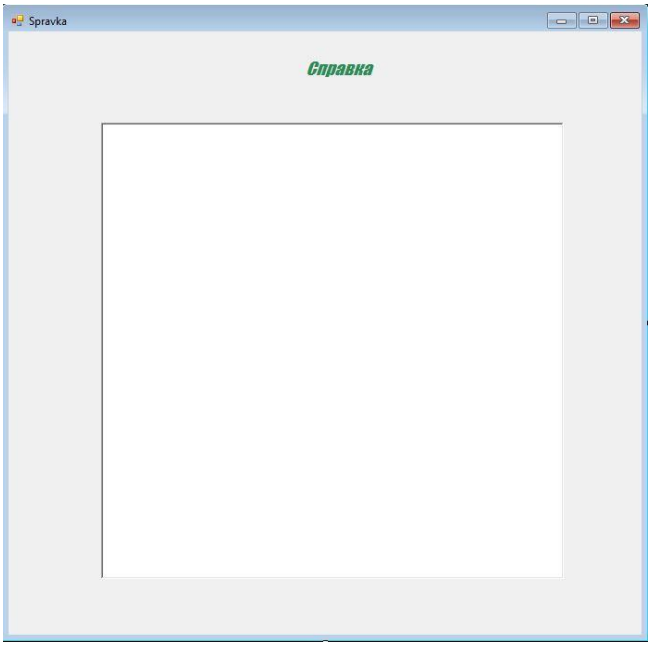

Рис. 2.8. Вікно довідки

Також для програми є майстер встановлення, який зображений на рис. 2.9. Його інтерфейс має базові налаштування додатку з обранням шляху розміщення та текстовими підказками.

Рис. 2.9. Майстер встановлення

2.4. Висновки до розділу

На цьому етапі провели повне проектування. Спершу за обраною методологією визначили нашу структуру програми та її модулів. Для кожного з них описали алгоритм дії. Визначили зв’язки між компонентами. Розробили алгоритм знаходження матричної експоненти та використали її у алгоритмі для будування графіку поведінки об’єкта від залежних параметрів та зміни часу. Провели аналіз динаміки, що використовувалася у даному проекті. Оформили повний опис розроблення інтерфейсу, вказавши усі використані елементи, зв’язки між формами та створення меню з функціональними компонентами. Окремо навели приклад інтерфейсу майстера встановлення для додатку.

РОЗДІЛ 3
ОПИС ТА ПЕРЕВІРКА

3.1. Тестування

Тестування програмного забезпечення це процес верифікації та валідації на предмет того, що програмний модуль відповідає технічним та бізнес вимогам. Верифікація це перевірка документації та коду, а валідація відповідає за оцінку продукту в цілому. Бувають різні види тестування тому опишемо лише по даному програмному модулю. Наше тестування буде: мануальним, тестування Альфа-тестування.

По напрямку сценарію:

– позитивне;

– негативне.

По ступені знань по системі:

– чорний;

– білий ящик.

Основним є тестування по об’єкту, що включає в себе функціональне та не функціональне тестування.

На сьогоднішній день багато обсягів інформації необхідні у деталізації та розумінні. Особисто у сфері ІТ. Незалежно від того, що світ максимально використовує інформаційні технології, деякі сфери людської діяльності досі не мають чіткої реалізації у комп’ютерних технологіях. Одним з таких шляхів є математичне моделювання, яке на сьогоднішній час широко використовуваний, але в теорії та математичних обрахуваннях. Так, є різні пакети, які дозволяють працювати з такими термінами та реалізовують певні аспекти цієї області.

Однак на даний час немає звужено профільних програм або ПЗ, що дозволяють фахівцям у певному напрямку цієї сфери діяльності більш легше використовувати свої знання, працювати з обраними темами та робити необхідні дії такі, як аналіз, дослідження, перевірку, пізнання математичних моделей певних об’єктів та їх реалізацію. Це також стосується огляду та розуміння основних понять що таке математичне моделювання, самі моделі, як вони використовуються, що вони відображають по відношенню до об’єкта та яке саме значення має розуміння та функціонування даного об’єкта у певній системі чи області використання.

Тому необхідне такі модулі, що будуть більш систематизувати та відокремлювати різні напрямки у математичному моделюванні, оскільки у різних спеціалістів є свій набір задач, який є дуже вузькопрофільним. Одним з таких є розуміння, робота, та отримання результатів динамічних властивостей математичних моделей.

Серед протестованих компонентів будуть:

– Навігаційне меню;

– Запис даних;

– Відображення результатів.

Щодо компонентів, які не тестуються то це:

– Вкладки: Про Автора, Вихід;

– Інформаційні компоненти(*Label, RichTextBox*).

Критерії якості та прийнятності

– Точність;

– Простота;

– Зрозумілий інтерфейс.

Робота програми як і сама програма є прийнятною, у разі дотримання усіх вище зазначених критеріїв.

Таблиця 3.1

Опис тест-кейсів

№ Тест-кейсу	Параметр	Вхідні дані	Тип вхідних даних	Очікуваний результат
1	Delta t	0,8	double	Запис у змінну
	Tp	2	integer	Запис у змінну
	k	1	integer	Запис у змінну
	Tmax	8	integer	Запис у змінну
	Матриця C	1	integer	Запис у змінну
2	Delta t	.юж	double	Повідомлення про помилку
	Tp	---	integer	Повідомлення про помилку
	k	1,367	integer	Повідомлення про помилку
	Tmax	ні	integer	Повідомлення про помилку
	Матриця C	28,7	integer	Повідомлення про помилку

Таблиця 3.2

Перевірка компонентів на проходження тестування

Компоненти	Тестування
Вкладка «Математичні моделі»	Passed
Вкладка «Алгоритми»	Passed
Вкладка «Довідка»	Passed
k, Tp, Матриця C	Passed
delta t, Tmax	Skipped

3.2. Інструкції користувача

Спочатку програму потрібно поставити на свій комп’ютер через майстра встановлювання, що продемонстровано на рис. 3.1.

Рис. 3.1. Майстер встановлення

Далі необхідно обрати необхідний шлях для встановлення даної програми як це зображено на рис. 3.2. Як для користувача, так і для розробника у пакеті для встановлення є як проект розробки на *Visual Studio*, так і *exe* файл.

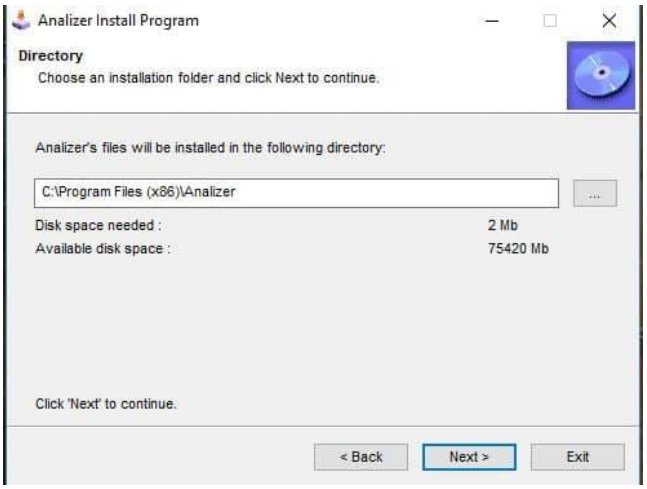

Рис. 3.2. Вибір шляху для зберігання

Після установлення запускається додаток та відкривається головне вікно. Зверху є навігаційне меню з усім необхідним для користувача. Перша вкладка це «Математичні моделі». Цей пункт меню містить в собі підменю з моделями, що досліджуються і яке відкрите на рис. 3.3.

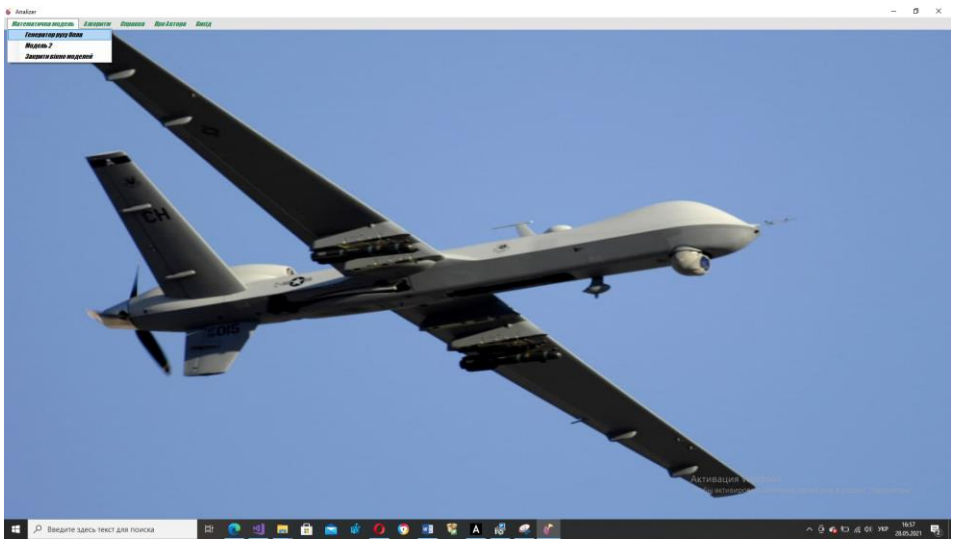

Рис. 3.3. Вибір моделі

При обиранні моделі відкриваються два вікна. Верхнє містить схему елементу об’єкту, а панель зліва містить параметри які впливають на подальший результат . Після введення параметрів обираємо вкладку «Алгоритм» і після у підменю обираємо необхідний нам алгоритм, як це робиться на рис. 3.4.

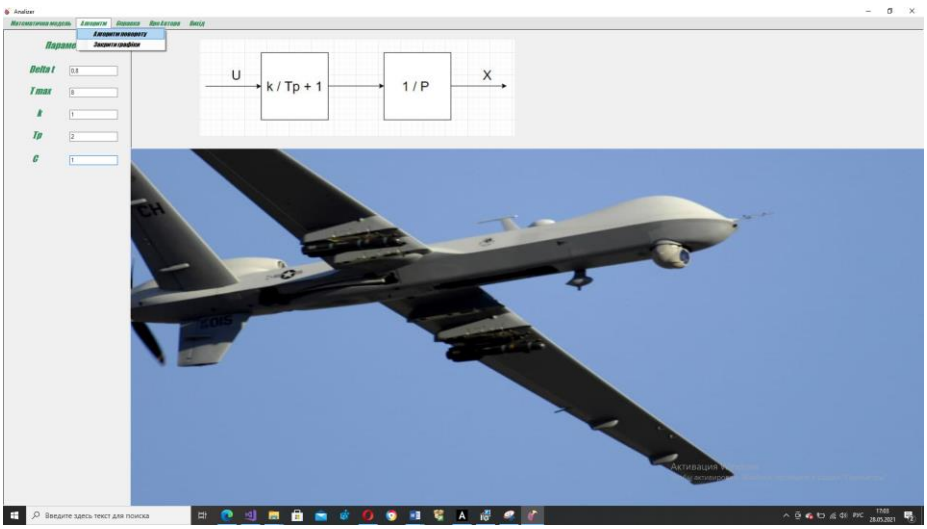

Рис. 3.4 Вибір алгоритму

Після обрання з’являється нижня панель з графіками зміни динамічних властивостей об’єктів, які також видні на рис. 3.5.

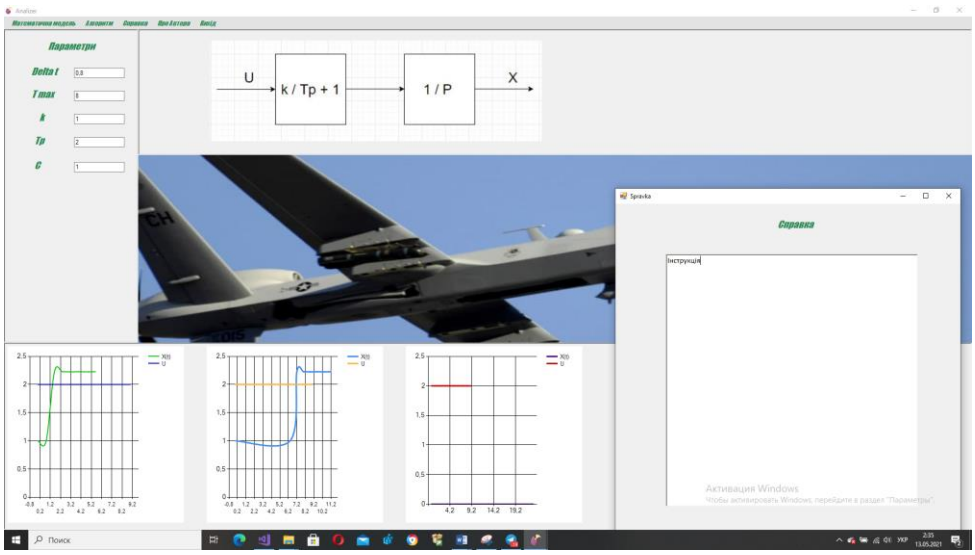

Рис. 3.5. Отримання результатів

Для закривання моделі у вкладці підменю обираємо вкладку «Закрити вікна моделі». Для відкриття довідки необхідно натиснути на вкладку «Довідка», а для виходу із програми вкладку «Вихід».

3.3. Висновки до розділу

В результаті роботи був створений модуль для необхідних робіт у сфері моделювання. В цьому описані основні, фундаментальні принципи моделювання та продемонстровано модель об’єкта у різних виглядах. У подальшій розробці можливо додати ще об’єкти для подальших досліджень, доробити інтерфейс, розробити кросс-платформленість. Також необхідно для подальших робіт використовувати вбудований класс *Matrix*. Він містить операції над матрицями, їх використання у необхідних методах та інше. Додатково необхідні допрацювання у відображенні результатів та інтерфейсі в цілому. Наприклад використання *WPF* замість *WinForms*. Дана платформа є більш гнучкою від використовуваної.

ВИСНОВКИ

У результаті виконаної роботи можна підбити підсумки. У дипломній роботі необхідно було створити модуль для аналізу динамічних властивостей математичних моделей керованих об’єктів. Інакше кажучи створити програму для аналізу та моделювання певних компонентів.

В процесі роботи було реалізовано базовий функціонал для відображення результатів при досліді об’єктів, наприклад генератор БПЛА та алгоритм повороту. Було реалізовано матрицю переходу через ряди та використано у обрахуванні значень для графічних результатів.

Щодо мінусів, то робота з моделями може бути більш функціональною та точнішою. Також малий коефіцієнт реалізації моделей для аналізу, оскільки їх реалізація у різних формах та робота з ними потребують вагомих знань у фізиці та теорії, а також розуміння математичних операцій, що використовуються.

В додаток є проблема через графічні можливості які використовувалися в процесі моделювання, оскільки такі фреймворки як *WPF* та *WinForms* не підтримують 3мірну графіку. Так, вони можуть робити так названу оманливу 3мірну графіку через промальовування по компоненту *timer* та алгоритму рейкастингу аби реалізувати 3мірну модель об’єкта, що буде виконувати певні дії в залежності від параметрів користувача.

Але це потребує багатих знать у *.NET*, додаткової бібліотеки *OpenGL* та знань з її використання або ж використання ігрової платформи *Unity*.

Оскільки для показу та простого дослідження достатньо 2-мірної графіки, було вирішено використовувати її, але деякі моделі просто неможливо буде коректно відобразити у такий спосіб. Було розроблено інтерфейс та створено майстер встановлювання програми для поширення у навчальних цілях чи набуття практичних навичок. Дизайн є зручним і простим. У заключення можна сказати, що продукт є досить сирим і мають місце допрацювання, проте основний скелет та реалізацію певних моделей реалізовано.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Комп'ютерне моделювання систем та процесів. Методи обчислень. Частина 1 : навчальний посібник / Квєтний Р. Н., Богач І. В., Бойко О. Р., Софина О. Ю., Шушура О.М.; за заг. ред. Р.Н. Квєтного. – Вінниця: ВНТУ, 2012. – 8с.

2. *Model Reduction and Coarse-Graining Approaches for Multiscale Phenomena, Springer*, *Complexity series*, *Berlin-Heidelberg-New York*, 2006. XII+562 pp.

3. *Данилов Ю. А.*, Лекции по нелинейной динамике. Элементарное введение. Серия «Синергетика: от прошлого к будущему». Изд.2. — М.: *URSS*, 2006. — 208 с.

4. *Анищенко В. С.*, Динамические системы, Соросовский образовательный журнал, 1997, № 11, с. 77-84.

5. *Советов Б. Я., Яковлев С. А.*, Моделирование систем: Учеб. для вузов — 3-е изд., перераб. и доп. — М.: Высш. шк., 2001. — 343 с.

6. Звонарев, С.В. З-42 Основы математического моделирования: учебное посо- бие / С.В. Звонарев. — Екатеринбург : Изд-во Урал. ун-та, 2019. с. 10-11.

7. Комп'ютерне моделювання систем та процесів. Методи обчислень. Частина 1 : навчальний посібник / Квєтний Р. Н., Богач І. В., Бойко О. Р., Софина О. Ю., Шушура О.М.; за заг. ред. Р.Н. Квєтного. – Вінниця: ВНТУ, 2012. – 45с.

8. *Дьяконов В. П.* Справочник по применению системы *PC MATLAB*. — М.: «Физматлит», 1993. — 112 с

9. *Джон Г. Мэтьюз, Куртис Д. Финк.* Численные методы. Использование *MATLAB = Numerical Methods: Using MATLAB*. — 3-е изд. — М.: «Вильямс», 2001. — 720 с.

10. *Moler, C. A Brief History of MATLAB - MATLAB & Simulink*.

11. *Books on wide variety of technical subjects referencing VisSim*.

12. *Visual simulation with student VisSim, by Karen Darnell*, 1996, *PWS Pub. Co., Boston, ISBN* 0-534-95485-5

13. *Myers, Glenford J.*, 1946-*The art of software testing / Glenford J. Myers, Corey Sandler, Tom Badgett*, 2012. — 41 с

14. ДСТУ 3008:2015 «Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення».
15. ГОСТ 2.106-96 «Единая система конструкторской документации. Текстовые документы».
16. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проєкти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.

ДОДАТОК А

Головне вікно програми та опис алгоритму для моделі

Фрагмент коду *Form.cs*

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp2
{
   public partial class Main : Form
   {
      public Main()
      {
         InitializeComponent();
      }

      private void toolStripMenuExit_Click(object sender, EventArgs e)
      {
         Close();
      }

      private void toolStripMenuGenR_Click(object sender, EventArgs e)
```

```
{

    panel2.Visible = true;
    panel1.Visible = true;
    pictureBox1.Visible = true;
}

private void toolStripMenuZakMod_Click(object sender, EventArgs e)
{
    panel1.Visible = false;
    panel2.Visible = false;
    pictureBox1.Visible = false;
}

public static double[,] Exp(double[,] A, double t, int n)
{
            double[,] exp = { { 1, 1, 1 }, { 1, 1, 1 }, { 1, 1, 1 } };

    for (int i = 1; i < n; i++)
    {
        double[,] mult = MultiplyMatrix(PowMatrix(A, i), Math.Pow(t, i));


        double[,] divis = DivisionMatrix(mult, Factorial(i));


        exp = AddMatrix(exp, divis);
    }
```

```
    return exp;
}

      public static int Factorial(int number)
{
    int fact = 1;

    for (int i = 2; i <= number; i++)
    {
       fact *= i;
    }

    return fact;
}


public static double[,] PowMatrix(double[,] A, int step)
{
    double[,] B = A;

    for (int i = 0; i < step; i++)
    {
       B = MultiplyMatrix(A, B);
    }

    return B;
}


public static double[,] AddMatrix(double[,] A, double[,] B)
```

```
{
    int rA = A.GetLength(0);
    int cA = A.GetLength(1);

    for (int i = 0; i < rA; i++)
    {
        for (int j = 0; j < cA; j++)
        {
            A[i, j] += B[i, j];
        }
    }

    return A;
}

        public static double[,] DivisionMatrix(double[,] A, double b)
{
    int rA = A.GetLength(0);
    int cA = A.GetLength(1);

    for (int i = 0; i < rA; i++)
    {
        for (int j = 0; j < cA; j++)
        {
            A[i, j] /= b;
        }
    }

    return A;
}
```

```
        public static double[,] MultiplyMatrix(double[,] A, double b)
{
    int rA = A.GetLength(0);
    int cA = A.GetLength(1);

    for (int i = 0; i < rA; i++)
    {
        for (int j = 0; j < cA; j++)
        {
            A[i, j] *= b;
        }
    }

    return A;
}


public static double[,] MultiplyMatrix(double[,] A, double[,] B)
{
    int rA = A.GetLength(0);
    int cA = A.GetLength(1);
    int rB = B.GetLength(0);
    int cB = B.GetLength(1);

    double temp = 0;
    double[,] C = new double[rA, cB];

    if (cA != rB)
    {
```

```
            throw new Exception("matrik can't be multiplied !!");
        }
        else
        {
            for (int i = 0; i < rA; i++)
            {
                for (int j = 0; j < cB; j++)
                {
                    temp = 0;
                    for (int k = 0; k < cA; k++)
                    {
                        temp += A[i, k] * B[k, j];
                    }
                    C[i, j] = temp;
                }
            }
            return C;
        }
          }
private void toolStripMenuAlPov_Click(object sender, EventArgs e)
              {
double[,] X = new double[3,3];
                  double[] t = new double[n];
double[,] A = { { 0, 1, 0 }, { 0, 0, 0 }, { 0, 0, 0 } };
        int u = 2;
        double deltat = Convert.ToDouble(textBox1.Text);
        int n = Convert.ToInt32(textBox2.Text);
        int k = Convert.ToInt32(textBox3.Text);
        int Tp = Convert.ToInt32(textBox4.Text);
              int c = Convert.ToInt32(textBox5.Text);
```

```
A[1,0] = (-1) / Tp;
        A[1,1] = k / Tp;
                A[2,1] = A[2,2] = c;

       X[0] = 1;
        double[,] exp = Exp(A, deltat, n);



        int rExp = A.GetLength(0);
        int cExp = A.GetLength(1);
        int k = 0;

        for (int i = 0; i < rExp; i++)
        {
           for (int j = 0; j < cExp; j++)
           {
              X[k] = exp[i,j] * deltat) * X[k-1];
              k++;
            }
        }
        for (int i=0; i<10;i++) {

           this.chart1.Series["U"].Points.AddXY(x,u);
           this.chart2.Series["U"].Points.AddXY(x, u);
x=x+1;
      }
for(int i = 1; i < n; i++)
        {

           t[i] = t0 + deltat;
```

```
            t0 = t0 + deltat;
        }
this.chart1.Series["X(t)"].Points.DataBindXY(t, X);
        A[0,2] = ((-1) * k) / Tp;  A3
        A[1,0] = k / Tp;
        double[,] exp = Exp(A, deltat, n);


        int rExp = A.GetLength(0);
        int cExp = A.GetLength(1);
        int k = 0;

        for (int i = 0; i < rExp; i++)
        {
            for (int j = 0; j < cExp; j++)
            {
                X[k] = exp[i,j] * deltat) * X[k-1];
                k++;
             }
        }
this.chart2.Series["X(t)"].Points.DataBindXY(t, X);


}
private void toolStripMenuGraf_Click(object sender, EventArgs e)
    {
        panel3.Visible = false;
    }

    private void toolStripMenuSprawka_Click(object sender, EventArgs e)
    {
```

```
        Spravka formspr = new Spravka();
        formspr.StartPosition = FormStartPosition.CenterScreen;
        formspr.ShowDialog();
            }
private void toolStripMenuSprawka_Click(object sender, EventArgs e)
    {
        Autor formaut = new Autor ();
        formaut.StartPosition = FormStartPosition.CenterScreen;
        formaut.ShowDialog();
            }
```

ДОДАТОК Б

Вікно Довідки

Фрагмент коду *Spravka.cs*

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp2
{
   public partial class Spravka : Form
   {
      public Spravka()
      {
```

```
            InitializeComponent();
        }

        private void Spravka_Load(object sender, EventArgs e)
        {

this.richTextBox1.LoadFile("C:\\Users\\Женя\\source\\repos\\диплом\\WindowsFormsA
pp2\\справка.rtf");
        }
    }
}
```

ДОДАТОК В

Вікно Автора

Фрагмент коду Spravka.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApp2
{
    public partial class Autor : Form
    {
        public Autor()
```

```
        {
            InitializeComponent();
        }

        private void Autor_Load(object sender, EventArgs e)
        {

this.richTextBox1.LoadFile("C:\\Users\\Женя\\source\\repos\\диплом\\WindowsFormsA
pp2\\автор.rtf");
        }
    }}
```