Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи
Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: «Розробка алгоритмів і програмного забезпечення для моделювання числових послідовностей даних з нерівномірним кроком»
за освітньою програмою 12 Інженерія програмного забезпечення
зі спеціальності: 121 Інженерія програмного забезпечення

Виконав: студент групи ПЗ20130: _____ /Ігор ПОПОВ/
(підпис)

Керівник: _____ _____ /Владислав СКАЛОЗУБ/
(посада) (підпис)

Нормоконтролер: _____ _____ /Світлана ВОЛКОВА/
(посада) (підпис)

Засвідчую, що у цій роботі нема запозичень з праць інших авторів б відповідних посилань.

Студент _____
(підпис)

Дніпро – 2023 рік

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty Computer technologies and systems
Department Computer information technology

Explanatory Note
to Bachelor's Thesis

on the topic: «Development of algorithms and software for modeling numerical sequences of data with an uneven step»
according to educational curriculum 12 software engineering
in the Speciality: 121 software engineering

Done by the student of the group PZ20130: ___ (посада) _____ (підпис) /Ihor POPOV/

Scientific Supervisor: ___ (посада) _____ (підпис) /Vladislav SKALOZUB/

Normative controller: ___ (посада) _____ (підпис) /Svitlana VOLKOVA/

Dnipro – 2023

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Спеціальність: «Інженерія програмного забезпечення»

«ДО ЗАХИСТУ»
Завідувач кафедри
_____ Вадим ГОРЯЧКІН
(підпис) (ПІБ)
202_ р. _____ «_____»

ЗАВДАННЯ
до дипломної роботи на здобуття ОС «бакалавр»

студента групи ПЗ20130 Попова Ігоря Юрійовича
номер групи (ПІБ)

1. Тема роботи: Розробка алгоритмів і програмного забезпечення для моделювання числових послідовностей даних з нерівномірним кроком
затверджена наказом по університету від «07» грудня 2022 р. № 1209 ст.
2. Термін подання студентом роботи «20» червня 2023 р.
3. Вихідні дані до дипломної роботи Приклади постановок завдань моделювання числових послідовностей даних з нерівномірним кроком, особливості завдань і спеціалізовані математичні моделі для моделювання процесів з нерівномірним кроком спостережень, нечіткі реляційні та квантильні алгоритми моделювання недетермінованих процесів, приклади застосування сепарабельних моделей та процедур для дослідження процесів з нерівномірним кроком спостережень
4. Зміст пояснювальної записки (перелік питань до розробки):
4.1 Аналітична частина: огляд досліджень щодо можливостей реалізації завдань моделювання числових послідовностей даних з нерівномірним кроком та відомих програмних засобів, постановки нових завдань сфери моделювання комбінованих процесів з нерівномірним кроком спостережень.
4.2 Основна частина: розробка комбінованих нечітких реляційних моделей та алгоритмів аналізу процесів з нерівномірним кроком першого порядку, розробка комбінованих нечітких реляційних моделей процесів з нерівномірним кроком другого порядку, розробка алгоритмів і програмних засобів для дослідження та прогнозування числових послідовностей даних з нерівномірним кроком, проведення тестування програми та числових експериментів з аналізу

ефективності запропонованих алгоритмів, результати досліджень достовірності та ефективності створених програмних засобів, технічна документація.
5. Перелік демонстраційного матеріалу: презентація результатів розробки програмного забезпечення для дослідження завдань моделювання числових послідовностей даних з нерівномірним кроком на основі комбінованих нечітких моделей першого та другого порядку, результати чисельних експериментальних досліджень алгоритмів і процедур комбінованих нечітких реляційних моделей процесів з нерівномірним кроком, програмні засоби для реалізації алгоритмів і процедур комбінованих нечітких реляційних моделей процесів з нерівномірним кроком, відео-демонстрація функціонування програмного комплексу.

КАЛЕНДАРНИЙ ПЛАН

№	Зміст роботи (розділу)	Термін виконання розділів роботи	Примітка
1	Вступ	01.02.2023 – 05.02.2023	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	06.02.2023 – 19.02.2023	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	20.02.2023 – 04.03.2023	
4	Постановка задачі, технічне завдання	05.03.2023 – 12.03.2023	30%
5	Техніко-економічні показники	13.03.2023 – 19.03.2023	
6	Розробка інструментальних засобів дослідження	20.03.2023 – 21.05.2023	60%
7	Виконання досліджень	22.05.2023 – 31.05.2023	
8	Оформлення результатів дипломної роботи	01.06.2023 – 18.06.2023	
9	Подання дипломної роботи до кафедри	20.06.2023	100%
10	Захист дипломної роботи на засіданні екзаменаційної комісії	28.06.2023	

Керівник дипломної роботи _____ Владислав СКАЛОЗУБ
(підпис) (ПІБ)

Завдання прийняв до виконання _____ Ігор ПОПОВ
(підпис) (ПІБ)

РЕФЕРАТ

Об’єктом дослідження та розробки дипломної роботи є нечіткі алгоритми і програмне забезпечення для моделювання числових послідовностей даних з нерівномірним кроком спостережень.

Мета роботи полягала у розвитку нечітких моделей недетермінованих часових послідовностей з нерівномірним кроком спостережень шляхом формування нових комбінованих математичних моделей, які відрізняються використанням двох (кількох) узгоджених часових послідовностей даних, а також у розробці програмного забезпечення, призначеного для реалізації комбінованих нечітких реляційних моделей процесів з нерівномірним кроком спостережень.

Для досягнення мети, розв'язання та опрацювання задач розробки були застосовані методи математичного моделювання та нечітких множин, методи Fuzzy Time Series першого та вищих порядків, а також методи програмної інженерії щодо формування вимог, проєктування та розробки програмного забезпечення.

При виконанні дипломної роботи було розроблено нову комбіновану нечітку реляційну математичну модель недетермінованих процесів з нерівномірним кроком спостережень, а також алгоритми операторів аналізу і прогнозування рівнів показників, що дозволяє моделювати та прогнозувати процеси зі змінними інтервалами. В роботі створене програмне забезпечення для дослідження процесів зазначеної категорії. Результати дипломної роботи становлять розвиток математичних моделей та програмного інструментарію, призначених для моделювання числових послідовностей даних з нерівномірним кроком спостережень, сприяють підвищенню точності результатів аналізу та ефективності комп’ютерних технологій моделювання недетермінованих процесів.

Розрахунково-пояснювальна записка складається зі вступу, 4 розділів, висновків, бібліографічного списку та додатків.

Ключові слова: недетерміновані процеси, нерівномірний інтервал спостережень, нечіткі реляційні моделі, комбіновані моделі нечітких послідовностей, програмне забезпечення, числові експерименти.

ЗМІСТ

ВСТУП

Застосування комп’ютерного моделювання зараз є майже необхідною складовою наукових та прикладних досліджень, проектування, розробок технологічних процесів. що забезпечує аналіз і прогнозування характеристик складних процесів і систем. При тому саме складність досліджуваних процесів являється одним із головних факторів, що визначають можливості застосування певних процедур аналізу та прогнозування, а також визначають ефективність програмних систем. У багатьох завданнях моніторингу стану інформаційних та виробничих систем, в завданнях аналізу станів технологічних процесів та складних систем тощо, в якості достовірних та доступних вихідних даних можливо отримати лише величини певних характеристик процесів, що упорядковані у часові послідовності, утворюють часові ряди. Завдання щодо формування моделей, методів та програмних засобів, призначених для виконання та достовірного застосування процедур комп’ютерне моделювання на основі часових рядів, широко застосовуються і являються актуальними.

Відзначається, що на практиці мають місце процеси, в яких характеристики часових рядів вимірюються з нерівномірним кроком міх спостереженнями. Такі часові ряди характерні для багатьох процесів моніторингу станів складних технічних, інформаційних процесів, а також для важливих і відповідальних процесів клінічного моніторингу лікування хворих, процесів реабілітації тощо. Широко відомі та застосовувані статистичні методи для завдань з перемінним кроком не можуть використані. Огляд літературних джерел показав, що математичні моделі та методи аналізу та моделювання часових рядів з перемінним кроком спостережень є не достатньо дослідженими. Також відзначається відсутність або дуже обмежене використання програмних засобів аналізу і прогнозування нерівномірних у часі часових послідовностей. В змістовному сенсі саме існування нерівномірного інтервалу мір спостереженнями в часових рядах представляє головну особливість завдань моделювання, робить дослідження та розробки своєчасними і актуальними.

Для реалізації завдань аналізу і прогнозування характеристик часових рядів з нерівномірним кроком спостережень була запропонована сепарабельна модель, а також удосконалений квантильний алгоритм. Засобами сепарабельної моделі були досліджені дані клінічного моніторингу процесів реабілітації хворих. Особливість сепарабельної моделі полягає в тому, що окремі характеристики таких часових рядів, в тому числі послідовності інтервалів, моделюються окремо. Їх узгодження забезпечується за рахунок номерів кроків процесу спостережень, що визначається аксіоматично, як реалізація принципу системної єдності. В дипломній роботі запропонований метод, який розвиває класичні моделі нечітких часових послідовностей (Fuzzy Time Series), коли на відміну від сепарабельних моделей розглядаються і явно поєднуються у моделі дані двох (кількох) характеристик процесу (сукупність ознак, одна з них – це інтервал між спостереженнями). Такі комбіновані моделі позначаються в роботі як Combined Fuzzy Time Series. Розробка алгоритмів і програмного забезпечення для моделювання числових послідовностей даних з нерівномірним кроком на основі Combined Fuzzy Time Series є актуальним науково-прикладним завданням.

Мета дипломної роботи полягає у розвитку нечітких моделей недетермінованих часових послідовностей у формі нечітких часових послідовностей, але з нерівномірним кроком спостережень, які також відрізняються використанням двох (кількох) узгоджених часових послідовностей даних, характеристик процесу (сукупність ознак, одна з них – це інтервал між спостереженнями), у формуванні та реалізації комбінованих моделей Combined Fuzzy Time Series. При цьому необхідно вирішити питання та виконати завдання щодо розробки математичного та програмного забезпечення, призначеного для реалізації комбінованих нечітких реляційних моделей процесів з нерівномірним кроком спостережень.

Призначення результатів розробки обумовлене новими можливостями застосування моделей та програмних засобів для методу комбінованих нечітких реляційних моделей Combined Fuzzy Time Series, що дозволяє реалізувати

процедури аналізу та прогнозування процесів з нерівномірним кроком спостережень.

Аналіз публікації та загальновідомих програмних засобів не дозволив виявити загально прийнятих і доступних програмних засобів реалізації завдань моделювання та дослідження даних з нерівномірним кроком спостережень, представлених у формі сукупності часових послідовностей часових рядів.

Створення спеціалізованого програмного забезпечення, як програмного інструментарію з автоматизації завдань аналізу та прогнозування процесів з нерівномірним кроком спостережень, визначає експлуатаційне призначення розробки.

РОЗДІЛ 1. ЗБІР ТА АНАЛІЗ ВИМОГ

1.1. Опис відомих методів

Були розглянуті відомі методи прогнозування, які використовуються для моделювання та прогнозування часових рядів. Враховуючи широке застосування прогнозування у різних галузях, таких як економіка, фінанси, природничі науки та інженерія, важливо мати на увазі різноманітні методи, які можуть бути використані для досягнення точних та надійних прогнозів.

1. Метод Чена

Основна ідея цього методу полягає в використанні моделі нечіткого часового ряду для прогнозування майбутніх значень. Метод Чена базується на використанні нечіткого набору правил для моделювання залежностей між вхідними та вихідними змінними. У цьому методі використовуються терміни та функції належності для виразу лінгвістичних змінних. Вхідні дані можуть бути нечіткими числами або нечіткими множинами, що характеризуються ступенями приналежності до різних лінгвістичних термінів.

Основні етапи методу Чена містять:

– згладжування даних. Спочатку необхідно виконати перший крок, а саме згладжування вхідного часового ряду даних. Це може бути виконано, наприклад, за допомогою таких методів як: ковзного середнього або експоненційного згладжування. Згладжування даних необхідно для того, щоб забезпечити зменшення шумів та випадкових варіацій, які можуть бути присутні в ряді;

– визначення нечіткого множника. Другим кроком є визначення нечіткого множника або нечіткого вимірювання, це необхідно зробити для того, щоб виконати оцінку неоднорідності та нечіткості в даних. Це може бути виконано, наприклад, за допомогою функцій належності, які вказують ступінь належності кожного значення до кожної категорії. На цьому кроці є можливість виокремити різні підінтервали або режими, які можуть існувати у часовому ряді;

– розбиття даних на підінтервали. Наступним кроком є розбиття даних на підінтервали на основі визначеного нечіткого множника. Це необхідно зробити

для того, щоб виділити різні режими або підмножини, що існують у часовому ряді. Розбиття на підінтервали допомагає утворити більш точні та релевантні моделі для кожного підінтервалу;

– визначення правил прогнозування. На цьому етапі виконується встановлення правила прогнозування на основі розбитих підінтервалів та вхідних даних. Ці правила описують залежність між попередніми та поточними значеннями у кожному підінтервалі. Використання нечітких правил дозволяє моделювати невизначеність та нечіткість у даних;

– прогнозування майбутніх значень. Застосовуючи встановлені правила прогнозування, метод Чена дозволяє прогнозувати майбутні значення відповідно до режиму, в якому знаходиться часовий ряд. Прогнози можуть бути здійснені на певний час вперед від останнього доступного значення;

– оцінка та аналіз прогнозів. Щоб оцінити точність прогнозів та їх ефективність вони піддаються оцінці та аналізу. Цей етап може включати порівняння прогнозів зі знаннями про дані, використання метрик оцінки прогнозів та інші аналітичні методи для оцінки результатів.

Метод Чена є ефективним інструментом для прогнозування часових рядів у нечіткій логіці. Він дозволяє моделювати невизначеність та нечіткість у даних, а також враховувати залежності та тенденції у часовому ряді [1; 3].

2. Метод Заде

Даний метод також відомий як нечітка логіка Заде, є одним із найвідоміших методів прогнозування у нечіткій логіці. Метод базується на ідеї використання нечітких множин для представлення нечіткості та невизначеності у даних. Він використовує концепцію лінгвістичних змінних та правил, що дозволяють враховувати нечіткість у процесі прийняття рішень.

Основна ідея методу Заде полягає у використанні чисел, які приймають значення від 0 до 1, для визначення ступенів приналежності об'єктів до певних нечітких множин. Ці числа називаються ступенями приналежності. Застосування

нечітких множин та ступенів приналежності дозволяє враховувати неоднозначність, невизначеність та нечіткість в даних.

Метод Заде використовує нечіткі правила для прогнозування. Ці правила виражають зв'язки між вхідними та вихідними змінними і базуються на ступенях приналежності. Наприклад, правило може мати вигляд «Якщо X є A, то Y є B», де X та Y – змінні, A та B – нечіткі множини [1].

Основними етапами методу Заде є:

– визначення лінгвістичних змінних. На першому етапі даного методу визначаються лінгвістичні змінні, які описують параметри часового ряду. Наприклад, якщо необхідно спрогнозувати температуру, можуть бути визначені лінгвістичні змінні, такі як «низька», «середня» та «висока»;

– визначення нечітких множин. На цьому етапі для кожної лінгвістичної змінної визначаються нечіткі множини, які відображають ступінь приналежності значень до кожної категорії. Наприклад, для лінгвістичної змінної «температура» можуть бути визначені нечіткі множини «холодно», «тепло» та «гаряче»;

– визначення нечітких правил: Цей етап передбачає формулювання нечітких правил, що описують зв'язки між лінгвістичними змінними. Наприклад, правило може мати вигляд: "Якщо температура є холодною і вологість є високою, то слід включити опалення";

– виконання нечітких правил. На цьому етапі виконуються нечіткі правила, які були визначені на попередньому, з метою отримання вихідного значення. Для цього застосовуються операції нечіткої логіки, такі як об'єднання, перетин та композиція, для того, щоб виконати обчислення вихідного значення на основі вхідних нечітких множин;

– дефазифікація. На останньому етапі використовується процес дефазифікації, він необхідний для того, щоб перетворити нечітке вихідне значення у конкретну числову величину або дискретний результат.

Метод Заде дозволяє моделювати та прогнозувати системи, які мають нечіткі та невизначені вхідні дані. Він дозволяє більш гнучко враховувати нечіткість та невизначеність, що часто зустрічаються у реальних системах [1; 3].

3. Метод Мамдані

Ідея даного методу полягає в тому, щоб змоделювати нечіткість і неоднозначність у вхідних даних, для цього використовують нечіткі множини та правила виведення. Замість того, щоб використовувати точні значення, як у традиційних методах, нечіткі множини визначають ступінь приналежності кожного елемента до кожної множини. Це забезпечує гнучкіше моделювання нечіткості та розмитості у вхідних даних. Правила базуються на логіці «якщо-це», «тоді-це» та виражають залежності між вхідними та вихідними змінними.

Метод Мамдані складається з таких етапів:

– визначення вхідних та вихідних змінних. На першому етапі необхідно спочатку визначити, які змінні будуть використовуватися для моделювання проблеми та які значення вони можуть приймати. Кожна із змінних пов'язується з нечіткою множиною, яка описує ступінь приналежності кожного значення до цієї множини;

– визначення лінгвістичних термів. На цьому етапі для кожної вхідної та вихідної змінної визначаються лінгвістичні терми, які описують значення цих змінних. Наприклад, якщо вхідна змінна – це швидкість, можуть бути визначені такі лінгвістичні терми, як «низька», «середня» і «висока»;

– визначення нечітких правил. Цей етап включає визначення нечітких правил, що вказують, які вихідні значення повинні бути прийняті в залежності від вхідних значень. Правила подаються у вигляді «ЯКЩО-ТОДІ» та використовують лінгвістичні терми та логічні оператори для моделювання умов та висновків. Наприклад, «ЯКЩО швидкість низька ТОДІ збільшити до рекомендованої»;

– розмиття вхідних значень. Перед застосуванням правил виведення потрібно розмити вхідні значення, використовуючи функції приналежності

нечітких множин. Це вирішується шляхом обчислення ступеня приналежності кожного значення до кожної нечіткої множини;

– виконання нечітких правил. Використовуючи розмиті нечіткі значення, застосовуються нечіткі правила для отримання вихідних значень. Це вирішується використанням логічних операцій, таких як максимум або мінімум, для того, щоб комбінувати нечіткі висновки;

– дефазифікація. Фінальний етап полягає у перетворенні нечітких вихідних значень у конкретні числові значення. Для цього можуть бути використані різні методи, такі як центр ваги або середнє значення [2; 4].

4. Метод Такагі-Сугено

Основна ідея даного методу полягає в тому, щоб використовувати лінійні регресійні моделі для виведення точних числових результатів замість нечітких висновків. Замість того, щоб мати нечіткі правила виведення, метод Такагі-Сугено використовує нечіткі правила для створення лінійних регресійних моделей для кожного правила. Кожне правило визначає лінійну функцію виведення, яка залежить від вхідних змінних. Основні етапи методу Такагі-Сугено наступні:

– визначення вхідних та вихідних змінних. На першому етапі, аналогічно до методу Мамдані, потрібно визначити вхідні та вихідні змінні, які моделюють проблему або систему, що вивчається. Кожна змінна пов'язується з нечіткою множиною, яка визначає ступінь належності кожного значення до цієї множини;

– визначення нечітких правил. Цей етап передбачає, що потрібно визначити нечіткі правила виведення, які пов'язують вхідні змінні із вихідними. Кожне правило визначає лінійну функцію виведення, яка залежить від вхідних змінних;

– обчислення ступенів активації. Для того, щоб розрахувати ступені активації для кожного правила, необхідно використовувати функції приналежності вхідних значень. Це визначає, наскільки кожне правило буде активовано на основі вхідних значень;

– використання лінійних регресійних моделей. Для кожного правила створюється лінійна регресійна модель, яка використовує ступені активації та вхідні значення для виведення точних числових результатів;

– агрегація вихідних значень. На останньому етапі результати, що були отримані від кожної лінійної регресійної моделі, агрегуються для отримання остаточного вихідного значення. Це може включати в себе використання вагових коефіцієнтів для кожного моделі або інших методів агрегації [1; 2].

5. Метод Чебишева

Метод Чебишева також відомий як метод максимального відхилення, є одним з методів прогнозування в нечіткій логіці. Цей метод базується на використанні множин Чебишева для апроксимації функції вхідних даних. Ці множини, які були введені Пафнутієм Чебишевим, є математичними об'єктами, що дозволяють апроксимувати функції з високою точністю. Дані множини характеризуються специфічним розподілом точок у заданому діапазоні. Вони можуть бути використані для опису нечітких множин, де кожна точка представляє ступінь приналежності об'єкту до даної множини.

У методі Чебишева для прогнозування використовуються правила, які описують залежності між вхідними і вихідними змінними. Кожне правило містить функцію приналежності, яка описує ступінь приналежності вхідних даних до кожної множини Чебишева. Ці функції приналежності можуть бути задані у вигляді трикутних, трапецевидних або інших форм [2; 4].

Процес прогнозування у методі Чебишева включає такі етапи:

– визначення вхідних та вихідних змінних. Спочатку потрібно визначити вхідні змінні, які впливають на прогноз, а також вихідні змінні, які потрібно прогнозувати;

– визначення множин Чебишева. На цьому етапі для кожної вхідної змінної визначаються множини Чебишева з відповідними функціями приналежності. Ці множини можуть бути задані експертно або визначені за допомогою алгоритмів кластеризації;

– формулювання правил. Цей етап передбачає створення правил, які описують залежності між вхідними та вихідними змінними. Кожне правило містить функцію приналежності для вхідних змінних і функцію вихідної змінної;

– інференція. На даному етапі застосовуються правила для визначення ступенів приналежності вихідних змінних до відповідних множин Чебишева. Тут також може бути пошук максимальної або середньої приналежності;

– агрегація. Для того, щоб виконати обчислення остаточних прогнозних значень вихідних змінних на основі ступенів приналежності використовуються алгоритми агрегації;

– дефазифікація. Останній етап передбачає перетворення остаточних прогнозних значень з нечітких множин у конкретні числові значення [4].

1.2. Опис аналогів програм

На сьогоднішній день, розробка програм, що реалізують методи нечіткої логіки та прогнозування, є активним напрямом досліджень. Велика кількість програм та бібліотек доступні для використання, які забезпечують реалізацію відомих методів, таких як методи Мамдані, Чена, Заде та інших.

У зв’язку з тим, що метод комбінованих нечітких реляційних моделей Combined Fuzzy Time Series є унікальним та новим, і не має явних аналогів серед програм. Це означає, що не існує програми або бібліотеки, яка спеціально розроблена для реалізації саме цього методу прогнозування в нечіткій логіці. Проте, відомі методи нечіткої логіки та прогнозування можуть бути реалізовані за допомогою існуючих бібліотек та середовищ програмування, які надають загальні інструменти для моделювання нечітких систем та прогнозування.

Значні досягнення в галузі нечіткої логіки та прогнозування зроблені завдяки розробці спеціалізованих програм та бібліотек. Ці програми надають інструменти для реалізації складних алгоритмів та методів, що використовуються у нечіткій логіці. Для цього можна використовувати існуючі бібліотеки та середовища програмування, такі як MATLAB, Python з

бібліотеками SciPy та scikit-fuzzy, або R з пакетами для нечіткої логіки, для реалізації відомих методів.

1.3. Постановка задачі

Розробити алгоритм моделювання числових послідовностей даних з нерівномірним кроком і програмного забезпечення, яке його реалізує. У програмі необхідно передбачити введення користувачем кількості послідовностей і комбіновану нечітку реляційну математичну модель процесів з нерівномірним кроком спостережень, яка поєднує дві перемінні послідовності – контрольований параметр та інтервал між спостереженнями.

Користувач, використовуючи розроблений алгоритм, та програмне забезпечення, може отримувати створені ряди спостережень, та прогнозування значень.

Перелік завдань, які необхідно вирішити:

– розробка комбінованих нечітких реляційних моделей та алгоритмів аналізу процесів з нерівномірним кроком першого порядку;

– розробка комбінованих нечітких реляційних моделей процесів з нерівномірним кроком другого порядку;

– розробка алгоритмів і програмних засобів для дослідження та прогнозування числових послідовностей даних з нерівномірним кроком.

Висновки до розділу:

У ході дослідження було розглянуто деякі відомі методи прогнозування в нечіткій логіці, зокрема методи Чена, Заде, Чебишева, Такагі-Сугено та Мамдані. Кожен з цих методів має свої особливості і застосовується для прогнозування в різних областях. На основі проведеного аналізу можна сформулювати список основних функціональних вимог для проектування та реалізації нового методу прогнозування.

Згідно огляду досліджених методів можна виокремити їх недоліки та використати це у розробці власного методу прогнозування, а саме необхідно досягти розвитку нечітких моделей недетермінованих часових послідовностей з

нерівномірним кроком спостережень шляхом формування нових комбінованих математичних моделей, які відрізняються використанням двох (кількох) узгоджених часових послідовностей даних, а також у розробці програмного забезпечення, призначеного для реалізації комбінованих нечітких реляційних моделей процесів з нерівномірним кроком спостережень. Для досягнення мети, розв'язання та опрацювання задач розробки були застосовані методи математичного моделювання та нечітких множин, методи FTS першого та вищих порядків, а також методи програмної інженерії щодо формування вимог, проєктування та розробки програмного забезпечення.

РОЗДІЛ 2. ПРОЄКТУВАННЯ

2.1. Задачі проєкту

На основі проведеного опису аналогів відомих методів, можна виділити особливості і недоліки, згідно з якими можна розробити новий математичний метод і його програмну реалізацію.

Необхідно передбачити при проектуванні нового методу:

– розробку нової комбінованої нечіткої реляційної математичної моделі процесів з нерівномірним кроком спостережень, яка поєднує дві перемінні послідовності – контрольований параметр та інтервал між спостереженнями, що дозволяє моделювати та прогнозувати процеси з нерівномірним кроком;

– розробку комбінованих нечітких реляційних моделей процесів першого та другого порядку, які на відміну від відомих враховують дані двох узгоджених процесів, що забезпечує аналіз та прогнозування процесів з нерівномірним кроком між інтервалами спостережень.

Необхідно передбачити для програмного продукту:

– формування відповідних комбінованих математичних моделей, які відрізняються використанням кількох часових послідовностей даних;

– формування графіків для методу трикутника;

– формування графіків для відображення рядів послідовності;

– прогнозування з урахуванням процесів з перемінним кроком спостережень.

2.2. Математична модель розробленого методу

Нечіткі реляційні моделі процесу визначаються системою реляційних відношень, отриманих на основі вихідної послідовності даних. яку узагальнено можливо представити так: R(V, T) = {R1(V, V); R2(T, T); R3(V*V, V); R4(T*V, T); R5(T*V, V); R6(T*T, V*V, V); R7(T*T, V*V, T)} [5; 8].

У відношенні R(V, T) параметр «V» відповідає певному контрольованому показнику процесу (наприклад, рівень цукру в крові), а «T» є показником часу. Відношення R6(*) та R7(*) нечітких реляційних моделей відповідають моделям

другого порядку, враховують кілька попередніх етапів розвитку процесів. Модель R(V, T) формується за первинними даними процесів, які перетворюються до табличних форм на основі нечітких моделей областей варіювання показників (V, T). Таблиці R(V, T) представляють нечіткі «правила» зв'язку між показниками (V, T) на послідовних кроках досліджуваного процесу. Кожна клітина таблиць містить два параметри – «номер нечіткого терму» що має бути на наступному кроці процесу, а також нечітка міра для очікуваного терму [8; 18].

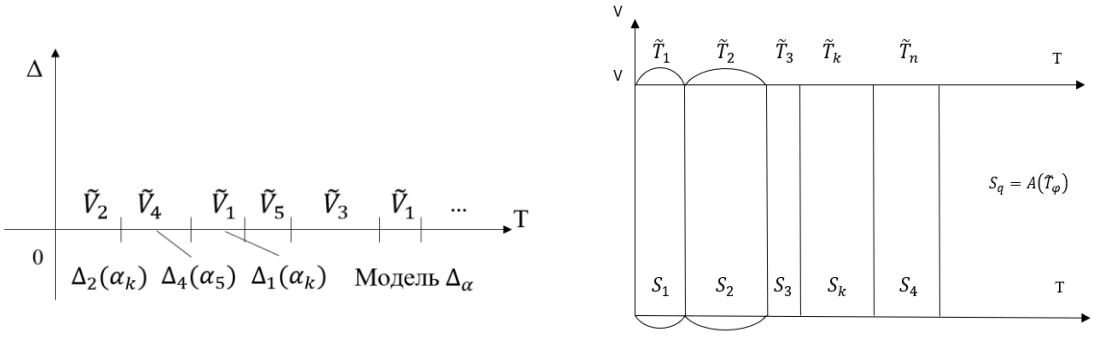
Загальна форма моделювання за реляційно-сепарабельним методом для показника V(*) на основі результатів сепарабельної V(Y) та нечіткої реляційної моделі V(Δ) процесу, отримана шляхом процедури експонентного згладжування, представлена у вигляді формули (2.1), де параметр «w» розраховується методом найменших квадратів [8]:

$$V(Y, \Delta, w) = w*V(Y) + (1-w)*V(\Delta), \quad 0 \leq w \leq 1. \qquad (2.1)$$

Представимо структуру моделювання за нечіткими реляційними моделями для процесів з одновимірними показниками. При цьому вихідні дані представляють числові послідовності пар величин рівнів «k» – («інтервал», «показник»)

$$(T, V)=\{(\Delta T(k), \Delta V(k))\}, k = 1, 2, \ldots, n. \qquad (2.2)$$

Величини (T, V) можуть бути дійсними або нечіткими, а інтервали нерівномірними. При моделюванні за нечіткими реляційними моделями передбачається перехід від довільних типів даних (2.2) до нечітких певного вигляду, трикутних [7;8].



(а) Первинні дані (б) Модель процесів

Рисунок 2.1 – Модель нечітких інтервалів та їх апроксимації

На рис. 2.1 приведені результати етапу попереднього моделювання даних (2.2) за допомогою системи трикутних величин, які далі застосовуються для формування нечітких відношень першого та другого порядку, що утворюють нечіткі реляційні множини. В результатів застосування апроксимацій S1, S2, S3, S4, S5., зображено на рис. 2.2 [8;10],

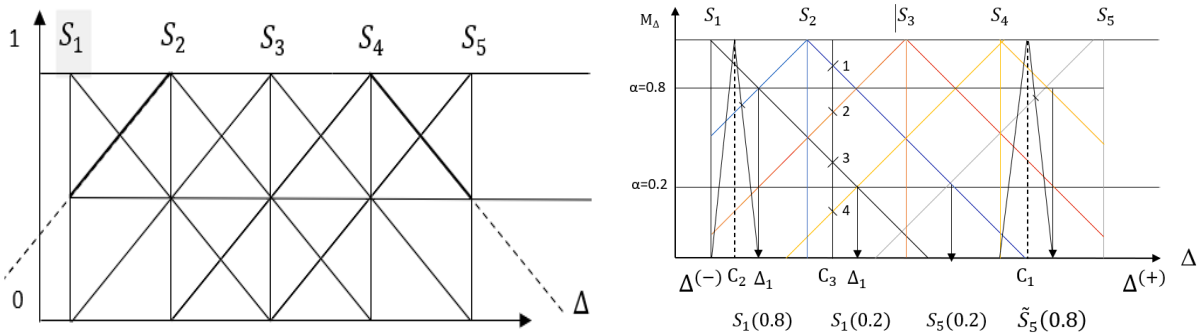


Рисунок 2.2 – Системи трикутних моделей для апроксимації «розмитої» області змін інтервалів між спостереженнями S_1, S_2, S_3, S_4, S_5

відображають область можливих значень показників (T, V), вхідним даним рівнів процесу приписують трійку ознак – («номер S_j » , ступінь приналежності «μ_j», код формули трикутної моделі (ліва (значення менше центру, права – більше чи рівні))). Для зменшення числа показників при зберіганні прийнята наступна умова – для значень зліва показник «μ_j» записують з мінусом « -μ_j», а для значень більших ніж центр «μ_j». Саме такі показники використовуються далі при формуванні відношень нечітких реляційних множин і в процедурах моделювання, аналізу та прогнозування [8; 9; 14; 18].

Головний зміст цього підходу формування реляційно-сепарабельних моделей полягає в тому, щоб використовувати не лише сепарабельну модель, а комбінувати її результати з нечіткими реляційними множинами. На відміну від сепарабельної моделі, в яких кожний параметр формується незалежно від інших, а вони узгоджуються на основі номерів спостережень (2.1), в нечітких реляційних множинах формується нечітка модель зв'язків між нечіткими кроками у часових інтервалах та змінами контрольованого параметру процесів.

Модель нечіткої реляційної множини змістовно відтворює нечітку регресію, яка дозволяє на основі поточних та попередніх значень (номери термів нечіткої величини Sj) характеристик процесу визначити, до яких нечітких величин процес «перейде» на наступному своєму кроці, етапі. То ж за попередніми нечіткими величинами (Sj) будуть відтворені терми, які визначають нечіткі величини що очікувані на наступному кроці процесу [8; 17].

Як зазначено, нечітка реляційна модель процесу являє систему реляційних відношень виду:

$$R(V, T) = \{RV(V, V); RT(T, T); RV1(V*V, V); RTV(T*V, T); RVT(T*V, V); RV2(T*T, V*V, V); RT2(T*T, V*V, T)\}. \quad (2.3)$$

Компоненти (2.3) RV(V, V), RT(T, T) не мають зв'язків з попередніми етапами, мають «нульовий» порядок, порядок інших відношень (2.3) визначається числом попередніх етапів, що враховуються моделлю [8;11].

Процедури формування та використання компонентів (2.3) подібні між собою. Сутність та призначення цієї математичної моделі полягає в представлення процесів перетворення нечітких послідовностей апроксимацій S1, S2, S3, S4, S5 для часових характеристик інтервалів (ПТ), а також таких характеристик для контрольованих показників процесів (ПР), у реляційну модель як нечіткого відношення. В якості однієї (вхідної) координати відношення виступає або період ПТ (заданий можливими нечіткими величинами), або показник процесів (ПР) (свої нечіткі величини). На перетині рядку та стовпця стоїть значення очікуваного нечіткого показника, як результат прогнозування наступного кроку [8; 15; 16].

Моделі нечітких відношень RTV(T*V, T); RVT(T*V, V) мають метою представлення зв'язків (ΔT_k, $\Delta V_k \rightarrow \Delta T_{k+1}$(I) також ($\Delta T_k$, $\Delta V_k \rightarrow \Delta V_{k+1}$(II)))

Структура реляційних таблиць подана на рис. 2.3, де $Z_i = \begin{cases} \tilde{T}_q, \text{ для моделі (I)} \\ \tilde{V}_r, \text{ для моделі (II)} \end{cases}$

Структури відношень представлені на рис.2.3 однакові, але значення Zi побудовані за даними для ΔT, модель (I) або за даними для ΔV, модель (II). У змішаних відношеннях RTV(T*V, T); RVT(T*V, V) значення прогнозованих показників визначається за рахунок перевірки всіх величин вихідних даних для {T,V}, що відповідають номерам рівня <k>, як моделі наступного кроку для показників процесу [8; 11; 18].

ΔT \ ΔV	$\tilde{V}_1$	$\tilde{V}_2$	$\tilde{V}_3$	$\tilde{V}_4$	$\tilde{V}_5$
T_1	Z_1	Z_5	Z_2	Z_5	Z_3
T_2	Z_3	..	..		
T_3					
T_4		..	..		
T_5	Z_3	Z_4	Z_1	Z_5	Z_2

Рисунок 2.3 – Структура із представлення змішаних відношень RTV(T*V, T); RVT(T*V, V) (період часу / показник)

Приведемо модель та процедуру аналізу та прогнозування на основі реляційного відношення другого порядку, за якими наступний елемент (очікувана на наступному кроці нечітка величина) визначається по двох попередніх як для параметру часу (T), так і для контрольованого параметру (V).

Змістовно модель реляційного відношення другого порядку формується таким чином, щоб відобразити вхідний процес, поданий через апроксимації S1, S2, S3, S4, S5, у вигляді табл. 2.1. А саме для всього процесу (подібно до формування таблиць рис. 2.3) послідовно визначають пари значень для {T,V}, через номери яких встановлюються рядки та стовпі табл. 2.1, в котрі необхідно

ввести інформацію про наступний крок процесу (окремо для процесу T і V номери апроксимацій S_j, значення ступенів приналежності ін.) [6; 7; 8].

При не можливості застосування реляційної моделі табл. 2.1, що потенційно можливо через, наприклад, відсутність в вихідній послідовності даних всіх передбачених табл.2.1 варіантів послідовностей (S_1, S_2, S_3, S_4, S_5), процедура рис. 2.3 використовується двічі. За ці два кроки наближено виконається урахування двох попередніх етапів процесу. При тому використовують сформовані таблиці безпосередньо для відношень першого порядку. Для реалізаціє моделі другого порядку використовують поточні значення вхідних (номер рядку, наприклад, i_1) та вихідних (номер стовпця, наприклад j_1) показників реляційних таблиць. При цьому на підставі значень (i_1, j_1), використовуючи таблиці для показників T, визначається номер рядку прогнозу, наприклад, i_2, а за таблицею вихідних характеристик (номер стовпця, наприклад j_2). Після цього на підставі значень (i_1, j_1) визначаються номери величин, що визначають прогноз відповідно до моделі другого порядку, наприклад (i_3, j_3). Саме ці значення (i_3, j_3) мали зберігатися в табл. 2.1 для моделі відношення 2-го порядку, (i_1, i_2, j_1, j_2) $\rightarrow$ (i_3, j_3) [8; 12; 16].

Таблиця 2.1 – Структура моделі реляційного відношення другого порядку

step	(n-1)	V_1	V_1	V_1	V_2	V_2	V_2	V_3	V_3	V_3	…
(n-1)	(n)	V_1	V_2	V_3	V_1	V_2	V_3	V_1	V_2	V_3	…
T_1	T_1	$r_{11,11}$									
T_1	T_2			$r_{13,12}$							
T_1	T_3							$r_{i1i2,j1j2}$			
T_2	T_1			$r_{21,13}$							
T_2	T_2										
T_2	T_3			$r_{i1i2,j1}$			$r_{23,23}$				
T_3	T_1										
T_3	T_2							$r_{32,31}$			
T_3	T_3										

Реляційні відношення другого порядку визначають зв’язки процесу

$R_v: \tilde{T}_{i1}, \tilde{T}_{i2}, \tilde{V}_{j1}, \tilde{V}_{j2} \rightarrow V_{iq} (q \in \{1,2,3\})$

$R_v: \tilde{T}_{i1}, \tilde{T}_{i2}, \tilde{V}_{j1}, \tilde{V}_{j2} \rightarrow T_{ip} (p \in \{1,2,3\})$ $\quad r_{i1i2,j1j2} = \begin{cases} T_i \text{ для } R_t(x) \\ V_j \text{ для } R_v(x) \end{cases}$ (2.4)

При розрахунках контрольованих показників нечітких реляційних моделей, представлених нечіткими відношеннями (2.3), (2.4) для дискретних таблиць використовується метод «центр ваги» (2.5) [13]

$$B(S_j)^* = \frac{\sum B_k * \mu_k}{\sum \mu_k}. \quad (2.5)$$

2.3. Моделювання методу

Для початку необхідно визначити пари значень, а саме перемінні послідовності – контрольований параметр та інтервал між спостереженнями. Наприклад, пари значення, які зображенні на рис. 2.4.

№	0	1	2	3	4	5	6	7	8
T	6	7	10	5	8	15	9	7	12
V	12	16	18	15	21	11	17	19	13

Рисунок 2.4 – Пари значень

Після цього, кожне число необхідно перетворити у нечітку реляційну множину. Для цього можна застосувати метод трикутників, який для значень T має наступний вигляд і зображений на рис. 2.5.

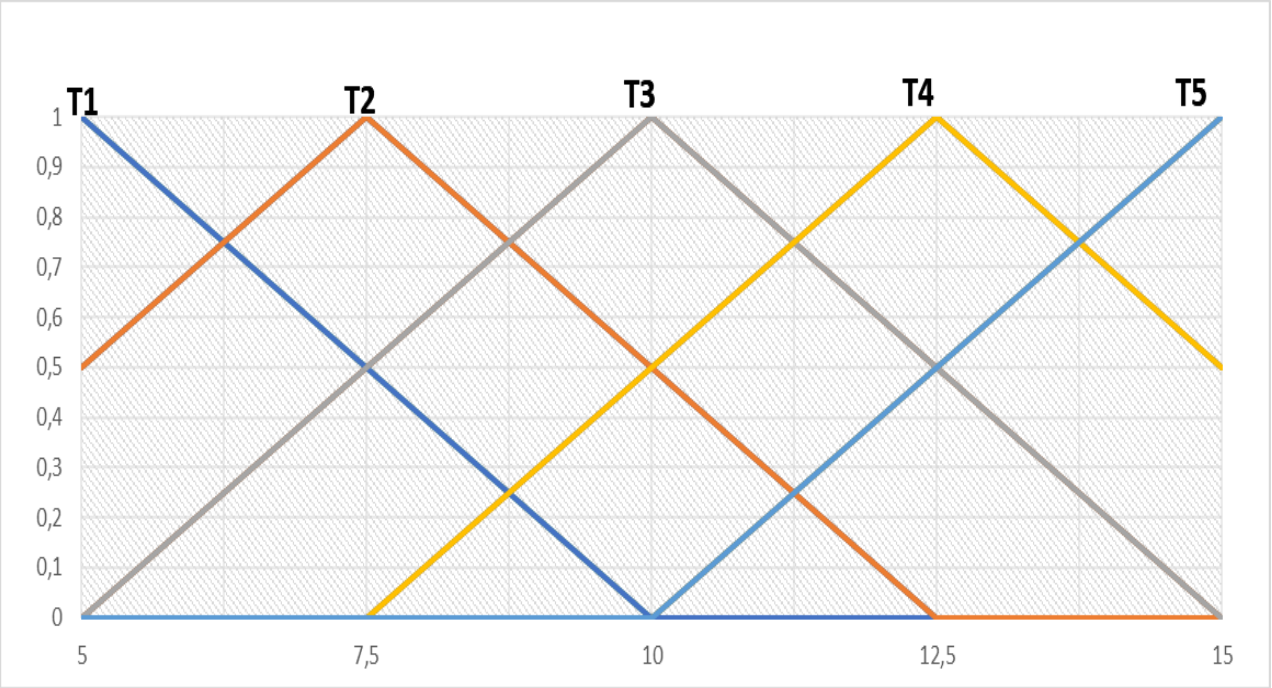


Рисунок 2.5 – Графік методу трикутників для T

Для значень V він має такий самий вигляд, але його границі інші, зображено на рис. 2.6.

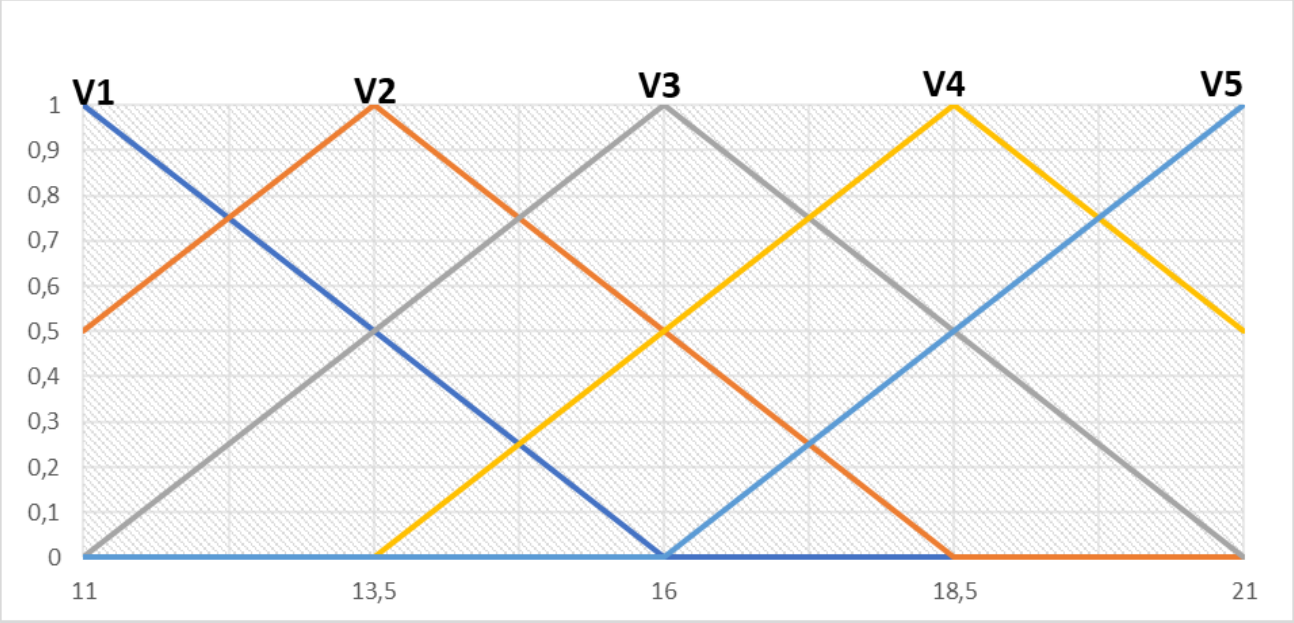


Рисунок 2.6 – Графік методу трикутників для V

Згідно методу трикутників, необхідно провести кожне значення через відповідний графік та знайти їх приналежність до границь із відповідним коефіцієнтом.

Результат представлено на рис. 2.7.

$\widetilde{T}$:	(T1; 0,8)	(T2; 0,9)	(T3; 1)	(T1; 1)	(T2; 0,9)	(T5; 1)	(T3; 0,8)	(T2; 0,9)	(T4; 0,9)
№	0	1	2	3	4	5	6	7	8
$\tilde{v}$	(V1; 0,8)	(V3; 1)	(V4; 0,9)	(V3; 0,8)	(V5; 1)	(V1; 1)	(V3; 0,8)	(V4; 0,9)	(V2; 0,9)

Рисунок 2.7 – Нечіткі реляційні множення

Після цього необхідно побудувати ряди спостережень відповідно до рис. 2.7 та відобразити усі переходи по графіку. Ряд спостережень для значень T зображено на рис. 2.8., а ряд спостережень для значень V зображено на рис. 2.9. відповідно.

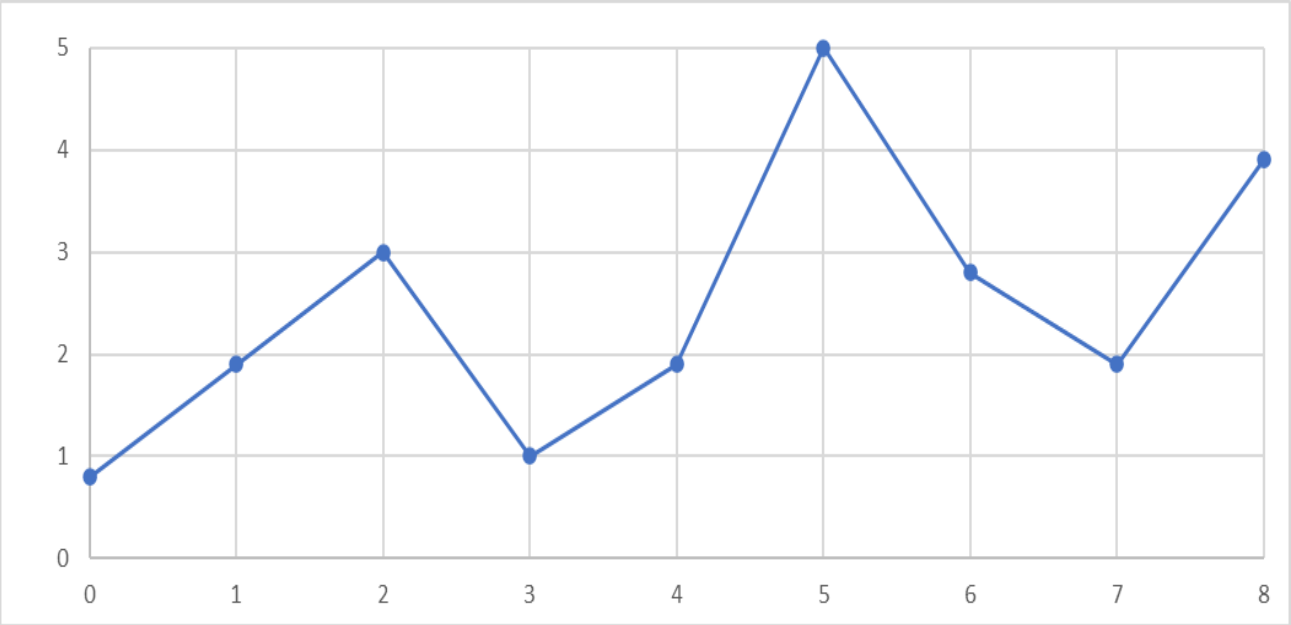


Рисунок 2.8 – Ряд спостережень T

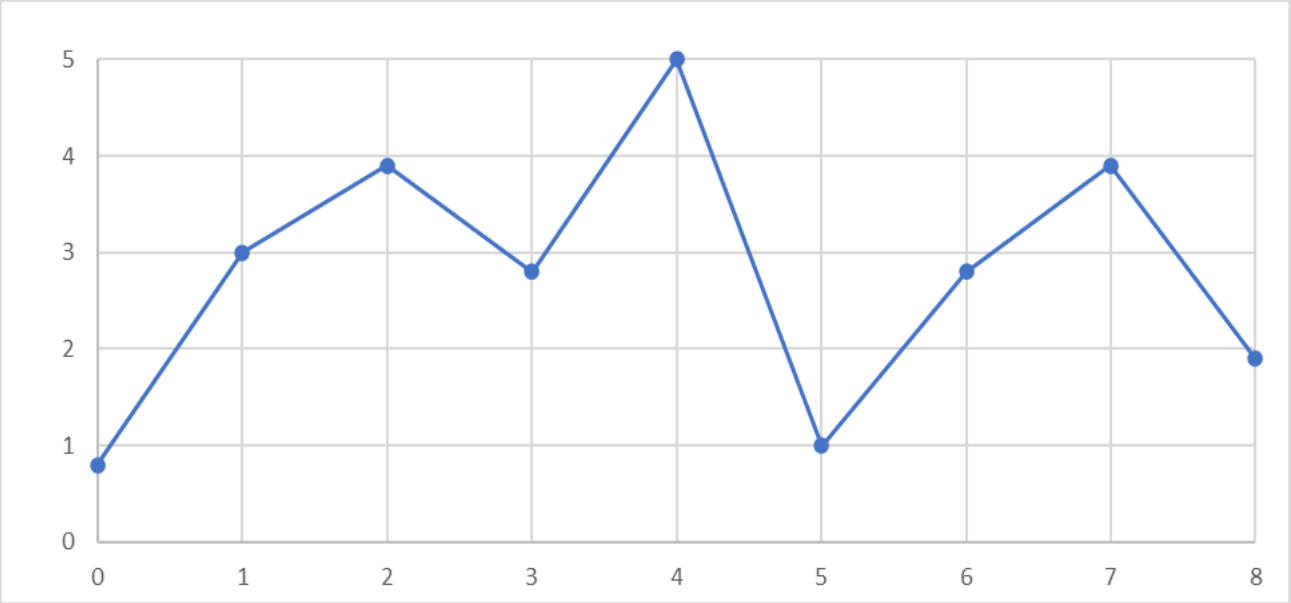


Рисунок 2.9 – Ряд спостережень V

Згідно до побудованих рядів спостережень можна визначити матриці переходів. Для значень T матрицю переходів зображено на рис. 2.10., а для значень V матрицю переходів зображено на рис. 2.11.

Ti\Vi	V1	V2	V3	V4	V5
T1	T2(0.9)	0	T2(0.9)	0	0
T2	0	0	T3(1)	T4(0.9)	T5(1)
T3	0	0	T2(0.9)	T1(1)	0
T4	0	0	0	0	0
T5	T3(0.8)	0	0	0	0

Рисунок 2.10 – Матриця переходів для T

Vi\Ti	T1	T2	T3	T4	T5
V1	V3(1)	0	0	0	V3(0.8)
V2	0	0	0	0	0
V3	V5(1)	V4(0.8)	V4(0.9)	0	0
V4	0	V2(0.9)	V3(0.8)	0	0
V5	0	V1(1)	0	0	0

Рисунок 2.11 – Матриця переходів для V

Після цього будуються матриці приналежності, відповідно до кожної точки T та V, приклад такої матриці зображено на рис. 2. 12.

R(T2 V3)					
T2\V3	0	0.5	1	0.5	0
0.5	0	0.5	0.5	0.5	0
1	0	0.5	1	0.5	0
0.5	0	0.5	0.5	0.5	0
0	0	0	0	0	0
0	0	0	0	0	0

Рисунок 2.12 – Матриця переходів для V

Наступним кроком буде розрахунок і прогнозування нової пари значень із урахування матриць переходів та часового ряду.

Приклад підрахунку зображено на рис. 2.13.

$$V` = \begin{bmatrix} 0*0+0*0+0*0+0*0+0*0 \\ 0*0+0*0+0.5*0.5+0.5*0.5+0.5*0.5 \\ 0*0+0*0+0.5*1+1*1+0.5*1 \\ 0*0+0*0+0.5*0.5+0.5*0.5+0.5*0.5 \\ 0*0+0*0+0*0+0*0+0*0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 1 \\ 0.5 \end{bmatrix}$$

Рисунок 2.13 – Приклад розрахунку

Порівняльні відмінності між Fuzzy Time Series та Combined Fuzzy Time Series полягають у тому, що для першого методу будуються окремі моделі для T та V, а розроблений метод комбінує ці процеси, тобто робить їх залежними один від одного. Наприклад, для рядів спостережень T (див. рис. 2.8), використовуючи метод Fuzzy Time Series було б записано так:

I. T1 –> T2; T2 –> T3; T3 –> T1; T1 –> T2; T2 –> T5; T5 –>T3; T3 –>T2; T2 –> T4;

II. T1 –> T2; T2 –> T3 V T5 V T4; T3 –> T1 V T2; T5 –> T3.

I для рядів спостережень V (див. рис. 2.9) відповідно було б так:

I. V1 –> V3; V3 –> V4; V4 –> V3; V3 –> V5; V5 –> V1; V1 –> V3; V3 –> V4; V4 –> V2;

II. V1 –> V3; V3 –> V4 V V5; V4 –> V3 V V2; V5 –> V1.

А для нового методу Combined Fuzzy Time Series ці моделі виглядають так: (T1; V1) –> (T2; V3), (T2; V3) –> (T3; V4), (T3; V4) –> (T1; V3), (T1; V3) –> (T2; V5), (T2; V5) –> (T5; V1), (T5; V1) –> (T3; V3), (T3; V3) –> (T2; V4), (T2; V2) –> (T4; V2)

2.4. Функціональні вимоги

Програмний продукт повинний забезпечити введення користувачем кількості значень для формування таблиці значень, введення часових послідовностей даних.

Аналізуючи введені користувачем початкові дані, програмний продукт має генерувати і відображати:

– графік для методу трикутника із відповідними границями до введених послідовностей даних;

– таблицю із фазифікованими значеннями (нечіткими множинами);

– графіки відображення дослідження рядів послідовностей;

– таблицю переходів першого порядку;

– таблицю із прогнозованими значеннями.

Діаграма прецедентів зображена на рис. 2.14.

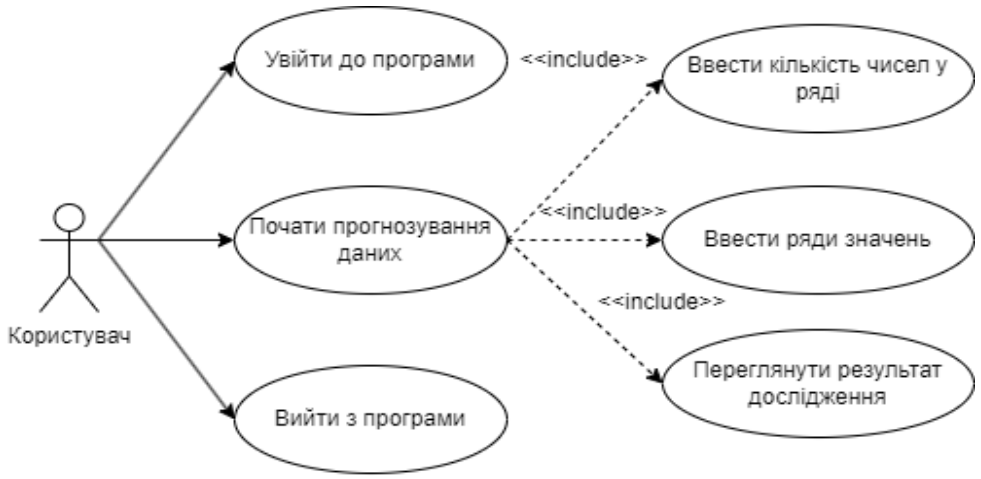


Рисунок 2.14 – Діаграма прецедентів

2.5. Вхідні та вихідні дані

Вхідними даними програмного додатку є:

– введення кількості чисел у послідовності;

– введення послідовності чисел.

Вихідними даними програмного додатку є:

– графіки методу трикутника;

– графіки рядів спостереження;

– таблиця нечітких реляційних значень;

– таблиця переходів першого порядку;

– прогнозовані значення.

2.6. Метод та засоби програмування

Для розробки програми під платформу Windows був використаний стек технологій, який включив в себе мову C# для програмування, середовище програмування – Visual Studio для розробки та налагодження, LiveCharts для візуалізації даних, Windows Presentation Foundation для створення графічного інтерфейсу користувача. Цей стек технологій допоміг створити функціональну та естетично привабливу програму для платформи Windows.

Visual Studio – це інтегроване середовище розробки, розроблене компанією Microsoft, яке надає розробникам широкий спектр інструментів для створення програмного забезпечення. Він підтримує різні мови програмування, такі як C#, C++, Visual Basic .NET, F# та інші, і дозволяє розробляти додатки для різних платформ, включаючи Windows, Android, iOS, macOS, Linux і хмарні сервіси.

Ключовими характеристики Visual Studio є:

– кодування та редактор: Visual Studio надає потужний редактор коду з відмінною підсвіткою синтаксису, автоматичним завершенням коду, перевіркою помилок та розширеними можливостями форматування. Він також підтримує версійний контроль коду і дозволяє співпрацювати з іншими розробниками;

– дебагер: Visual Studio має потужні інструменти для дебагування, які допомагають виявляти та усувати помилки в програмах. Розробники можуть

крокувати через код, встановлювати точки зупину, аналізувати значення змінних та використовувати інші інструменти для відлагодження програм;

– дизайнер інтерфейсу: Visual Studio має вбудовані дизайнери інтерфейсу, такі як Windows Forms Designer та XAML Designer, які дозволяють візуально створювати користувацький інтерфейс додатків. Розробники можуть перетягувати та налаштовувати елементи керування, задавати властивості та розміщення елементів UI безпосередньо в середовищі Visual Studio;

– інструменти для тестування: Visual Studio має вбудовані інструменти для автоматичного тестування, які дозволяють розробникам створювати, виконувати та аналізувати тести на різних рівнях, включаючи модульні тести, інтеграційні тести та UI-тести;

– інтеграція з хмарними сервісами: Visual Studio має підтримку різних хмарних сервісів, зокрема Microsoft Azure. Розробники можуть легко підключатися до хмарних ресурсів, керувати ними та розгортати свої додатки в хмарному середовищі;

– розширюваність: Visual Studio дозволяє розробникам створювати власні розширення, додавати нові функціональність, інструменти та шаблони проектів. Є широкий вибір розширень, створених спільнотою розробників, які полегшують роботу та забезпечують налаштування середовища розробки під конкретні потреби.

Visual Studio є одним з найпопулярніших інструментів розробки програмного забезпечення, і його потужність і гнучкість роблять його відмінним вибором для професійних розробників. Можна відзначити, чому Visual Studio може бути вигідним вибором для розробки програми. Середовище має вбудовану підтримку для розробки програм під платформу .NET Framework та .NET Core. Це дозволяє розробникам створювати різноманітні типи додатків, включаючи веб-додатки, настільні додатки, мобільні додатки, служби хмарних обчислень та багато інших, використовуючи потужні бібліотеки та функціональність .NET. Visual Studio має широку спільноту розробників, що активно підтримується Microsoft. Це означає,

що можна знайти багато ресурсів, документацію, статей та форумів, де можна отримати допомогу, поради та вирішити проблеми, з якими можна зіткнутись під час розробки [20].

Visual Studio 2022 є найновішою версією інтегрованого середовища розробки від компанії Microsoft. Вона є наступником Visual Studio 2019 і включає ряд нововведень та поліпшень. Ось деякі ключові риси Visual Studio 2022, які є ключовими для вибором саме цієї версії для розробки програми:

– вдосконалене продуктивне середовище: Visual Studio 2022 надає збільшену продуктивність та швидкість роботи завдяки оптимізаціям та оновленням. Відкриття проектів, завантаження редактора та виконання операцій стали швидшими та ефективнішими;

– покращена підтримка різних платформ: Visual Studio 2022 дозволяє розробляти програми для різних платформ, включаючи Windows, Android, iOS, macOS і Linux. Вона включає в себе оновлену версію .NET, яка підтримує .NET 6, що дає можливість розробляти кросплатформенні додатки;

– удосконалені інструменти для розробки: Visual Studio 2022 має ряд поліпшених інструментів для розробки, таких як розширений редактор коду, дебагер, автоматичне доповнення коду, інструменти тестування та інше. Це полегшує процес розробки та покращує досвід програміста;

– інтеграція з хмарними сервісами: Visual Studio 2022 надає розширену підтримку хмарних сервісів Microsoft Azure. Це дозволяє розробникам легко підключати та взаємодіяти з різними хмарними послугами, такими як облікові записи, бази даних, розгортання та багато інших;

– модульність та розширюваність: Visual Studio 2022 підтримує модульну архітектуру, що дозволяє розширювати функціональність за допомогою різних розширень. Розробники можуть створювати власні розширення, додавати нові функції та інструменти, що відповідають їхнім потребам [20].

Visual Studio 2022 є потужним інструментом для розробки програмного забезпечення різного типу. Вона надає широкий набір функціональностей та

інструментів для полегшення процесу розробки та забезпечення високої продуктивності розробника.

C# є мовою програмування, яку розробила компанія Microsoft. Вона була представлена в 2000 році як частина ініціативи Microsoft для розробки програмного забезпечення на платформі .NET Framework. C# є однією з найпопулярніших мов програмування та має широке застосування в розробці різноманітних додатків [19].

Серед основних характеристик мови програмування C# можна відзначити наступні:

– синтаксис: C# має синтаксис, схожий на інші мови програмування, такі як C, C++ та Java. Це дозволяє розробникам, які знайомі з цими мовами, легко освоїти C#;

– об’єктно-орієнтоване програмування: C# підтримує об'єктно-орієнтоване програмування. Це означає, що програми розробляються шляхом створення класів, об'єктів, спадковості, інкапсуляції та поліморфізму;

– керування пам’яттю: у C# використовується система збирання сміття, що відповідає за автоматичне управління пам’яттю. Це дозволяє розробникам уникати проблем, пов’язаних з ручним виділенням та звільненням пам’яті;

– розширення .NET Framework: C# інтегрується з .NET Framework, великою платформою розробки програмного забезпечення. Це дає доступ до багатьох потужних бібліотек та функцій для розробки різноманітних додатків, включаючи веб-додатки, настільні програми, мобільні додатки та ігри;

– мультиплатформеність: З’явившись як мова для .NET Framework, C# тепер підтримується іншими платформами, такими як .NET Core і Xamarin. Це дозволяє розробляти кросплатформенні додатки, що працюють на різних операційних системах, таких як Windows, macOS і Linux;

– інтеграція з Microsoft-технологіями: C# щільно пов’язана з екосистемою Microsoft. Вона має вбудовану підтримку для розробки додатків для Windows, Microsoft Azure, Office, SQL Server та багатьох інших продуктів Microsoft;

– розширюваність: C# дозволяє розробникам створювати власні бібліотеки, компоненти та фреймворки, що дозволяє використовувати код знову та полегшує розробку масштабованих додатків.

C# має багато інших функцій та можливостей, які роблять її потужним інструментом для розробки різноманітних програмних додатків [19].

.NET Core – це вільна та відкрита платформа, розроблена компанією Microsoft. Вона є модульною, кросплатформеною версією .NET Framework і призначена для створення сучасних застосунків, які працюють на різних операційних системах, таких як Windows, macOS і Linux [20].

Ключові риси .NET Core:

– кросплатформеність: одна з основних переваг .NET Core полягає в його кросплатформеній підтримці. За допомогою .NET Core ви можете розробляти додатки, які працюють на різних операційних системах, не залежно від того, чи це Windows, macOS або Linux. Це дозволяє вам створювати кросплатформенні рішення, які можуть бути використані на різних пристроях;

– висока продуктивність: .NET Core пропонує високу продуктивність і швидкодію завдяки своїй оптимізації та новим функціям. Він має покращений збирач сміття, вбудовану підтримку асинхронного програмування та оптимізовані бібліотеки, що дозволяють виконувати завдання швидше і ефективніше;

– модульність: .NET Core побудований на принципах модульності, що дозволяє вам використовувати тільки необхідні компоненти для вашого додатка. Можна використовувати тільки ті бібліотеки та компоненти, які потрібні для вашого проекту, що зменшує об'єм виконуваного коду та сприяє покращенню продуктивності;

– велика екосистема: .NET Core має широку екосистему бібліотек, фреймворків та інструментів розробки, що полегшують створення різноманітних додатків. Можна використовувати бібліотеки класів .NET Framework, а також багато інших сторонніх бібліотек, що розширюють можливості вашого додатка;

– підтримка мікросервісної архітектури: .NET Core добре підходить для розробки мікросервісних застосунків. Він має вбудовану підтримку для контейнеризації та можливість використовувати масштабовану архітектуру, де окремі компоненти додатка можуть бути розгорнуті та масштабовані незалежно один від одного;

– отримання оновлень: однією з переваг .NET Core є можливість отримувати оновлення та виправлення безпеки безпосередньо від Microsoft. Це дозволяє підтримувати додатки актуальними та безпечними з мінімальними зусиллями.

.NET Core є потужним інструментом для розробки сучасних кросплатформених додатків. Він дозволяє розробникам створювати швидкі, масштабовані та продуктивні рішення для різних платформ і сценаріїв.

Windows Presentation Foundation (WPF) є одним з фреймворків розробки користувацького інтерфейсу в середовищі .NET. Він був розроблений компанією Microsoft і вперше випущений разом з .NET Framework 3.0 [21].

Ключові характеристики фреймворку:

– декларативний синтаксис: У фреймворку використовує декларативний синтаксис для опису інтерфейсу користувача. Замість традиційного програмного коду, можна використовувати мову розмітки XAML (eXtensible Application Markup Language) для визначення вигляду та поведінки елементів користувацького інтерфейсу. Це робить розробку інтерфейсу більш зрозумілою та зручною;

– розділення логіки та представлення: WPF використовує шаблон проектування Model-View-ViewModel (MVVM), що дозволяє розділити логіку додатка від представлення. Це полегшує тестування, підтримку та розширення додатків;

– багатофункціональність: WPF надає широкий набір вбудованих елементів керування, таких як кнопки, тексти, списки, таблиці і багато інших. Можна створювати власні елементи керування, що дає можливість налаштувати вигляд та поведінку інтерфейсу на власний розсуд;

– розширені можливості стилізації: WPF має потужну систему стилів та шаблонів, яка дозволяє змінювати вигляд елементів інтерфейсу за допомогою CSS-подібного підходу. Можна використовувати стилі, тригери, анімацію та інші техніки для створення привабливого та динамічного інтерфейсу;

– векторна графіка: WPF використовує векторну графіку для відображення елементів UI, що дозволяє створювати високоякісні графічні об’єкти. Це означає, що інтерфейс буде зберігати високу якість та розмір не залежно від роздільної здатності екрану;

– інтеграція з .NET Framework: WPF має повну інтеграцію з .NET Framework, що дозволяє використовувати всю могутність мови програмування C# або іншої мови .NET для розробки бізнес-логіки додатка.

WPF надає потужні можливості для розробки багатофункціональних та привабливих інтерфейсів користувача для Windows додатків. Він є вибором багатьох розробників для створення сучасних програмних продуктів з високоякісним користувацьким інтерфейсом.

LiveCharts є бібліотекою для створення інтерактивних графіків та діаграм в програмах на мові програмування C#. Вона надає зручні інструменти для візуалізації даних та динамічного оновлення графіків у реальному часі. Основні особливості LiveCharts включають [22]:

– простота використання: LiveCharts має простий та зрозумілий API, що дозволяє легко створювати графіки та діаграми без глибоких знань графічного програмування. Вона пропонує велику кількість вбудованих типів графіків, таких як лінійні графіки, стовпчикові діаграми та багато інших;

– інтерактивність: LiveCharts дозволяє створювати інтерактивні графіки, які реагують на дії користувача. Можна додавати взаємодію, таку як наведення, вибір елементів графіку, масштабування та прокручування. Це дозволяє користувачам взаємодіяти з даними та отримувати більше контексту про їх значення;

– підтримка даних в реальному часі: LiveCharts дозволяє оновлювати графіки в реальному часі, відображаючи зміни даних без перезавантаження всього графіку. Це корисно для моніторингу даних у реальному часі або створення анімаційних ефектів;

– підтримка Model-View-ViewModel (MVVM): LiveCharts інтегрується з популярним шаблоном проектування MVVM, що дозволяє розділити логіку даних від представлення графіків. Це полегшує тестування, розширення та управління даними;

– підтримка Windows Presentation Foundation (WPF) та WinForms: LiveCharts можна використовувати як в програмах, розроблених на платформі WPF, так і в програмах, розроблених на платформі Windows Forms. Вона пропонує спеціальні компоненти для кожної платформи.

LiveCharts є відкритим джерелом і доступна через пакети NuGet. Ця бібліотека надає зручний спосіб візуалізувати дані та створити інтерактивні графіки у програмах на C# [22].

2.7. Проектування форм програмного продукту

Для програмного продукту було спроектовано головну форму, яка повинна містити усі необхідні компоненти для вирішення поставленої задачі згідно реалізації методу. Ескіз головної форми зображено на рис. 2.15.

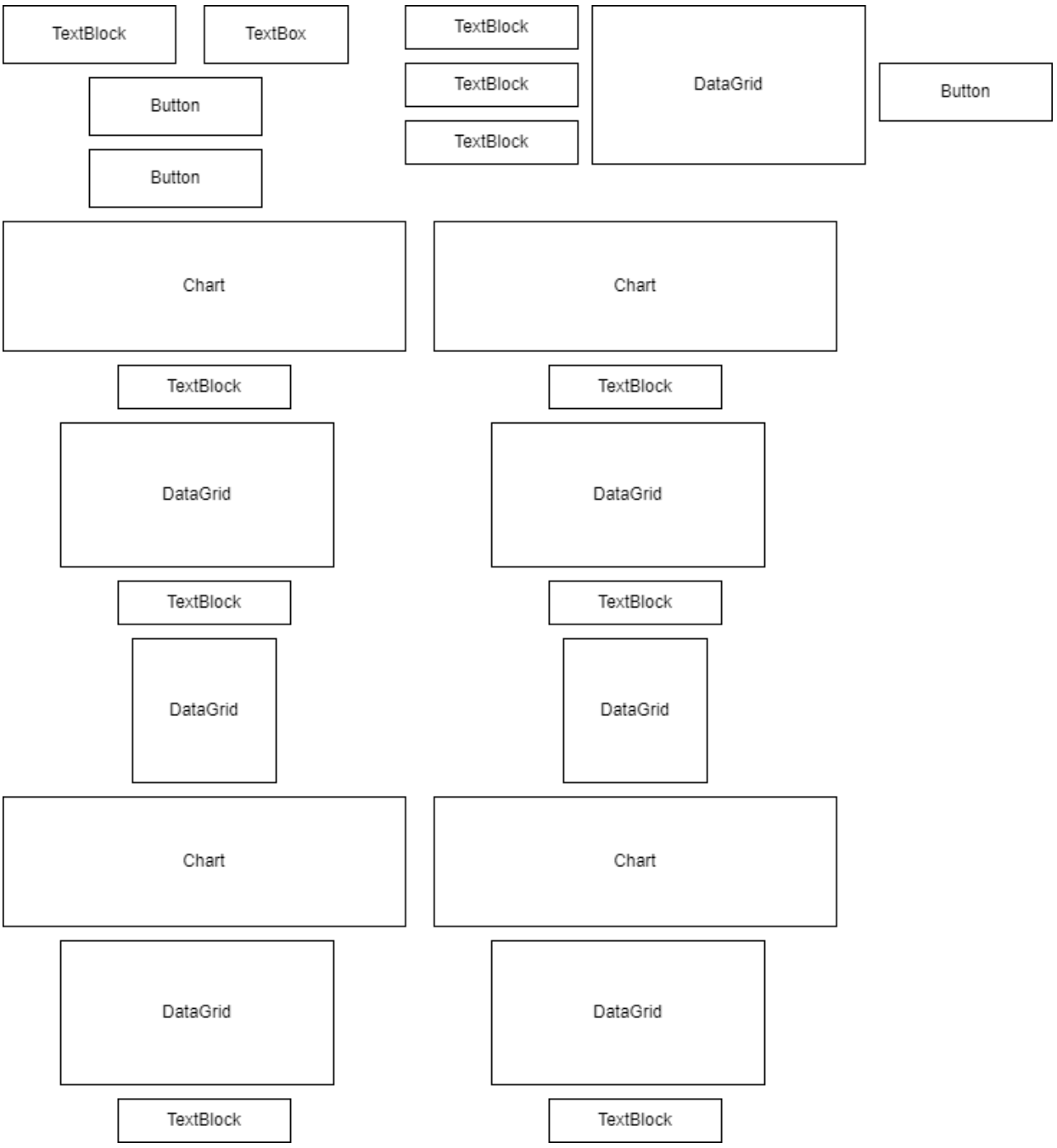


Рисунок 2.15 – Ескіз головної форми

2.8. Проектування інтерфейсу користувача

На головній формі повинні розміщуватись наступні компоненти:

– ScrollViewer. Цей компонент використовується для надання прокрутки вмісту, якщо його розміри перевищують доступний простір;

– TextBlock. Цей компонент використовується для відображення текстового вмісту без можливості редагування. У програмі містить назви відповідних таблиць;

– TextBox. Це текстове поле, в яке користувач може вводити та редагувати текст. Його можна використовувати для отримання введення від користувача або відображення текстового вмісту. У програмі необхідний для того, щоб користувач ввів кількість пар;

– Button. Використовується для створення кнопок, на які користувач може натискати для виклику певних подій або виконання дій. Кнопка може містити текст або зображення і призначатися для виконання певної функції при натисканні. У програмі необхідна для запуску подій прогнозування та підтвердження вводу;

– DataGrid. Цей компонент використовується для відображення табличних даних у форматі сітки. Він дозволяє відображати дані в стовпцях і рядках, підтримує сортування, фільтрацію і редагування даних. На формі необхідний для введення початкових пар значень та відображення нечітких реляційних множень, матриць переходів та нечіткого результуючого значення;

– CartesianChart. Компонент є графічним елементом, який використовується для візуалізації даних у вигляді графіків на координатній площині. Він може бути використаний для відображення графіків, діаграм, залежностей і багатьох інших видів даних. У програмі застосовується для того, щоб відобразити графічно метод трикутників для кожного із графіків та ряди переходів.

Висновки до розділу

Аналіз публікації та загальновідомих програмних засобів не дозволив виявити загально прийнятих і доступних програмних засобів реалізації завдань моделювання та дослідження даних з нерівномірним кроком спостережень, представлених у формі сукупності часових послідовностей.

Отримано розвиток нечітких моделей недетермінованих часових послідовностей шляхом формування відповідних комбінованих математичних

моделей, які відрізняються використанням двох (кількох) часових послідовностей даних, що дозволяє одночасно досліджувати кілька узгоджених процесів, в тому числі з нерівномірним кроком спостережень.

Також згідно поставленої задачі було знайдено середовище та мову програмування, які найкраще будуть застосовані для реалізації програмного продукту. У процесі проектування було описано форму та спроектовано її прототип і визначено компоненти, які будуть найкраще підходити згідно реалізації.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМИ

3.1. Розробка форми програмного продукту

Розроблена форма, з урахуванням опису проектування містить такі компоненти:

– 1 кнопка для підтвердження даних;

– 1 кнопка для завантаження прикладу;

– 1 кнопка для початку прогнозування;

– 2 графіки для відображення методу трикутника;

– 2 графіки для відображення рядів спостережень;

– 1 поле введення кількості значень;

– 1 таблиця для введення початкових даних;

– 2 таблиці, які містить нечіткі реляційні множини;

– 2 таблиці, які відображають матриці переходів;

– 2 таблиці із результуючими нечіткими множинами;

– 7 текстових позначень;

– 2 текстових позначення із дефазифікованими значеннями.

3.2. Розробка інтерфейсу програмного продукту

Інтерфейс головної форми зображено на рис. 3.1.

3.3. Розробка класів

Розробка класів є важливою складовою процесу програмування об'єктно-орієнтованого підходу. Класи визначають структуру та поведінку об'єктів, які можуть бути створені на основі цих класів.

У програмі передбачено такі класи, як:

– DataGridHelper: допоміжний клас, який містить функції по обробці компонента dataGrid:

– StatisticsCalculator: допоміжний клас, який містить функції обробки матриць та масивів;

– CharData: клас, який необхідний для виконання дій із графіками;

– MainWindow: головний клас, містить події форми та функції формування таблиць тощо;

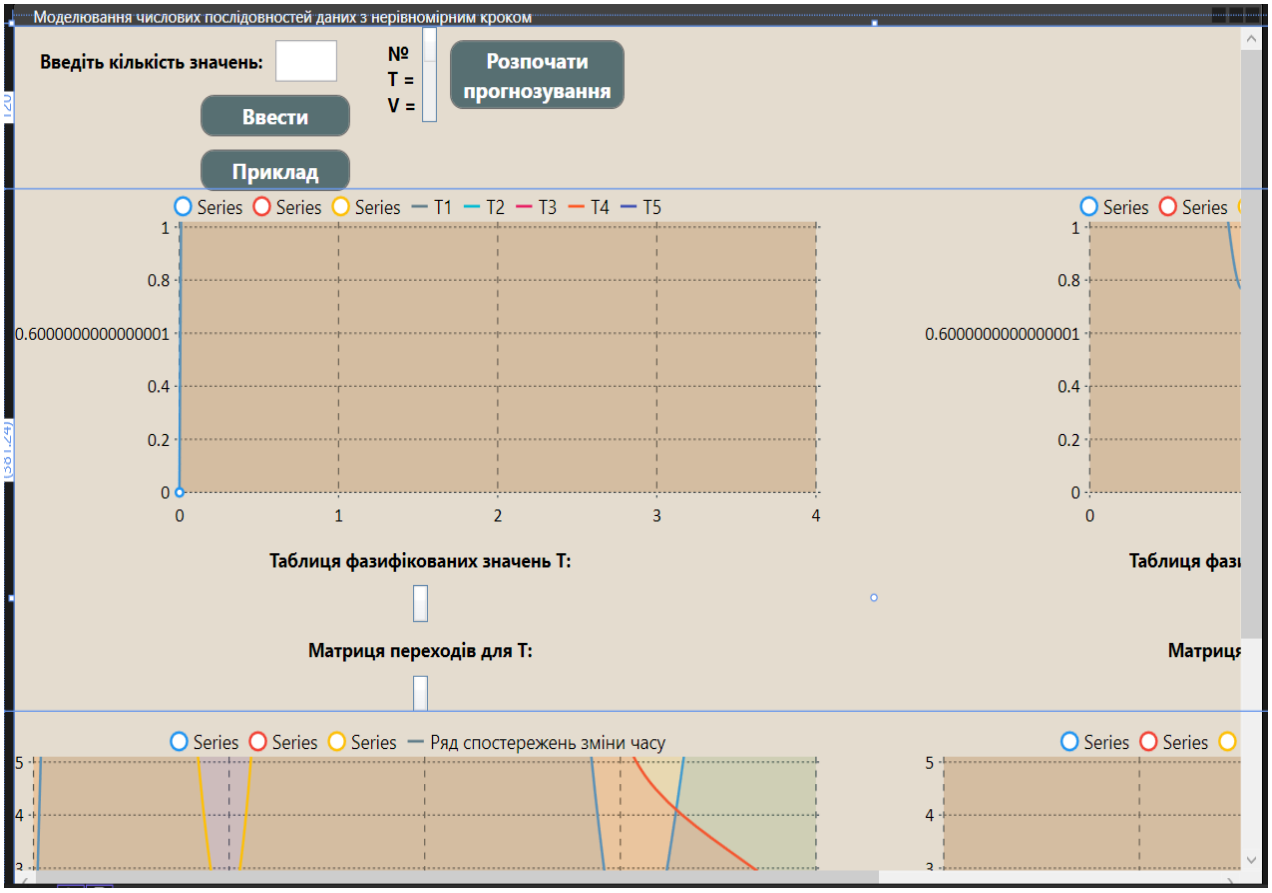


Рисунок 3.1 – Інтерфейс головної форми

Прототипи функцій, які містить клас DataGridHelper:

– void CreateDataGridColumns(int, DataGrid, DataTable): функція для створення стовпців у компонента dataGrid;

– void CreateDataGridRows(DataTable, int): функція для створення рядків у компонента dataGrid;

– List<int> GetDataGridItems(int, DataGrid): функція, яка повертає вміст компонента dataGrid;

– void CreateMatrixStep(DataGrid, string, string): функція, яка створює і заповнює певний dataGrid, який необхідний для відображення матриці переходів;

– void UpdateDataGrid(DataGrid, List<Tuple<int, double>>, string, int): функція для оновлення елементів компонента dataGrid;

– void CreateColumnsForArray(DataGrid, int, double[]): функція для створення результуючої таблиці на основі одномірного масиву;

– void UpdateMatrixStep(DataGrid, List<Tuple<int, double>>, List<Tuple<int, double>>, string): функція для оновлення таблиці із матрицею переходів;

– DataGridCell GetCell(DataGrid, DataGridCellInfo): функція для отримання клітинок з таблиці.

Прототипи функцій, які містить клас StatisticsCalculator:

– double[,] GetCellValues(ItemCollection, int, int): функція для вилучення числових значень із колекції та збереження цих значень у двовимірному масиві для подальшої обробки чи аналізу;

– double[] CalculateMinValues(double[,], int, int): функція для знаходження мінімального значення в двовимірному масиві;

– double[] CalculateMaxValues(double[,], int, int): функція для знаходження максимального значення в двовимірному масиві;

– double[] CalculateAverageBetweenValues(double[], double[], int): функція для знаходження середнього між двома елементами;

– List<Tuple<int, double>> FuzzificationAllData(double, double, double, double, double, List<int>): функція для виконання фазифікації над початковими даними та приведення значення до певної границі;

– double[] Fuzzification(double, double, double, double, double, double): функція фазифікації над числом, та приведення його відношення до кожної із границь;

– double CalculateMembershipValue(double, double, double, double): функція, яка необхідна для фазифікації, повертає фазифіковане число;

– void UpdateDesiredArrays(double[,], double[,], double[], double[]): функція для оновлення матриць.

Прототипи функцій, які містить клас CharData:

– void OnPropertyChanged(string): метод для створення події PropertyChanged.

Прототипи функцій, які містить клас MainWindow:

– double[,,] CreateTables(): функція, яка відповідає за обробку та створення таблиць;

– double[,] GetTwoDimensionalArray(int): функція для побудови двомірного масиву;

– void CreateColumns_Click(object, RoutedEventArgs): функція, яка відповідає за подію натискання на кнопку;

– void btnTest_Click(object, RoutedEventArgs): функція, яка відповідає за подію натискання на кнопку;

– void dataGrid1_PreviewTextInput(object, TextCompositionEventArgs): функція, яка відповідає за обмеження вводу у компонент dataGrid;

– void CalculateStatistics(out double[], out double[], out double[], out double[], out double[]): функція, яка відповідає за визначення границь значень, які необхідні для методу трикутника;

– void UpdateChartData(double[], double[], double[], double[], double[]): функція відповідає за оновлення графіків;

– void ShowDataGrids(): функція відповідає за відображення раніше прихованих таблиць;

– void UpdateChartLineData(List<Tuple<int, double>>, List<Tuple<int, double>>): функція відповідає за оновлення підписів на графіках;

– void btnCalc_Click(object, RoutedEventArgs): функція, яка відповідає за подію натискання на кнопку;

– double[] GetRowFromArray(double[,], int): функція, яка повертає рядок із двомірного масиву.

Висновки до розділу

Після опису аналогів та розділу проектування, було обрано необхідні компоненти та розроблено головну форму згідно поставленої задачі.

У процесі розробки було розроблено програмний продукт, який вирішує поставлену задачі із використанням нового математичного методу.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

4.1. Аналіз методів тестування та відлагодження

Тестування є важливою складовою процесу розробки програмного забезпечення і інших продуктів. Основна мета тестування полягає у тому, щоб перевірити, чи відповідає продукт заданим вимогам, функціональності та якості. Тестування допомагає виявити помилки, дефекти та проблеми у програмному забезпеченні або продукті.

Виявлення й усунення помилок є важливим етапом, оскільки це дозволяє забезпечити високу якість та надійність продукту. Завдяки тестуванню можна переконатися, що програмне забезпечення або продукт працює відповідно до очікувань і виконує всі функції, для яких воно призначене. Це дозволяє забезпечити задоволення користувачів та уникнути негативного впливу на бізнес.

Також воно допомагає забезпечити високу якість продукту, та дозволяє перевірити, чи відповідає продукт стандартам якості, включаючи ефективність, безпеку, надійність та зручність використання.

Виявлення проблем у ранній стадії – це одна із переваг тестування, чим раніше виявляються проблеми або помилки у процесі розробки, тим легше їх усунути. Тестування в ранній стадії дозволяє виявляти та виправляти проблеми до виходу продукту на ринок, що зберігає час і ресурси. Якісне тестування сприяє підвищенню довіри користувачів до продукту. Користувачі хочуть мати впевненість у безпеці, надійності та якості продукту, і тестування допомагає цю довіру заслужити.

Для тестування використовують методи «чорної» та «білої» скриньки.

Метод білої скриньки, відомий також як тестування на основі структури, заснований на дослідженні внутрішньої структури програми. Основна ідея полягає у тому, що тестувальник аналізує код, структуру даних та логіку програми, щоб створити тести, які покривають всі можливі шляхи виконання

програми. Цей метод дозволяє виявляти помилки, що пов'язані з неправильними умовами, циклами, гілками у коді тощо.

Метод чорної скриньки, відомий також як тестування на основі вимог або функціонального тестування, зосереджений на функціональності програми, без знання її внутрішньої реалізації. Основна ідея цього методу полягає у тому, що тестувальник створює тести на основі вимог, специфікацій або користувацьких сценаріїв, щоб перевірити, чи виконує програма очікувані функції і поводиться згідно з вимогами. Цей метод дозволяє перевірити поведінку програми зовнішнього користувача, але може не виявляти помилки, пов'язані зі структурою програми.

4.2. Тестування

Опис специфікації та тексту функції CalculateMaxValues.

Атрибути функції:

- вхідна змінна, яка містить двомірний масив;
- вхідна змінна, яка містить кількість рядків;
- вхідна змінна, яка містить кількість стовпців.

Функція повертає одномірний масив максимальних значень, для кожного рядка двомірної матриці.

```
/// <summary>
/// Function for finding the maximum value in a two-dimensional array.
/// </summary>
/// <param name="values"></param>
/// <param name="rowCount"></param>
/// <param name="columnCount"></param>
/// <returns></returns>
 internal double[] CalculateMaxValues(double[,] values, int rowCount, int columnCount)
        {
            var maxValue = new double[rowCount];
```

```
        for (int rowIndex = 0; rowIndex < rowCount; rowIndex++)
        {
            double max = double.MinValue;
            for (int columnIndex = 0; columnIndex < columnCount; columnIndex++)
            {
                var value = values[rowIndex, columnIndex];
                if (value > max)
                    max = value;
            }
            maxValue[rowIndex] = Math.Round(max, 1);
        }
        return maxValue;
    }
```

Блок схема функції CalculateMaxValues зображено на рис. 4.1.

Тести функції CalculateMaxValues приведено у табл. 4.1.

Таблиція 4.1 – Умовні позначення параметрів

Номер тесту	Вхідні значення			Вихідні значення
	Масив значень	Кількість рядків	Кількість стовпців	
1	{{6,1,4,5},{5,6,7,8}}	2	4	{6,8}
2	{{},{}}	0	0	NULL
3	{{7,7,7,7},{7,7,7,7}}	2	4	{7,7}

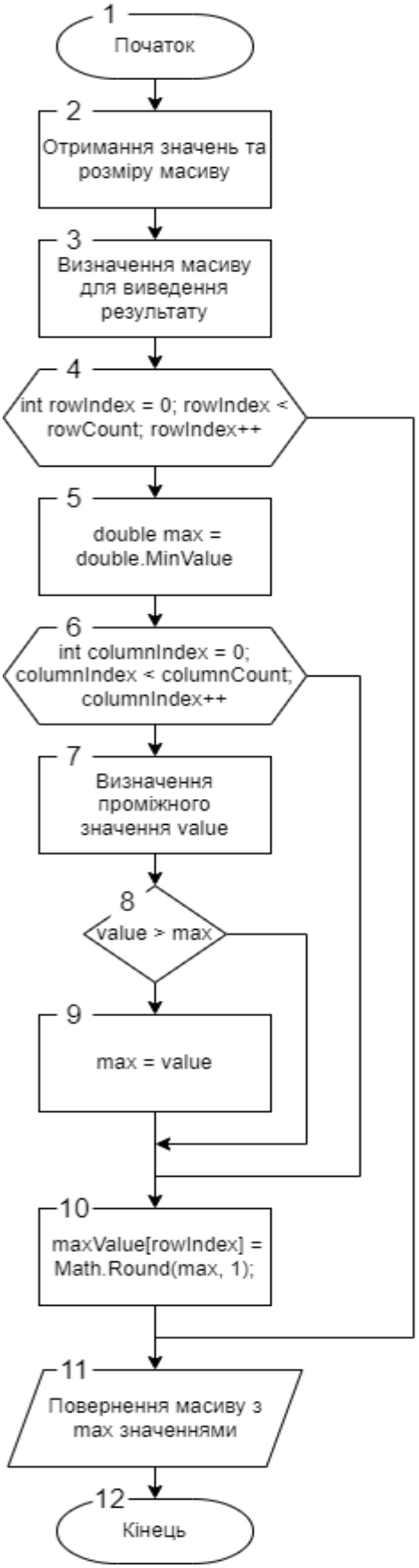


Рисунок 4.1 – Блок-схема алгоритму

Тестування методом покриття операторів описано у табл. 4.2.

Таблиція 4.2 – Тестування методом покриття операторів

Номер тесту	Номер оператора											
	1	2	3	4	5	6	7	8	9	10	11	12
1	+	+	+	+	+	+	+	+	+	+	+	+
2	-	-	-	-	-	-	-	-	-	-	-	-
3	+	+	+	+	+	+	+	+	+	+	+	+

При виконанні тестів кожен оператор, хоч один раз виконався

Висновки до розділу

Методи «білої» та «чорної» скриньки є важливими і можуть доповнювати один одного в процесі тестування. Метод білої скриньки спрямований на глибинне тестування внутрішніх компонентів програми, тоді як метод чорної скриньки перевіряє функціональність зовнішнього інтерфейсу.

Обрана функція була протестована методом покриття операторів, результат тестування є задовільним та відповідає очікуваним результатам.

Загальні висновки

Дипломна робота присвячена завданням щодо розробки нечітких алгоритмів і програмного забезпечення для моделювання, аналізу та прогнозування числових послідовностей даних, які отримані з нерівномірним кроком спостережень. Відмінність розроблених моделей, процедур та програмних засобів полягає і використанні двох (кількох) узгоджених часових послідовностей даних. Для завдань дослідження з нерівномірним кроком спостережень враховуються дві компоненти процесів – контрольований параметр (міра процесу), послідовність часових інтервалів між спостереженнями.

Мета роботи полягала у розвитку нечітких моделей недетермінованих часових послідовностей Fuzzy Time Series першого та вищих порядків, які характеризуються рівномірним кроком спостережень, шляхом урахування нерівномірного кроку за рахунок формування нових комбінованих математичних моделей, Combined Fuzzy Time Series першого і другого порядку. Також необхідно було розробити математичне та програмне забезпечення для процедур для процедур Combined Fuzzy Time Series.

Завданнями дипломної роботи полягали у наступному – розвитку постановок завдань щодо моделювання, аналізу та прогнозування даних, які відзначаються нерівномірним інтервалом між спостереженнями, розробкою математичних моделей і алгоритмів процедур Combined Fuzzy Time Series, проєктування і розробки програмного забезпечення для завдань у формі Combined Fuzzy Time Series. Для досягнення мети були використані методи математичного моделювання та нечітких множин, методи Fuzzy Time Series першого та вищих порядків, а також методи програмної інженерії щодо формування вимог, проєктування та розробки програмного забезпечення.

Дипломна робота складається з 4 розділів, має 17 рисунків та 3 таблиці, а також 2 додатка і список використаних джерел з 22 найменувань.

Наукова та технологічна новизна роботи визначається наступним: було отримано розвиток нечітких моделей недетермінованих часових послідовностей Fuzzy Time Series шляхом формування відповідних комбінованих математичних моделей Combined Fuzzy Time Series, які відрізняються використанням двох (кількох) часових послідовностей даних, в тому числі з нерівномірним кроком спостережень, розроблено нову комбіновану нечітку реляційну математичну модель процесів з нерівномірним кроком спостережень Combined Fuzzy Time Series, яка поєднує дві перемінні послідовності – контрольований параметр та інтервал між спостереженнями, розроблено комбіновані нечіткі реляційні моделі процесів першого та другого порядку, які на відміну від відомих враховують дані двох узгоджених процесів, що забезпечує аналіз та прогнозування процесів з нерівномірним кроком між інтервалами спостережень, розроблене програмне забезпечення для реалізації комбінованих нечітких реляційних моделей Combined Fuzzy Time Series.

Призначення результатів розробки обумовлене новими можливостями застосування моделей та програмних засобів для методу комбінованих нечітких реляційних моделей Combined Fuzzy Time Series, що дозволяє реалізувати процедури аналізу та прогнозування процесів з нерівномірним кроком спостережень.

Аналіз публікації та загальновідомих програмних засобів не дозволив виявити загально прийнятих і доступних програмних засобів реалізації завдань моделювання та дослідження даних з нерівномірним кроком спостережень, представлених у формі сукупності часових послідовностей.

Створення спеціалізованого програмного забезпечення, як програмного інструментарію з автоматизації завдань аналізу та прогнозування процесів з нерівномірним кроком спостережень, визначає експлуатаційне призначення розробки.

В результаті виконання дипломної роботи було розроблено удосконалений методу багатопараметричної лінійної екстраполяції, сформована нова процедура щодо реалізації зворотних завдань вибору та проєктування, створене програмне

забезпечення з автоматизації відповідних завдань процесів вибору та проєктування на основі даних про аналоги та прототипи створюваних систем.

Дипломна робота складається з реферату, вступу, розділу по збору та аналізу вимог, розділу проєктування, розділу розробки програми, розділу тестування та налагодження, загальних висновків та списку використаних джерел.

Наукова та технологічна новизна роботи визначається наступним: було удосконалена процедура багатовимірної лінійної екстраполяції при суттєво нерівних величинах параметрів моделей, розроблена нова процедура застосування методу багатовимірної лінійної екстраполяції із реалізації зворотних завдань вибору та проєктування, що відрізняється структурами шаблонів даних та забезпечує розрахунок оцінок параметрів моделі (типу підбору параметру), сформовані математичні моделі та реалізовані нові завдання вибору та проєктуваня на основі багатовимірної лінійної екстраполяції.

Значні можливості багатовимірної лінійної екстраполяції обумовлюють його широке прикладне значення, цей метод має велику апробації і показав досить високу ефективність на етапах попереднього аналізу. Аналіз не дозволив виявити загально відомих та прийнятих і доступних програмних засобів реалізації завдань багатовимірної лінійної екстраполяції. На основі наведено можна розглядати розроблені програмні засоби з застосування багатовимірної лінійної екстраполяції в якості інструментарію попереднього аналізу, вибору та формування наборів параметрів систем, які проєктуються, на основі наборів відомих аналогів та прототипів. В цілому результати дипломної роботи сприяють підвищенню ефективності процесів проєктування та аналізу складних систем, за умов існування аналогів або прототипів об’єктів.

Список використаних джерел

1. Ross T. J. Fuzzy Logic with Engineering Applications. Wiley & Sons, Incorporated, John, 2009. 606 p.

2. Yen J. Fuzzy logic: Intelligence, control, and information. Upper Saddle River, N.J : Prentice Hall, 1999. 548 p.

3. Pedrycz W. Fuzzy control and fuzzy systems. 2nd ed. Taunton, Somerset, England : Research Studies Press, 1993. 350 p.

4. FSKD 2005 (2005 Changsha, Hunan Sheng, China). Fuzzy systems and knowledge discovery: Second international conference, FSKD 2005, Changsha, China, August 27-29, 2005 : proceedings. Berlin : Springer, 2005.

5. David J. Forecasting Data Published at Irregular Time Intervals Using an Extension of Holt's Method: Published by: INFORMS Stable URL: Accessed: 26-05-2015 UTC REFERENCES: http://www.jstor.org/stable/2631582?seq=1&cid=pdf-reference#references_tab_contents

6. Elorrieta F., Eyheramendy S., Palma W. Discrete-time autoregressive model for unequally spaced time-series observations. A&A 627, A120 (2019) https://doi.org/10.1051/0004-6361/201935560, ESO 2019 Astronomy & Astrophysics.

7. Jan Prüser. Adaptive learning from model space. DOI: 10.1002/for.2549.

8. Скалозуб В.В., Мурашов О.В. (2021) Моделювання даних процесів моніторингу при нерівномірних і нечітких інтервалах спостережень. «Системні технології». Випуск 4 (135). 135-148.

9. Q. Song, and B. S. Chissom, “Forecasting enrollments with fuzzy time series — Part I,” Fuzzy Sets and Systems, vol. 54, issue 1, 1993a, pp. 1¬9.

10. Piegat A. Fuzzy modelling and control [Тект] / Pfysica-Verlag, Heidelberg, 2001. – 698 pp.

11. Koenker R. “Quantile Regression”, Cambridge University Press, NY- 2005. pp. 137 – 143.

12. Tahseen A., Aqil S., Burney Cemal A. A New Quantile Based Fuzzy Time Series Forecasting Model [Електронний ресурс] – Режим доступу: https://publications.waset.org/14214/pdf

13. E. Bas, U. Yolcu and E. Egrioglu, “Intuitionistic fuzzy time series functions approach for time series forecasting”, Granular Computing, 2020.

14. Pal S.S., Kar S. “Fuzzy Time Series Model for Unequal Interval Length Using Genetic Algorithm”. Advances in Intelligent Systems and Computing, vol. 699 2019.

15. Bernal J.L., Cummins S., Gasparrini A. “Interrupted time series regression for the evaluation of public health interventions: a tutorial”. International Journal of Epidemiology, vol. 46, Issue 1, 2016 pp. 348–355.

16. Carla S. M¨oller-Levet, F. Klawonn, Kwang-Hyun Cho and O. Wolkenhauer, “Fuzzy Clustering of Short Time-Series and Unevenly Distributed Sampling Points”, Advances in Intelligent Data Analysis V, 2003 pp. 330–340.

17. Геєць В.М. Моделі і методи соціально-економічного прогнозування /Геєць В.М., Клебанова Т.С., Черняк О.І. – Харків: ВД «ІНЖЕК», 205. – 396 с.

18. Скалозуб В.В. Методи інтелектуального моделювання процесів з перемінним інтервалом спостережень та конструктивного упорядкування «з вагою» / Скалозуб В.В., Білий Б.Б, Галабут О.О., Мурашов О.В.. // Системні технології. – 2020. – Випуск 5 (132). – С. 83-98.

19. [Електронний ресурс] C# documentation – https://docs.microsoft.com/en-us/dotnet/csharp/

20. [Електронний ресурс] Visual Studio documentation – https://docs.microsoft.com/en-us/visualstudio/

21. [Електронний ресурс] Microsoft Developer Community – https://developercommunity.visualstudio.com/

22. [Електронний ресурс] LiveCharts – https://lvcharts.net/App/examples/v1

Додатки

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор Українського державного університету науки і технологій

Анатолій РАДКЕВИЧ

16.11.23

РОЗРОБКА АЛГОРИТМІВ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОДЕЛЮВАННЯ ЧИСЛОВИХ ПОСЛІДОВНОСТЕЙ ДАНИХ З НЕРІВНОМІРНИМ КРОКОМ

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

1116130.01323-01-ЛЗ

Представники

підприємства-розробника

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

16.11.23

Керівник розробки

Владислав СКАЛОЗУБ

16.11.23

Виконавець

Ігор ПОПОВ

16.11.23

Нормоконтролер

Світлана ВОЛКОВА

16.11.23

2023

ЗАТВЕРДЖЕНО

1116130.01323-01-ЛЗ

РОЗРОБКА АЛГОРИТМІВ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОДЕЛЮВАННЯ ЧИСЛОВИХ ПОСЛІДОВНОСТЕЙ ДАНИХ З НЕРІВНОМІРНИМ КРОКОМ

Технічне завдання

1116130.01323-01

Листів 14

2023

ЗМІСТ

1 ВВЕДЕННЯ

Застосування комп’ютерного моделювання зараз є майже необхідною складовою наукових та прикладних досліджень, проектування, розробок технологічних процесів. що забезпечує аналіз і прогнозування характеристик складних процесів і систем. При тому саме складність досліджуваних процесів являється одним із головних факторів, що визначають можливості застосування певних процедур аналізу та прогнозування, а також визначають ефективність програмних систем. У багатьох завданнях моніторингу стану інформаційних та виробничих систем, в завданнях аналізу станів технологічних процесів та складних систем тощо, в якості достовірних та доступних вихідних даних можливо отримати лише величини певних характеристик процесів, що упорядковані у часові послідовності, утворюють часові ряди. Завдання щодо формування моделей, методів та програмних засобів, призначених для виконання та достовірного застосування процедур комп’ютерне моделювання на основі часових рядів, широко застосовуються і являються актуальними.

Відзначається, що на практиці мають місце процеси, в яких характеристики часових рядів вимірюються з нерівномірним кроком міх спостереженнями. Такі часові ряди характерні для багатьох процесів моніторингу станів складних технічних, інформаційних процесів, а також для важливих і відповідальних процесів клінічного моніторингу лікування хворих, процесів реабілітації тощо. Широко відомі та застосовувані статистичні методи для завдань з перемінним кроком не можуть використані. Огляд літературних джерел показав, що математичні моделі та методи аналізу та моделювання часових рядів з перемінним кроком спостережень є не достатньо дослідженими. Також відзначається відсутність або дуже обмежене використання програмних засобів аналізу і прогнозування нерівномірних у часі часових послідовностей. В змістовному сенсі саме існування нерівномірного інтервалу мір спостереженнями в часових рядах представляє головну особливість завдань моделювання, робить дослідження та розробки своєчасними і актуальними.

Для реалізації завдань аналізу і прогнозування характеристик часових рядів з нерівномірним кроком спостережень була запропонована сепарабельна модель, а також удосконалений квантильний алгоритм. Засобами сепарабельної моделі були досліджені дані клінічного моніторингу процесів реабілітації хворих. Особливість сепарабельної моделі полягає в тому, що окремі характеристики таких часових рядів, в тому числі послідовності інтервалів, моделюються окремо. Їх узгодження забезпечується за рахунок номерів кроків процесу спостережень, що визначається аксіоматично, як реалізація принципу системної єдності. В дипломній роботі запропонований метод, який розвиває класичні моделі нечітких часових послідовностей (Fuzzy Time Series), коли на відміну від сепарабельних моделей розглядаються і явно поєднуються у моделі дані двох (кількох) характеристик процесу (сукупність ознак, одна з них – це інтервал між спостереженнями). Такі комбіновані моделі позначаються в роботі як Combined Fuzzy Time Series. Розробка алгоритмів і програмного забезпечення для моделювання числових послідовностей даних з нерівномірним кроком на основі Combined Fuzzy Time Series є актуальним науково-прикладним завданням.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 07.12.22 №1209ст ректора Українського державного університету науки і технологій “Про призначення наукових керівників та затвердження тем бакалаврських робіт” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

Тема дипломної роботи – “РОЗРОБКА АЛГОРИТМІВ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОДЕЛЮВАННЯ ЧИСЛОВИХ ПОСЛІДОВНОСТЕЙ ДАНИХ З НЕРІВНОМІРНИМ КРОКОМ”. Керівник – проф. Скалозуб В. В.

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення – програмний продукт має реалізовувати новий метод прогнозування із одночасним ведення двох процесів, із використанням методів трикутника, числових рядів та прогнозування методом середньої ваги.

Експлуатаційне призначення – за допомогою програмного продукту, можна застосувати новий метод із використанням двох (кількох) часових послідовностей даних, що дозволяє одночасно досліджувати кілька узгоджених процесів, в тому числі з нерівномірним кроком спостережень.

4 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

4.1 Вимоги до функціональних характеристик

Програмний продукт повинний забезпечити введення користувачем кількості значень для формування таблиці значень, введення часових послідовностей даних.

Аналізуючи введені користувачем початкові дані, програмний продукт має генерувати і відображати:

– графік для методу трикутника із відповідними границями до введених послідовностей даних;

– таблицю із фазифікованими значеннями (нечіткими множинами);

– графіки відображення дослідження рядів послідовностей;

– таблицю переходів першого порядку;

– таблицю із прогнозованими значеннями.

4.2 Вимоги до надійності

Вимоги до надійності наступні:

– забезпечення надійності та стабільності роботи системи протягом тривалого періоду часу;

– контроль вхідної та вихідної інформації для запобігання витоку конфіденційної інформації та введенню неправдивих даних;

– захист системи від несанкціонованого доступу для запобігання крадіжкам даних та негативних наслідків для користувачів.

4.3 Вимоги експлуатації

Для забезпечення стійкого функціонування програмного продукту необхідно дотримуватись таких умов:

– допустима температура повітря від +10°C до +30°C, допустима відносна вологість повітря — від 40% до 60%;

– програмний продукт повинен використовуватись в кліматичних умовах, які підходять для стабільної роботи комп’ютера;

– для повноцінного використання необхідно робити системне технічне обслуговування обладнання та програмного забезпечення, а також регулярно оновлювати програмне забезпечення;

– користувач повинен мати достатній досвід роботи з ПК.

4.4. Вимоги до складу та параметрів технічних засобів

Розроблюваний програмний продукт розрахований на використання на персональному комп’ютері з операційною системою Windows, що має наступні характеристики:

– процесор з тактовою частотою мінімум 1 ГГц 4-ядерний або більш потужний;

– оперативна пам’ять: мінімум 4 ГБ RAM (рекомендується 8 ГБ або більше);

– простір на жорсткому диску: мінімум 10 ГБ вільного місця;

– відеокарта: підтримка DirectX9 або вищої версії з драйверами WDDM1.0;

– монітор: мінімальна роздільна здатність 800 х 600 пікселів;

– комунікації: USB-порт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмний продукт розробляється для операційної системи Windows 10 або вище.

4.6 Вимоги до маркування і упаковки

Упаковка програмного продукту, включаючи документацію повинна бути захищена від пошкоджень різного роду (механічних, кліматичних).

На упаковці повинно бути вказана назва продукту, номер версії, мінімальні системні вимоги.

На зворотній стороні упаковки вказується розробник та його юридична адреса.

Програмний продукт **РОЗРОБКА АЛГОРИТМІВ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОДЕЛЮВАННЯ ЧИСЛОВИХ ПОСЛІДОВНОСТЕЙ ДАНИХ З НЕРІВНОМІРНИМ КРОКОМ** **Мінімальні системні вимоги:** • процесор – 4-ядерний з частотою 1 ГГц; • оперативна пам'ять – 4 ГБ RAM; • вбудована пам'ять – 10 ГБ; • відеокарта з підтримкою DirectX9 і драйверами WDDM1.0; • монітор – мінімальна роздільна здатність 800 х 600; • комунікації – USB-порт.	Програмний продукт **РОЗРОБКА АЛГОРИТМІВ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОДЕЛЮВАННЯ ЧИСЛОВИХ ПОСЛІДОВНОСТЕЙ ДАНИХ З НЕРІВНОМІРНИМ КРОКОМ** **Розробник: Попов І. Ю.** **Кафедра «КІТ», УДУНТ** **м. Дніпро, вул. Лазаряна 2** **2023**

Рисунок 1 – Вимоги до маркування

4.7 Вимоги до транспортування та зберігання

Транспортування повинне забезпечувати збереження програмного продукту, його цілісність і запобігання несанкціонованого доступу до нього. Місце збереження повинно бути сухим, захищеним від прямих сонячних променів та пилу. Для транспортування буде використовуватися флеш-накопичувач.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу документації мають входити:

- специфікація;
- текст програми;
- опис програми;
- керівництво користувача.

Вся документація програмного додатку повинна задовольняти вимоги до програмної документації [1].

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиця 1 – Стадії та етапи розробки

№	Зміст роботи (розділу)	Термін виконання розділів роботи	Примітка
1	Вступ	01.02.2023 – 05.02.2023	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	06.02.2023 – 19.02.2023	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	20.02.2023 – 04.03.2023	
4	Постановка задачі, технічне завдання	05.03.2023 – 12.03.2023	30%
5	Техніко-економічні показники	13.03.2023 – 19.03.2023	
6	Розробка інструментальних засобів дослідження	20.03.2023 – 21.05.2023	60%
7	Виконання досліджень	22.05.2023 – 31.05.2023	
8	Оформлення результатів дипломної роботи	01.06.2023 – 18.06.2023	
9	Подання дипломної роботи до кафедри	20.06.2023	100%
10	Захист дипломної роботи на засіданні екзаменаційної комісії	28.06.2023	

7 ПОРЯДОК І КОНТРОЛЬ ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник розробки проф. Скалозуб В. В.

Прийом здійснюється комісією у складі:

– Горячкін В. М. (керівник підрозділу);

– Скалозуб В. В. (керівник розробки).

8 БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем: методичні вказівки до дипломного проектування та лабораторних робіт/уклад.: Ю.М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с

Додаток Б

Текст програми

```
<Window x:Class="Diploma.MainWindow"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:lvc="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"
      mc:Ignorable="d"
      Title="Моделювання числових послідовностей даних з нерівномірним кроком"
      Height="650" Width="1000"
      ResizeMode="CanResize"
      WindowStartupLocation="CenterScreen"
      WindowState="Maximized"
      FontFamily="Gadugi">
   <ScrollViewer VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Auto">
      <Grid Background="#E4DCCF">
         <Grid.RowDefinitions>
            <RowDefinition Height="120"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
         </Grid.RowDefinitions>

         <StackPanel Grid.Row="0" Orientation="Horizontal" HorizontalAlignment="Left"
VerticalAlignment="Center" Margin="20,0,0,0">
            <StackPanel Orientation="Vertical">
               <StackPanel Orientation="Horizontal">
                  <TextBlock x:Name="textBlockCount" Text="Введіть кількість значень:"
                  VerticalAlignment="Center" FontSize="14" FontWeight="Bold"/>

                  <!--Cell for entering numbers-->
                  <TextBox x:Name="textBoxCount" Height="30" Width="50"
                HorizontalContentAlignment="Center" VerticalContentAlignment="Center" FontSize="14"
Margin="10"/>
               </StackPanel>

               <Button x:Name="btnBrowse" Content="Ввести"
                     Height="30" Width="120" Background="#576F72" Foreground="White"
                     FontWeight="Bold" FontSize="16" HorizontalAlignment="Right" Click="CreateColumns_Click">
                  <Button.Resources>
                     <Style TargetType="Border">
```

```
                    <Setter Property="CornerRadius" Value="10"></Setter>
                  </Style>
                </Button.Resources>
              </Button>

              <Button x:Name="btnTest" Content="Приклад"
                  Height="30" Width="120" Background="#576F72" Foreground="White"
                  FontWeight="Bold" FontSize="16" HorizontalAlignment="Right" Click="btnTest_Click"
Margin="0,10,0,0">
                <Button.Resources>
                  <Style TargetType="Border">
                    <Setter Property="CornerRadius" Value="10"></Setter>
                  </Style>
                </Button.Resources>
              </Button>
            </StackPanel>

            <StackPanel x:Name="StackDataGrid1" Visibility="Hidden" VerticalAlignment="Stretch"
Margin="30,0,0,0">
              <StackPanel Orientation="Horizontal">
                <StackPanel Orientation="Vertical" Margin="0,10,5,0">
                  <TextBlock Text="№" VerticalAlignment="Center" FontSize="14" FontWeight="Bold"/>
                  <TextBlock Text="T =" VerticalAlignment="Center" FontSize="14" FontWeight="Bold"/>
                  <TextBlock Text="V =" VerticalAlignment="Center" FontSize="14" FontWeight="Bold"/>
                </StackPanel>

                <DataGrid  x:Name="dataGrid1" ItemsSource="{Binding}"
                  AutoGenerateColumns="False" CanUserAddRows="False"
                  CanUserDeleteRows="False" CanUserReorderColumns="False" BorderThickness="1"
                  CanUserResizeColumns="False" CanUserResizeRows="False" CanUserSortColumns="False"
                  FontSize="14" HorizontalAlignment="Left" ScrollViewer.CanContentScroll="True"
ColumnHeaderHeight="25"
                  SelectionMode="Single" SelectionUnit="Cell"
                  PreviewTextInput="dataGrid1_PreviewTextInput"
                  HeadersVisibility="Column" Width="Auto" VerticalAlignment="Stretch"/>

                <Button x:Name="btnCalc"
                  Height="50" Width="140" Margin="10"
                  Background="#576F72" Foreground="White"
                  FontWeight="Bold" FontSize="16"
                  Click="btnCalc_Click">
                  <Button.Resources>
```

```
                    <Style TargetType="Border">
                        <Setter Property="CornerRadius" Value="10"/>
                    </Style>
                </Button.Resources>

                <TextBlock Text="Розпочати прогнозування" TextWrapping="Wrap" TextAlignment="Center"/>

            </Button>
        </StackPanel>
    </StackPanel>
</StackPanel>

<!--Graphs for the triangle method-->
<StackPanel Visibility="Hidden" x:Name="StackDataGrid2" Grid.Row="1" Orientation="Horizontal"
HorizontalAlignment="Center">
    <!--Triangle method for time T-->
    <StackPanel Orientation="Vertical">
        <!--Creating a Graph-->
        <lvc:CartesianChart Hoverable="False" LegendLocation="Top" FontSize="14" Width="650"
Height="250">

            <!--X axis setting-->
            <lvc:CartesianChart.AxisX>
                <lvc:Axis Labels="{Binding LabelsT}" Foreground="Black" FontSize="14">
                    <lvc:Axis.Separator>
                        <lvc:Separator StrokeThickness="1" StrokeDashArray="5">
                            <lvc:Separator.Stroke>
                                <SolidColorBrush Color="#404F56" />
                            </lvc:Separator.Stroke>
                        </lvc:Separator>
                    </lvc:Axis.Separator>
                </lvc:Axis>
            </lvc:CartesianChart.AxisX>

            <!--Y axis setting-->
            <lvc:CartesianChart.AxisY>
                <lvc:Axis LabelFormatter="{Binding YFormatter}" MinValue="0" MaxValue="1"
Foreground="Black" FontSize="14">
                    <lvc:Axis.Separator>
                        <lvc:Separator StrokeThickness="1" StrokeDashArray="2">
                            <lvc:Separator.Stroke>
                                <SolidColorBrush Color="#404F56" />
```

```
                        </lvc:Separator.Stroke>
                    </lvc:Separator>
                </lvc:Axis.Separator>
            </lvc:Axis>
        </lvc:CartesianChart.AxisY>

        <!--Row setup-->
        <lvc:CartesianChart.Series>
            <lvc:LineSeries Title="T1" LineSmoothness="0" Fill="Transparent" PointGeometry="{x:Null}"
Values="{Binding LineT1}" />
            <lvc:LineSeries Title="T2" LineSmoothness="0" Fill="Transparent" PointGeometry="{x:Null}"
Values="{Binding LineT2}" />
            <lvc:LineSeries Title="T3" LineSmoothness="0" Fill="Transparent" PointGeometry="{x:Null}"
Values="{Binding LineT3}" />
            <lvc:LineSeries Title="T4" LineSmoothness="0" Fill="Transparent" PointGeometry="{x:Null}"
Values="{Binding LineT4}" />
            <lvc:LineSeries Title="T5" LineSmoothness="0" Fill="Transparent" PointGeometry="{x:Null}"
Values="{Binding LineT5}" />
        </lvc:CartesianChart.Series>
    </lvc:CartesianChart>

    <TextBlock Text="Таблиця фазифікованих значень T:" VerticalAlignment="Center"
HorizontalAlignment="Center" FontSize="14" FontWeight="Bold" Margin="0,10,0,10"/>

    <!--Table for displaying fuzzy sets for time T-->
    <DataGrid x:Name="dataGrid2" ItemsSource="{Binding}"
        AutoGenerateColumns="False" CanUserAddRows="False"
        CanUserDeleteRows="False" CanUserReorderColumns="False" BorderThickness="1"
        CanUserResizeColumns="False" CanUserResizeRows="False" CanUserSortColumns="False"
        FontSize="14" ScrollViewer.CanContentScroll="True" ColumnHeaderHeight="25"
        SelectionMode="Single" SelectionUnit="Cell"
        HeadersVisibility="Column" HorizontalAlignment="Center"/>

    <TextBlock Text="Матриця переходів для T:" VerticalAlignment="Center"
HorizontalAlignment="Center" FontSize="14" FontWeight="Bold" Margin="0,10,0,10"/>

    <DataGrid x:Name="dataGridFirstT" ItemsSource="{Binding}"
          AutoGenerateColumns="False" CanUserAddRows="False"
          CanUserDeleteRows="False" CanUserReorderColumns="False" BorderThickness="1"
          CanUserResizeColumns="False" CanUserResizeRows="False" CanUserSortColumns="False"
          FontSize="14" ScrollViewer.CanContentScroll="True" ColumnHeaderHeight="25"
          SelectionMode="Single" SelectionUnit="Cell"
```

```
                HeadersVisibility="Column" HorizontalAlignment="Center"/>
        </StackPanel>
        <!--Triangle method for value V-->
        <StackPanel Orientation="Vertical">
          <!--Creating a Graph-->
          <lvc:CartesianChart Hoverable="False" LegendLocation="Top" FontSize="14" Width="650"
Height="250" Margin="80,0,0,0">

            <!--X axis setting-->
            <lvc:CartesianChart.AxisX>
              <lvc:Axis Labels="{Binding LabelsV}" Foreground="Black" FontSize="14">
                <lvc:Axis.Separator>
                  <lvc:Separator StrokeThickness="1" StrokeDashArray="5">
                    <lvc:Separator.Stroke>
                      <SolidColorBrush Color="#404F56" />
                    </lvc:Separator.Stroke>
                  </lvc:Separator>
                </lvc:Axis.Separator>
              </lvc:Axis>
            </lvc:CartesianChart.AxisX>

            <!--Y axis setting-->
            <lvc:CartesianChart.AxisY>
              <lvc:Axis LabelFormatter="{Binding YFormatter}" MinValue="0" MaxValue="1"
Foreground="Black" FontSize="14">
                <lvc:Axis.Separator>
                  <lvc:Separator StrokeThickness="1" StrokeDashArray="2">
                    <lvc:Separator.Stroke>
                      <SolidColorBrush Color="#404F56" />
                    </lvc:Separator.Stroke>
                  </lvc:Separator>
                </lvc:Axis.Separator>
              </lvc:Axis>
            </lvc:CartesianChart.AxisY>

            <!--Row setup-->
            <lvc:CartesianChart.Series>
              <lvc:LineSeries Title="V1" LineSmoothness="0" Fill="Transparent" PointGeometry="{x:Null}"
Values="{Binding LineV1}" />
              <lvc:LineSeries Title="V2" LineSmoothness="0" Fill="Transparent" PointGeometry="{x:Null}"
Values="{Binding LineV2}" />
```

```
                    <lvc:LineSeries Title="V3" LineSmoothness="0" Fill="Transparent" PointGeometry="{x:Null}"
Values="{Binding LineV3}" />
                    <lvc:LineSeries Title="V4" LineSmoothness="0" Fill="Transparent" PointGeometry="{x:Null}"
Values="{Binding LineV4}" />
                    <lvc:LineSeries Title="V5" LineSmoothness="0" Fill="Transparent" PointGeometry="{x:Null}"
Values="{Binding LineV5}" />
                </lvc:CartesianChart.Series>
            </lvc:CartesianChart>

            <TextBlock Text="Таблиця фазифікованих значень V:" VerticalAlignment="Center"
HorizontalAlignment="Center" FontSize="14" FontWeight="Bold" Margin="0,10,0,10"/>

            <!--Table for displaying fuzzy sets for value V-->
            <DataGrid x:Name="dataGrid3" ItemsSource="{Binding}"
                AutoGenerateColumns="False" CanUserAddRows="False"
                CanUserDeleteRows="False" CanUserReorderColumns="False"
                BorderThickness="1" CanUserResizeColumns="False"
                CanUserResizeRows="False" CanUserSortColumns="False"
                FontSize="14" ScrollViewer.CanContentScroll="True" ColumnHeaderHeight="25"
                SelectionMode="Single" SelectionUnit="Cell"
                HeadersVisibility="Column" Width="Auto" HorizontalAlignment="Center"/>

            <TextBlock Text="Матриця переходів для V:" VerticalAlignment="Center"
HorizontalAlignment="Center" FontSize="14" FontWeight="Bold" Margin="0,10,0,10"/>

            <DataGrid x:Name="dataGridFirstV" ItemsSource="{Binding}"
                    AutoGenerateColumns="False" CanUserAddRows="False"
                    CanUserDeleteRows="False" CanUserReorderColumns="False" BorderThickness="1"
                    CanUserResizeColumns="False" CanUserResizeRows="False" CanUserSortColumns="False"
                    FontSize="14" ScrollViewer.CanContentScroll="True" ColumnHeaderHeight="25"
                    SelectionMode="Single" SelectionUnit="Cell"
                    HeadersVisibility="Column" HorizontalAlignment="Center"/>
        </StackPanel>
    </StackPanel>

    <!--Graphs for series of observations-->
    <StackPanel x:Name="StackDataGrid3" Grid.Row="2" Orientation="Horizontal" VerticalAlignment="Top"
HorizontalAlignment="Center" Margin="0,10,0,0" Visibility="Hidden">
        <StackPanel Orientation="Vertical">
            <!--Creating a Graph-->
            <lvc:CartesianChart Hoverable="False" LegendLocation="Top" FontSize="14" Width="650"
Height="250" Margin="0,0,80,0">
```

```
            <lvc:CartesianChart.AxisX>
              <lvc:Axis Labels="{Binding LabelsNum}" Foreground="Black" FontSize="14" MinValue="0"
MaxValue="{Binding}">
                <lvc:Axis.Separator>
                  <lvc:Separator StrokeThickness="1" StrokeDashArray="5">
                    <lvc:Separator.Stroke>
                      <SolidColorBrush Color="#404F56" />
                    </lvc:Separator.Stroke>
                  </lvc:Separator>
                </lvc:Axis.Separator>
              </lvc:Axis>
            </lvc:CartesianChart.AxisX>

            <lvc:CartesianChart.AxisY>
              <lvc:Axis LabelFormatter="{Binding YFormatter}" MinValue="0" MaxValue="5"
Foreground="Black" FontSize="14">
                <lvc:Axis.Separator>
                  <lvc:Separator StrokeThickness="1" StrokeDashArray="2">
                    <lvc:Separator.Stroke>
                      <SolidColorBrush Color="#404F56" />
                    </lvc:Separator.Stroke>
                  </lvc:Separator>
                </lvc:Axis.Separator>
              </lvc:Axis>
            </lvc:CartesianChart.AxisY>

            <lvc:CartesianChart.Series>
              <lvc:LineSeries Title="Ряд спостережень зміни часу" LineSmoothness="0" Fill="Transparent"
PointGeometry="{x:Null}" Values="{Binding LineT}" />
            </lvc:CartesianChart.Series>

          </lvc:CartesianChart>

          <DataGrid x:Name="dataGrid4" ItemsSource="{Binding}"
              AutoGenerateColumns="False" CanUserAddRows="False"
              CanUserDeleteRows="False" CanUserReorderColumns="False"
              BorderThickness="1" CanUserResizeColumns="False"
              CanUserResizeRows="False" CanUserSortColumns="False"
              FontSize="14" ScrollViewer.CanContentScroll="True" ColumnHeaderHeight="25"
              SelectionMode="Single" SelectionUnit="Cell"
              HeadersVisibility="Column" Width="Auto" HorizontalAlignment="Center">
```

```
            </DataGrid>

            <TextBlock x:Name="textBlockResultT" Text="{Binding}" VerticalAlignment="Center" FontSize="14"
FontWeight="Bold" Margin="0,10,0,20" HorizontalAlignment="Center"/>

        </StackPanel>

        <StackPanel Orientation="Vertical">
            <!--Creating a Graph-->
            <lvc:CartesianChart Hoverable="False" LegendLocation="Top" FontSize="14" Width="650"
Height="250">

                <lvc:CartesianChart.AxisX>
                    <lvc:Axis Labels="{Binding LabelsNum}" Foreground="Black" FontSize="14" MinValue="0"
MaxValue="{Binding}">
                        <lvc:Axis.Separator>
                            <lvc:Separator StrokeThickness="1" StrokeDashArray="5">
                                <lvc:Separator.Stroke>
                                    <SolidColorBrush Color="#404F56" />
                                </lvc:Separator.Stroke>
                            </lvc:Separator>
                        </lvc:Axis.Separator>
                    </lvc:Axis>
                </lvc:CartesianChart.AxisX>

                <lvc:CartesianChart.AxisY>
                    <lvc:Axis LabelFormatter="{Binding YFormatter}" MinValue="0" MaxValue="5"
Foreground="Black" FontSize="14">
                        <lvc:Axis.Separator>
                            <lvc:Separator StrokeThickness="1" StrokeDashArray="2">
                                <lvc:Separator.Stroke>
                                    <SolidColorBrush Color="#404F56" />
                                </lvc:Separator.Stroke>
                            </lvc:Separator>
                        </lvc:Axis.Separator>
                    </lvc:Axis>
                </lvc:CartesianChart.AxisY>

                <lvc:CartesianChart.Series>
                    <lvc:LineSeries Title="Ряд спостережень зміни значення" LineSmoothness="0" Fill="Transparent"
PointGeometry="{x:Null}" Values="{Binding LineV}" />
                </lvc:CartesianChart.Series>
```

```
            </lvc:CartesianChart>

            <DataGrid x:Name="dataGrid5" ItemsSource="{Binding}"
                AutoGenerateColumns="False" CanUserAddRows="False"
                CanUserDeleteRows="False" CanUserReorderColumns="False"
                BorderThickness="1" CanUserResizeColumns="False"
                CanUserResizeRows="False" CanUserSortColumns="False"
                FontSize="14" ScrollViewer.CanContentScroll="True" ColumnHeaderHeight="25"
                SelectionMode="Single" SelectionUnit="Cell"
                HeadersVisibility="Column" Width="Auto" HorizontalAlignment="Center">
            </DataGrid>

            <TextBlock x:Name="textBlockResultV" Text="{Binding}" VerticalAlignment="Center" FontSize="14"
FontWeight="Bold" Margin="0,10,0,20" HorizontalAlignment="Center"/>

          </StackPanel>
        </StackPanel>
      </Grid>
   </ScrollViewer>
</Window>
        using LiveCharts;
        using System;
        using System.ComponentModel;

        namespace Diploma
        {
           /// <summary>
           /// Declare internal class ChartData, which implements the interface INotifyPropertyChanged.
           /// </summary>
           internal class ChartData: INotifyPropertyChanged
           {
              public string[] LabelsT { get; set; }
              public string[] LabelsV { get; set; }
              public string[] LabelsNum { get; set; }
              public Func<double, string> YFormatter { get; set; }

              private ChartValues<double> lineT;
              private ChartValues<double> lineT1;
              private ChartValues<double> lineT2;
              private ChartValues<double> lineT3;
              private ChartValues<double> lineT4;
              private ChartValues<double> lineT5;
```

```
private ChartValues<double> lineV;
private ChartValues<double> lineV1;
private ChartValues<double> lineV2;
private ChartValues<double> lineV3;
private ChartValues<double> lineV4;
private ChartValues<double> lineV5;
public event PropertyChangedEventHandler PropertyChanged;

/// <summary>
/// Property declarations for accessing graph values
/// </summary>
public ChartValues<double> LineT1
{
   get { return lineT1; }
   set
   {
      lineT1 = value;
      OnPropertyChanged(nameof(LineT1));
   }
}

public ChartValues<double> LineT2
{
   get { return lineT2; }
   set
   {
      lineT2 = value;
      OnPropertyChanged(nameof(LineT2));
   }
}

public ChartValues<double> LineT3
{
   get { return lineT3; }
   set
   {
      lineT3 = value;
      OnPropertyChanged(nameof(LineT3));
   }
}
```

```
public ChartValues<double> LineT4
{
   get { return lineT4; }
   set
   {
      lineT4 = value;
      OnPropertyChanged(nameof(LineT4));
   }
}

public ChartValues<double> LineT5
{
   get { return lineT5; }
   set
   {
      lineT5 = value;
      OnPropertyChanged(nameof(LineT5));
   }
}

public ChartValues<double> LineV1
{
   get { return lineV1; }
   set
   {
      lineV1 = value;
      OnPropertyChanged(nameof(LineV1));
   }
}

public ChartValues<double> LineV2
{
   get { return lineV2; }
   set
   {
      lineV2 = value;
      OnPropertyChanged(nameof(LineV2));
   }
}

public ChartValues<double> LineV3
{
```

```
    get { return lineV3; }
    set
    {
        lineV3 = value;
        OnPropertyChanged(nameof(LineV3));
    }
}

public ChartValues<double> LineV4
{
    get { return lineV4; }
    set
    {
        lineV4 = value;
        OnPropertyChanged(nameof(LineV4));
    }
}

public ChartValues<double> LineV5
{
    get { return lineV5; }
    set
    {
        lineV5 = value;
        OnPropertyChanged(nameof(LineV5));
    }
}
public ChartValues<double> LineT
{
    get { return lineT; }
    set
    {
        lineT = value;
        OnPropertyChanged(nameof(LineT));
    }
}

public ChartValues<double> LineV
{
    get { return lineV; }
    set
    {
```

```
                lineV = value;
                OnPropertyChanged(nameof(LineV));
            }
        }

        public ChartData()
        {
            // Filling data for graphs
            LineT1 = new ChartValues<double> { 1, 0.5, 0, 0, 0 };
            LineT2 = new ChartValues<double> { 0.5, 1, 0.5, 0, 0 };
            LineT3 = new ChartValues<double> { 0, 0.5, 1, 0.5, 0 };
            LineT4 = new ChartValues<double> { 0, 0, 0.5, 1, 0.5 };
            LineT5 = new ChartValues<double> { 0, 0, 0, 0.5, 1 };

            LineV1 = new ChartValues<double> { 1, 0.5, 0, 0, 0 };
            LineV2 = new ChartValues<double> { 0.5, 1, 0.5, 0, 0 };
            LineV3 = new ChartValues<double> { 0, 0.5, 1, 0.5, 0 };
            LineV4 = new ChartValues<double> { 0, 0, 0.5, 1, 0.5 };
            LineV5 = new ChartValues<double> { 0, 0, 0, 0.5, 1 };

            YFormatter = value => value.ToString("N");
        }

        /// <summary>
        /// A method for generating a PropertyChanged event.
        /// </summary>
        /// <param name="propertyName"></param>
        protected virtual void OnPropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
using System;
using System.Collections.Generic;
using System.Data;
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Data;
using System.Windows.Media;

namespace Diploma
```

```
{
    internal class DataGridHelper
    {
        /// <summary>
        /// Function for creating columns in a dataGrid.
        /// </summary>
        /// <param name="columnCount"></param>
        internal void CreateDataGridColumns(int columnCount, DataGrid dataGrid, DataTable dataTable)
        {
            dataGrid.Columns.Clear();
            dataTable.Columns.Clear();

            for (int i = 0; i < columnCount; i++)
            {
                var columnName = $"Column {i + 1}";
                dataTable.Columns.Add(columnName);

                var column = new DataGridTextColumn
                {
                    Header = i,
                    Binding = new Binding(columnName)
                    {
                        TargetNullValue = true,
                        Mode = BindingMode.TwoWay
                    }
                };
                dataGrid.Columns.Add(column);
            }

            dataGrid.ItemsSource = dataTable.DefaultView;
        }

        /// <summary>
        /// Function for creating rows in the dataGrid.
        /// </summary>
        internal void CreateDataGridRows(DataTable dataTable, int index)
        {
            dataTable.Rows.Clear();
            if(index == 1)
            {
                for (int i = 0; i < 2; i++)
                {
```

```
            var newRow = dataTable.NewRow();
            dataTable.Rows.Add(newRow);
        }
    }
    else
    {
        for (int i = 0; i < 2; i++)
        {
            var newRow = dataTable.NewRow();
            if (i == 0)
            {
                newRow["Column 1"] = 6;
                newRow["Column 2"] = 7;
                newRow["Column 3"] = 10;
                newRow["Column 4"] = 5;
                newRow["Column 5"] = 8;
                newRow["Column 6"] = 15;
                newRow["Column 7"] = 9;
                newRow["Column 8"] = 7;
                newRow["Column 9"] = 12;
            }
            else
            {
                newRow["Column 1"] = 12;
                newRow["Column 2"] = 16;
                newRow["Column 3"] = 18;
                newRow["Column 4"] = 15;
                newRow["Column 5"] = 21;
                newRow["Column 6"] = 11;
                newRow["Column 7"] = 17;
                newRow["Column 8"] = 19;
                newRow["Column 9"] = 13;
            }
            dataTable.Rows.Add(newRow);
        }
    }
}

internal List<int> GetDataGridItems(int rowIndex, DataGrid dataGrid)
{
    List<int> listDataGrid1 = new List<int>();
```

```
            var items = dataGrid.Items;
            var columnCount = dataGrid.Columns.Count;

            var item = items[rowIndex] as DataRowView;

            for (int columnIndex = 0; columnIndex < columnCount; columnIndex++)
            {
                var cellValue = item[columnIndex];

                if (int.TryParse(cellValue.ToString(), out var intValue))
                {
                    listDataGrid1.Add(intValue);
                }
            }

            return listDataGrid1;
        }

        internal void CreateMatrixStep(DataGrid dataGrid, string prefix1, string prefix2)
        {
            for (int i = 1; i <= 6; i++)
            {
                DataGridTextColumn column = new DataGridTextColumn();
                if (i != 1) column.Header = $"{prefix2}" + (i - 1);
                column.Binding = new Binding("[" + (i - 1) + "]");
                column.IsReadOnly = true;
                dataGrid.Columns.Add(column);
            }

            for (int i = 1; i <= 5; i++)
            {
                dataGrid.Items.Add(new string[] { $"{prefix1}" + i, "", "", "", "", "" });
            }
        }

        internal void UpdateDataGrid(DataGrid dataGrid, List<Tuple<int, double>> values, string prefix, int
ColumnCount)
        {
            dataGrid.Columns.Clear();
            DataTable dataTable = new DataTable();

            for (int j = 0; j < ColumnCount; j++)
```

```
    {
        var columnName = $"{prefix}{j + 1}";
        dataTable.Columns.Add(columnName);

        var column = new DataGridTextColumn
        {
            Header = j,
            Binding = new Binding(columnName)
            {
                TargetNullValue = true,
                Mode = BindingMode.TwoWay
            }
        };

        dataGrid.Columns.Add(column);
    }

    for (int i = 0; i < 2; i++)
    {
        var newRow = dataTable.NewRow();
        for (int j = 0; j < ColumnCount; j++)
        {
            Tuple<int, double>? tuple = values[j];

            if (i == 0)
                newRow[$"{prefix}{j + 1}"] = $"{prefix}" + tuple.Item1;
            else
                newRow[$"{prefix}{j + 1}"] = Math.Round(tuple.Item2, 1);
        }
        dataTable.Rows.Add(newRow);
    }

    dataGrid.ItemsSource = dataTable.DefaultView;
}

internal void CreateColumnsForArray(DataGrid dataGrid, int arrayLength, double[] array)
{
    DataTable dataTable = new DataTable();

    for (int j = 0; j < 5; j++)
    {
        var columnName = $"{j + 1}";
```

```
            dataTable.Columns.Add(columnName);

            var column = new DataGridTextColumn
            {
                Header = j,
                Binding = new Binding(columnName)
                {
                    TargetNullValue = true,
                    Mode = BindingMode.TwoWay
                }
            };

            dataGrid.Columns.Add(column);
        }

        for (int i = 0; i < 1; i++)
        {
            var newRow = dataTable.NewRow();
            for (int j = 0; j < arrayLength; j++)
            {
                newRow[$"{j + 1}"] = array[j];
            }
            dataTable.Rows.Add(newRow);
        }

        dataGrid.ItemsSource = dataTable.DefaultView;
    }

    internal void UpdateMatrixStep(DataGrid dataGrid, List<Tuple<int, double>> resultT, List<Tuple<int, double>> resultV, string prefix)
    {
        for (int i = 0; i < resultT.Count - 1; i++)
        {
            int columnIndexT = resultT[i].Item1;
            int columnIndexV = resultV[i].Item1;

            DataGridCellInfo cellInfoT = new DataGridCellInfo(dataGrid.Items[columnIndexT - 1], dataGrid.Columns[columnIndexV]);
            DataGridCell cellT = GetCell(dataGrid, cellInfoT);
            cellT.Content = $"{prefix}" + resultT[i + 1].Item1;
        }
    }
```

```
        private DataGridCell GetCell(DataGrid dataGrid, DataGridCellInfo cellInfo)
        {
            DataGridRow row = (DataGridRow)dataGrid.ItemContainerGenerator.ContainerFromItem(cellInfo.Item);
            if (row != null)
            {
                int columnIndex = dataGrid.Columns.IndexOf(cellInfo.Column);
                if (columnIndex >= 0)
                {
                    DataGridCellsPresenter presenter = GetVisualChild<DataGridCellsPresenter>(row);
                    if (presenter != null)
                    {
                        return presenter.ItemContainerGenerator.ContainerFromIndex(columnIndex) as DataGridCell;
                    }
                }
            }
            return null;
        }

        private static T GetVisualChild<T>(Visual parent) where T : Visual
        {
            T child = default(T);
            int numVisuals = VisualTreeHelper.GetChildrenCount(parent);
            for (int i = 0; i < numVisuals; i++)
            {
                Visual v = (Visual)VisualTreeHelper.GetChild(parent, i);
                child = v as T;
                if (child == null)
                {
                    child = GetVisualChild<T>(v);
                }
                if (child != null)
                {
                    break;
                }
            }
            return child;
        }
    }
}
    using System;
```

```
using System.Collections.Generic;
using System.Data;
using System.Windows.Controls;

namespace Diploma
{
    internal class StatisticsCalculator
    {
        /// <summary>
        /// A function for extracting numeric values from a collection, and saving those values to a two-dimensional array for further processing or analysis.
        /// </summary>
        /// <param name="items"></param>
        /// <param name="rowCount"></param>
        /// <param name="columnCount"></param>
        /// <returns></returns>
        internal double[,] GetCellValues(ItemCollection items, int rowCount, int columnCount)
        {
            var values = new double[rowCount, columnCount];

            for (int rowIndex = 0; rowIndex < rowCount; rowIndex++)
            {
                var item = items[rowIndex] as DataRowView;

                if (item == null)
                    continue;

                for (int columnIndex = 0; columnIndex < columnCount; columnIndex++)
                {
                    var cellValue = item[columnIndex];

                    if (double.TryParse(cellValue.ToString(), out var value))
                    {
                        values[rowIndex, columnIndex] = value;
                    }
                }
            }

            return values;
        }

        /// <summary>
```

```
/// A function for finding the minimum value in a two-dimensional array.
/// </summary>
/// <param name="values"></param>
/// <param name="rowCount"></param>
/// <param name="columnCount"></param>
/// <returns></returns>
internal double[] CalculateMinValues(double[,] values, int rowCount, int columnCount)
{
   var minValue = new double[rowCount];

   for (int rowIndex = 0; rowIndex < rowCount; rowIndex++)
   {
      double min = double.MaxValue;

      for (int columnIndex = 0; columnIndex < columnCount; columnIndex++)
      {
         var value = values[rowIndex, columnIndex];

         if (value < min)
            min = value;
      }

      minValue[rowIndex] = Math.Round(min, 1);
   }

   return minValue;
}

/// <summary>
/// Function for finding the maximum value in a two-dimensional array.
/// </summary>
/// <param name="values"></param>
/// <param name="rowCount"></param>
/// <param name="columnCount"></param>
/// <returns></returns>
internal double[] CalculateMaxValues(double[,] values, int rowCount, int columnCount)
{
   var maxValue = new double[rowCount];

   for (int rowIndex = 0; rowIndex < rowCount; rowIndex++)
   {
      double max = double.MinValue;
```

```
            for (int columnIndex = 0; columnIndex < columnCount; columnIndex++)
            {
                var value = values[rowIndex, columnIndex];

                if (value > max)
                    max = value;
            }

            maxValue[rowIndex] = Math.Round(max, 1);
        }

        return maxValue;
    }

    /// <summary>
    /// A function for finding the average between two elements.
    /// </summary>
    /// <param name="values1"></param>
    /// <param name="values2"></param>
    /// <param name="rowCount"></param>
    /// <returns></returns>
    internal double[] CalculateAverageBetweenValues(double[] values1, double[] values2, int rowCount)
    {
        var averageBetweenValue = new double[rowCount];

        for (int i = 0; i < rowCount; i++)
        {
            averageBetweenValue[i] = Math.Round((values1[i] + values2[i]) / 2, 1);
        }

        return averageBetweenValue;
    }

    internal List<Tuple<int, double>> FuzzificationAllData(double veryLow, double low, double medium, double high, double veryHigh, List<int> dataGrid)
    {
        var item = dataGrid;
        var columnCount = dataGrid.Count;

        List<Tuple<int, double>> resultTuple = new List<Tuple<int, double>>();
```

```
                double membershipValue = 0;

                if (item != null)
                {
                    for (int columnIndex = 0; columnIndex < columnCount; columnIndex++)
                    {
                        var cellValue = item[columnIndex];

                        if (double.TryParse(cellValue.ToString(), out var value))
                        {
                            if (value <= veryLow || (value > veryLow && value <= (low + veryLow) / 2))
                            {
                                membershipValue = CalculateMembershipValue(value, 0, veryLow, medium);
                                resultTuple.Add(new Tuple<int, double>(1, membershipValue));
                            }
                            else if (((value > (low + veryLow) / 2) && value <= low) || (value > low && value <= (low +
medium) / 2))
                            {
                                membershipValue = CalculateMembershipValue(value, (low - veryLow), low, high);
                                resultTuple.Add(new Tuple<int, double>(2, membershipValue));
                            }
                            else if ((value > (low + medium) / 2) && value <= medium || ((value > medium && value <=
(medium + high) / 2)))
                            {
                                membershipValue = CalculateMembershipValue(value, veryLow, medium, veryHigh);
                                resultTuple.Add(new Tuple<int, double>(3, membershipValue));
                            }
                            else if ((value > (medium + high) / 2) && value <= medium || (value > medium && value <=
(veryHigh + high) / 2))
                            {
                                membershipValue = CalculateMembershipValue(value, low, high, veryHigh + (veryHigh -
high));
                                resultTuple.Add(new Tuple<int, double>(4, membershipValue));
                            }
                            else if (value >= veryHigh || (value > high && value <= (veryHigh + high) / 2))
                            {
                                membershipValue = CalculateMembershipValue(value, medium, veryHigh, veryLow +
veryHigh);
                                resultTuple.Add(new Tuple<int, double>(5, membershipValue));
                            }
                        }
                    }
```

```
            return resultTuple;
        }
        else
        {
            return resultTuple;
        }
    }

    internal double[] Fuzzification(double veryLow, double low, double medium, double high, double veryHigh, double value)
    {
        double[] resultTuple = new double[5];

        double membershipValue = 0;

        membershipValue = CalculateMembershipValue(value, 0, veryLow, medium);
        resultTuple[0] = membershipValue;

        membershipValue = CalculateMembershipValue(value, (low - veryLow), low, high);
        resultTuple[1] = membershipValue;

        membershipValue = CalculateMembershipValue(value, veryLow, medium, veryHigh);
        resultTuple[2] = membershipValue;

        membershipValue = CalculateMembershipValue(value, low, high, veryHigh + (veryHigh - high));
        resultTuple[3] = membershipValue;

        membershipValue = CalculateMembershipValue(value, medium, veryHigh, veryLow + veryHigh);
        resultTuple[4] = membershipValue;

        return resultTuple;
    }

    private double CalculateMembershipValue(double x, double lowerBound, double peakPoint, double upperBound)
    {
        if (x <= lowerBound || x >= upperBound)
            return 0;
        else if (x >= lowerBound && x <= peakPoint)
            return (x - lowerBound) / (peakPoint - lowerBound);
        else if (x > peakPoint && x <= upperBound)
            return (upperBound - x) / (upperBound - peakPoint);
```

```
            else
                return 0;
        }

        internal double[] FindMaxValues(double[,] array)
        {
            int columnCount = array.GetLength(1);
            double[] maxValues = new double[columnCount];

            for (int columnIndex = 0; columnIndex < columnCount; columnIndex++)
            {
                double max = double.MinValue;

                for (int rowIndex = 0; rowIndex < array.GetLength(0); rowIndex++)
                {
                    double value = array[rowIndex, columnIndex];

                    if (value > max)
                        max = value;
                }

                maxValues[columnIndex] = max;
            }

            return maxValues;
        }

        internal void UpdateDesiredArrays(double[,] desiredArray1, double[,] desiredArray2, double[] result1, double[] result2)
        {
            for (int rowIndex = 0; rowIndex < result1.Length; rowIndex++)
            {
                for (int columnIndex = 0; columnIndex < result1.Length; columnIndex++)
                {
                    desiredArray1[rowIndex, columnIndex] = Math.Min(result1[rowIndex], desiredArray1[rowIndex, columnIndex]);
                    desiredArray2[rowIndex, columnIndex] = Math.Min(result2[rowIndex], desiredArray2[rowIndex, columnIndex]);
                }
            }
        }
    }
```

```
}
        using LiveCharts;
        using System;
        using System.Collections.Generic;
        using System.Data;
        using System.Linq;
        using System.Text.RegularExpressions;
        using System.Windows;
        using System.Windows.Controls;
        using System.Windows.Data;
        using System.Windows.Input;

        namespace Diploma
        {
            public partial class MainWindow : Window
            {
                DataTable dataTable;
                ChartData chartData;
                DataGridHelper dataGridHelper;
                StatisticsCalculator statisticsCalculator;

                static int ColumnCount { get; set; }
                static double[,,] tables;

                public MainWindow()
                {
                    InitializeComponent();

                    dataTable = new DataTable();
                    chartData = new ChartData();
                    dataGridHelper = new DataGridHelper();
                    statisticsCalculator = new StatisticsCalculator();

                    DataContext = chartData;
                    tables = CreateTables();
                    dataGridHelper.CreateMatrixStep(dataGridFirstT, "T", "V");
                    dataGridHelper.CreateMatrixStep(dataGridFirstV, "V", "T");
                }

                private double[,,] CreateTables()
                {
                    double[,,] tables = new double[25, 5, 5];
```

```
            double[,] patterns = new double[,]
            {
                { 1, 0.5, 0, 0, 0 },
                { 0.5, 1, 0.5, 0, 0 },
                { 0, 0.5, 1, 0.5, 0 },
                { 0, 0, 0.5, 1, 0.5 },
                { 0, 0, 0, 0.5, 1 }
            };

            int test = 0, numTab = 0, rowIndex = 0;

            for (int countTable = 0; countTable < 5; countTable++)
            {
                test += rowIndex;
                for (rowIndex = 0; rowIndex < 5; rowIndex++)
                {
                    numTab = test + rowIndex;
                    for (int columnIndex = 0; columnIndex < 5; columnIndex++)
                    {
                        for (int cellIndex = 0; cellIndex < 5; cellIndex++)
                        {
                            tables[numTab, columnIndex, cellIndex] = Math.Min(patterns[countTable, columnIndex],
patterns[rowIndex, cellIndex]);
                        }
                    }
                }
            }

            return tables;
        }

        private double[,] GetTwoDimensionalArray(int arrayIndex)
        {
            int rows = tables.GetLength(1);
            int columns = tables.GetLength(2);
            double[,] desiredArray = new double[rows, columns];

            for (int rowIndex = 0; rowIndex < rows; rowIndex++)
            {
                for (int columnIndex = 0; columnIndex < columns; columnIndex++)
                {
```

```
            desiredArray[rowIndex, columnIndex] = tables[arrayIndex, rowIndex, columnIndex];
        }
    }

    return desiredArray;
}

private void CreateColumns_Click(object sender, RoutedEventArgs e)
{
    if (int.TryParse(textBoxCount.Text, out int columnCount))
    {
        ColumnCount = columnCount;
        dataGridHelper.CreateDataGridColumns(columnCount, dataGrid1, dataTable);
        dataGridHelper.CreateDataGridRows(dataTable, 1);
        StackDataGrid1.Visibility = Visibility.Visible;
    }
    else
    {
        MessageBox.Show("Будь-ласка, введіть ціле число.");
    }
}

private void btnTest_Click(object sender, RoutedEventArgs e)
{
    textBoxCount.Text = "9";
    if (int.TryParse(textBoxCount.Text, out int columnCount))
    {
        ColumnCount = columnCount;
        dataGridHelper.CreateDataGridColumns(columnCount, dataGrid1, dataTable);
        dataGridHelper.CreateDataGridRows(dataTable, 0);
        StackDataGrid1.Visibility = Visibility.Visible;
    }
    else
    {
        MessageBox.Show("Будь-ласка, введіть ціле число.");
    }
}

/// <summary>
/// Function for checking the number input.
/// </summary>
/// <param name="sender"></param>
```

```
        /// <param name="e"></param>
        private void dataGrid1_PreviewTextInput(object sender, TextCompositionEventArgs e)
        {
            e.Handled = new Regex("[^0-9]+").IsMatch(e.Text);
        }

        /// <summary>
        /// A function for getting elements in the boundaries of a triangle method graph.
        /// </summary>
        /// <param name="minValue"></param>
        /// <param name="maxValue"></param>
        /// <param name="averageValue"></param>
        /// <param name="averageBetweenMinAndAverage"></param>
        /// <param name="averageBetweenAverageAndMax"></param>
        private void CalculateStatistics(out double[] minValue, out double[] maxValue, out double[] averageValue,
out double[] averageBetweenMinAndAverage, out double[] averageBetweenAverageAndMax)
        {
            var items = dataGrid1.Items;
            minValue = maxValue = averageValue = averageBetweenMinAndAverage =
averageBetweenAverageAndMax = null;

            if (items.Count == 0)
            {
                MessageBox.Show("DataGrid is empty.");
                return;
            }

            var rowCount = items.Count;
            var columnCount = dataGrid1.Columns.Count;
            var values = statisticsCalculator.GetCellValues(items, rowCount, columnCount);
            minValue = statisticsCalculator.CalculateMinValues(values, rowCount, columnCount);
            maxValue = statisticsCalculator.CalculateMaxValues(values, rowCount, columnCount);
            averageValue = statisticsCalculator.CalculateAverageBetweenValues(minValue, maxValue, rowCount);
            averageBetweenMinAndAverage = statisticsCalculator.CalculateAverageBetweenValues(minValue,
averageValue, rowCount);
            averageBetweenAverageAndMax =
statisticsCalculator.CalculateAverageBetweenValues(averageValue, maxValue, rowCount);
        }

        private void UpdateChartData(double[] minValue, double[] averageBetweenMinAndAverage, double[]
averageValue, double[] averageBetweenAverageAndMax, double[] maxValue)
        {
```

```
            chartData.LabelsT = new[] { $"{minValue[0]}", $"{averageBetweenMinAndAverage[0]}", $"{averageValue[0]}", $"{averageBetweenAverageAndMax[0]}", $"{maxValue[0]}" };
            chartData.LabelsV = new[] { $"{minValue[1]}", $"{averageBetweenMinAndAverage[1]}", $"{averageValue[1]}", $"{averageBetweenAverageAndMax[1]}", $"{maxValue[1]}" };
        }

        private void ShowDataGrids()
        {
            StackDataGrid2.Visibility = Visibility.Visible;
            StackDataGrid3.Visibility = Visibility.Visible;
        }

        private void UpdateChartLineData(List<Tuple<int, double>> resultT, List<Tuple<int, double>> resultV)
        {
            chartData.LineT = new ChartValues<double> { };
            chartData.LineV = new ChartValues<double> { };
            chartData.LabelsNum = new string[ColumnCount];
            for (int j = 0; j < ColumnCount; j++)
            {
                Tuple<int, double>? tuple1 = resultT[j];
                Tuple<int, double>? tuple2 = resultV[j];
                chartData.LineT.Add(tuple1.Item2 + tuple1.Item1 - 1);
                chartData.LineV.Add(tuple2.Item2 + tuple2.Item1 - 1);
                chartData.LabelsNum[j] = j.ToString();
            }
        }

        private void btnCalc_Click(object sender, RoutedEventArgs e)
        {
            CalculateStatistics(out double[] minValue, out double[] maxValue, out double[] averageValue, out double[] averageBetweenMinAndAverage, out double[] averageBetweenAverageAndMax);
            UpdateChartData(minValue, averageBetweenMinAndAverage, averageValue, averageBetweenAverageAndMax, maxValue);
            ShowDataGrids();
            List<int> firstRow = dataGridHelper.GetDataGridItems(0, dataGrid1);
            List<int> secondRow = dataGridHelper.GetDataGridItems(1, dataGrid1);

            for (int i = 0; i < 1; i++)
            {
                List<Tuple<int, double>> resultT = statisticsCalculator.FuzzificationAllData(minValue[0], averageBetweenMinAndAverage[0], averageValue[0], averageBetweenAverageAndMax[0], maxValue[0], firstRow);
```

```
List<Tuple<int, double>> resultV = statisticsCalculator.FuzzificationAllData(minValue[1], averageBetweenMinAndAverage[1], averageValue[1], averageBetweenAverageAndMax[1], maxValue[1], secondRow);
dataGridHelper.UpdateDataGrid(dataGrid2, resultT, "T", ColumnCount);
dataGridHelper.UpdateDataGrid(dataGrid3, resultV, "V", ColumnCount);

dataGridHelper.UpdateMatrixStep(dataGridFirstT, resultT, resultV, "T");
dataGridHelper.UpdateMatrixStep(dataGridFirstV, resultV, resultT, "V");

UpdateChartLineData(resultT, resultV);

int rowIndex;
double[,] patterns = new double[,]
{
   { 1, 0.5, 0, 0, 0 },
   { 0.5, 1, 0.5, 0, 0 },
   { 0, 0.5, 1, 0.5, 0 },
   { 0, 0, 0.5, 1, 0.5 },
   { 0, 0, 0, 0.5, 1 }
};


double[] result1 = statisticsCalculator.Fuzzification(minValue[0], averageBetweenMinAndAverage[0], averageValue[0], averageBetweenAverageAndMax[0], maxValue[0], firstRow[firstRow.Count - 2]);
double[] result2 = statisticsCalculator.Fuzzification(minValue[1], averageBetweenMinAndAverage[1], averageValue[1], averageBetweenAverageAndMax[1], maxValue[1], secondRow[secondRow.Count - 2]);

int arrayIndex1 = (resultT[resultT.Count - 2].Item1 - 1) * 5 + resultV[resultV.Count - 2].Item1 - 1;
int arrayIndex2 = (resultV[resultV.Count - 2].Item1 - 1) * 5 + resultT[resultT.Count - 2].Item1 - 1;
double[,] desiredArray1 = GetTwoDimensionalArray(arrayIndex1);
double[,] desiredArray2 = GetTwoDimensionalArray(arrayIndex2);

statisticsCalculator.UpdateDesiredArrays(desiredArray1, desiredArray2, result1, result2);

double[] maxValues1 = statisticsCalculator.FindMaxValues(desiredArray1);
double[] maxValues2 = statisticsCalculator.FindMaxValues(desiredArray2);

desiredArray1 = GetTwoDimensionalArray(arrayIndex1);
desiredArray2 = GetTwoDimensionalArray(arrayIndex2);
if (maxValues1.Sum() == 0)
```

```
{
   double[] row = GetRowFromArray(patterns, resultT[resultT.Count - 2].Item1 - 1);
   statisticsCalculator.UpdateDesiredArrays(desiredArray1, desiredArray2, row, result2);
   maxValues1 = statisticsCalculator.FindMaxValues(desiredArray1);
}
else if (maxValues2.Sum() == 0)
{
   double[] row = GetRowFromArray(patterns, resultV[resultV.Count - 2].Item1 - 1);
   statisticsCalculator.UpdateDesiredArrays(desiredArray1, desiredArray2, result1, row);
   maxValues2 = statisticsCalculator.FindMaxValues(desiredArray2);
}
else if (maxValues1.Sum() == 0 && maxValues2.Sum() == 0)
{
   double[] row1 = GetRowFromArray(patterns, resultT[resultT.Count - 2].Item1 - 1);
   double[] row2 = GetRowFromArray(patterns, resultV[resultV.Count - 2].Item1 - 1);

   statisticsCalculator.UpdateDesiredArrays(desiredArray1, desiredArray2, row1, row2);
   maxValues1 = statisticsCalculator.FindMaxValues(desiredArray1);
   maxValues2 = statisticsCalculator.FindMaxValues(desiredArray2);
}

arrayIndex1 = (resultT[resultT.Count - 1].Item1 - 1) * 5 + resultV[resultV.Count - 1].Item1 - 1;
arrayIndex2 = (resultV[resultV.Count - 1].Item1 - 1) * 5 + resultT[resultT.Count - 1].Item1 - 1;
desiredArray1 = GetTwoDimensionalArray(arrayIndex1);
desiredArray2 = GetTwoDimensionalArray(arrayIndex2);

statisticsCalculator.UpdateDesiredArrays(desiredArray1, desiredArray2, maxValues1, maxValues2);

maxValues1 = statisticsCalculator.FindMaxValues(desiredArray1);
maxValues2 = statisticsCalculator.FindMaxValues(desiredArray2);

desiredArray1 = GetTwoDimensionalArray(arrayIndex1);
desiredArray2 = GetTwoDimensionalArray(arrayIndex2);
if (maxValues1.Sum() == 0)
{
   double[] row = GetRowFromArray(patterns, resultT[resultT.Count - 1].Item1 - 1);
   statisticsCalculator.UpdateDesiredArrays(desiredArray1, desiredArray2, row, result2);
   maxValues1 = statisticsCalculator.FindMaxValues(desiredArray1);
}
else if (maxValues2.Sum() == 0)
{
   double[] row = GetRowFromArray(patterns, resultV[resultV.Count - 1].Item1 - 1);
```

```
                    statisticsCalculator.UpdateDesiredArrays(desiredArray1, desiredArray2, result1, row);
                    maxValues2 = statisticsCalculator.FindMaxValues(desiredArray2);
                }
                else if (maxValues1.Sum() == 0 && maxValues2.Sum() == 0)
                {
                    double[] row1 = GetRowFromArray(patterns, resultT[resultT.Count - 1].Item1 - 1);
                    double[] row2 = GetRowFromArray(patterns, resultV[resultV.Count - 1].Item1 - 1);

                    statisticsCalculator.UpdateDesiredArrays(desiredArray1, desiredArray2, row1, row2);
                    maxValues1 = statisticsCalculator.FindMaxValues(desiredArray1);
                    maxValues2 = statisticsCalculator.FindMaxValues(desiredArray2);
                }

                double resultTime = (maxValues1[0] * minValue[0] + maxValues1[1] * averageBetweenMinAndAverage[0] + maxValues1[2] * averageValue[0] + maxValues1[3] * averageBetweenAverageAndMax[0] + maxValues1[4] * maxValue[0]) / maxValues1.Sum();
                double resultValue = (maxValues2[0] * minValue[1] + maxValues2[1] * averageBetweenMinAndAverage[1] + maxValues2[2] * averageValue[1] + maxValues2[3] * averageBetweenAverageAndMax[1] + maxValues2[4] * maxValue[1]) / maxValues2.Sum();

                firstRow.Add((int)resultTime);
                secondRow.Add((int)resultValue);

                dataGridHelper.CreateColumnsForArray(dataGrid4, maxValues1.Length, maxValues1);
                dataGridHelper.CreateColumnsForArray(dataGrid5, maxValues1.Length, maxValues2);

                textBlockResultT.Text = $"Згідно розрахунку методом середньої ваги, наступне значення T: {(int)resultTime}";
                textBlockResultV.Text = $"Згідно розрахунку методом середньої ваги, наступне значення V: {(int)resultValue}";
            }
        }

        private double[] GetRowFromArray(double[,] array, int rowIndex)
        {
            int columnCount = array.GetLength(1);
            double[] row = new double[columnCount];

            for (int columnIndex = 0; columnIndex < columnCount; columnIndex++)
            {
                row[columnIndex] = array[rowIndex, columnIndex];
            }
```

```
                return row;
            }
        }
    }
```