МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра фундаментальної та прикладної математики

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«АВТОМАТИЗАЦІЯ ДОСЛІДЖЕННЯ ОСНОВНИХ ВЛАСТИВОСТЕЙ СКЛАДНИХ КЕРОВАНИХ ДИНАМІЧНИХ СИСТЕМ»**

Виконав: студент 4 курсу, групи 6.1139

спеціальності 113 прикладна математика
(шифр і назва спеціальності)

освітньої програми прикладна математика
(назва освітньої програми)

О. В. Старовойтов
(ініціали та прізвище)

Керівник доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Леонтьєва В. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент декан математичного факультету, професор, д.т.н. Гоменюк С.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра фундаментальної та прикладної математики

Рівень вищої освіти бакалавр

Спеціальність 113 прикладна математика

(шифр і назва)

Освітня програма Прикладна математика

ЗАТВЕРДЖУЮ

завідувач кафедри фундаментальної та прикладної математики, д. т. н., професор

_____ Гребенюк С.М.

(підпис)

« ____ » _____ 2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Старовойтова Олександра Вікторовича

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Автоматизація дослідження основних властивостей складних керованих динамічних систем

керівник роботи (проекту) Леонтьєва Вікторія Володимирівна, к.ф.-м.н., доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від ««26» січня 2023 року № 120-с

2. Строк подання студентом роботи 15.06.2023

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 26.01.2023

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	27.01.2023	
2.	Збір вихідних даних.	01.02.2023	
3.	Обробка методичних та теоретичних	06.03.2023	
	джерел.		
4.	Розробка першого розділу.	03.04.2023	
5.	Розробка другого розділу.	04.05.2023	
6.	Оформлення та нормоконтроль	16.06.2023	
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	22.06.2023	

Студент _____ О.В.Старовойтов
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.В. Леонтьєва
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Автоматизація дослідження основних властивостей складних керованих динамічних систем»: 80 с., 23 рис., 41 джерело, 2 додатки.

АВТОМАТИЗАЦІЯ, ДИСКРЕТНА МАТЕМАТИЧНА МОДЕЛЬ, КЕРОВАНА СИСТЕМА, КЕРОВАНІСТЬ, ПОЗИТИВНА ДИНАМІЧНА СИСТЕМА, ПОЗИТИВНІСТЬ.

Об’єкт дослідження – складна позитивна динамічна система.

Мета роботи: розробка методики дослідження та реалізуючої її програми для автоматизації процесу моделювання та аналізу основних властивостей складної керованої позитивної динамічної системи.

Метод дослідження – аналітичний.

У роботі розглядається дискретна математична модель складної позитивної динамічної системи, описувана лінійним неоднорідним векторно-матричним різницевим рівнянням із матрицями сталих коефіцієнтів, описуються основні обмеження математичної моделі та доступні керування, що надають можливості як стабілізувати нестійкі системи, так і покращувати їх динамічні властивості.

За проведеним дослідженням розглядуваної моделі в роботі пропонується методика проведення аналізу математичної моделі та основних властивостей складної керованої позитивної динамічної системи й отримуваних за нею результатів.

На основі пропонованої методики дослідження в роботі проводиться розробка програмного забезпечення, яке дозволяє автоматизувати процес виконання широкого спектру трудомістких обчислень, проведення аналізу основних характеристик моделі та властивостей складної системи.

SUMMARY

Bachelor’s qualifying paper «Automation of research of the main properties of complex controlled dynamical systems»: 80 pages, 23 figures, 41 references, 2 supplements.

AUTOMATION, DISCRETE MATHEMATICAL MODEL, CONTROLLABLE SYSTEM, CONTROLLABILITY, POSITIVE DYNAMICAL SYSTEM, POSITIVITY.

The object of the study is a complex positive dynamical system.

The aim of the study is development of a research methodology and its implementation program for automating the process of modeling and analysis of the main properties of a complex controlled positive dynamical system.

The method of research is analytical.

The qualification paper considers a discrete mathematical model of a complex positive dynamical system, described by a linear inhomogeneous vector-matrix difference equation with matrices of constant coefficients, describes the main limitations of the mathematical model and available controls that provide opportunities both to stabilize unstable systems and to improve their dynamical properties.

Based on the study of the considered model, the paper proposes a method for analyzing the mathematical model and the main properties of a complex controlled positive dynamical system and the results obtained from it.

Based on the proposed research methodology, software is being developed in the work, which allows automating the process of performing a wide range of time-consuming calculations, analyzing the main characteristics of the model and the properties of a complex system.

ЗМІСТ

ВСТУП

У сучасному світі велика кількість складних керованих динамічних систем потребує детального дослідження їх основних властивостей для ефективного проектування, управління та оптимізації. Однак, проведення таких досліджень може бути вимогливим і часом витратним процесом. Тому, розробка автоматизованої системи для дослідження складних керованих динамічних систем стає актуальною задачею, яка спрямована на забезпечення швидкості, ефективності та точності досліджень.

Метою роботи є розробка методики дослідження та реалізуючої її програми для автоматизації процесу моделювання та аналізу основних властивостей складної керованої позитивної динамічної системи. При цьому важливим аспектом є забезпечення точності та ефективності досліджень, що дозволить покращити процес проектування та оптимізації досліджуваних складних систем.

Об'єктом дослідження в роботі виступає складна керована позитивна динамічна система, поведінка якої описується дискретною математичною моделлю, що представляється лінійним неоднорідним векторно-матричним різницевим рівнянням із матрицями сталих коефіцієнтів та може включати в себе елементи з різних галузей, такі як техніка, економіка, біологія тощо. Дослідження проводяться з метою аналізу та оцінки основних властивостей зазначеної системи, таких як позитивність, асимптотична стійкість розв’язків та керованість системи.

Для реалізації поставленої в роботі мети необхідно виконання наступних завдань дослідження:

а) аналіз вхідних даних та основних складових математичних моделей, які застосовуються для опису складних керованих динамічних систем, зокрема, дискретної моделі складної позитивної системи;

б) виокремлення основних обмежень дискретної моделі досліджуваної системи, що забезпечують виконання властивості її позитивності та асимптотичної стійкості (за Ляпуновим) отримуваних за моделлю розв’язків на нескінченному інтервалі часу;

в) дослідження керованості досліджуваної динамічної системи;

г) визначення доступних для досліджуваної позитивної системи керувань, здатних як стабілізувати нестійкі системи, так і покращити їх динамічні властивості;

д) розробка методики проведення аналізу математичної моделі та основних властивостей складної керованої позитивної динамічної системи й отримуваних за нею результатів;

е) розробка та реалізація програмного засобу, що дозволяє проводити автоматизовані дослідження на основі розробленої методики.

Структурно робота складається з 3 розділів.

Перший розділ присвячено висвітленню основних теоретичних відомостей, розкриттю понять та визначень теорії автоматичного керування, що використовуються при описі складної керованої динамічної системи. Крім того, в даному розділі наводяться результати аналізу основних доступних методів розв'язування систем рівнянь математичних моделей динамічних систем, на окремі з яких спирається пропонована у другому розділі роботи методика проведення аналізу математичної моделі та основних властивостей складної керованої позитивної динамічної системи й отримуваних за нею результатів.

У другому розділі наводяться основні характеристики об’єкта дослідження – складної керованої позитивної динамічної системи – та використовуваної для опису її поведінки дискретної математичної моделі, наводяться основні вимоги до математичної моделі та засоби їх досягнення, розкриваються доступні керування для позитивної системи та виникаючі на їх основі задачі керування, представляються основні етапи пропонованої в роботі методики проведення аналізу математичної моделі та властивостей

позитивності й керованості складної позитивної динамічної системи й отримуваних за нею результатів, на якій ґрунтується реалізація розроблюваного в роботі програмного забезпечення, що дозволяє автоматизувати процес здійснення відповідних досліджень.

Третій розділ присвячений висвітленню особливостей програмної реалізації розробленого програмного продукту за представленою методикою здійснення дослідження, виділенню основних переваг та особливостей використання обраної мови програмування, наданню детального опису інтерфейсу та інструкції з використання створеного продукту для кінцевого користувача. Також в даному розділі наводяться результати проведеного обчислювального експерименту на моделі із використанням розробленої програмної системи.

Отримані в кваліфікаційній роботі результати є важливими з точки зору наукових досліджень і практичного застосування. Розроблене програмне забезпечення може бути використано при дослідженні промислових, економічних, біологічних, екологічних процесів, у робототехніці, системах керування й регулювання і т. ін.

1 ДИНАМІЧНІ СИСТЕМИ: ОСНОВНІ ПОНЯТТЯ ТА ВИЗНАЧЕННЯ, МАТЕМАТИЧНІ МОДЕЛІ РУХУ ТА ОСНОВНІ МЕТОДИ ДОСЛІДЖЕННЯ

З метою розкриття основних характеристик складних керованих динамічних систем в даному розділі надаються основні поняття та визначення теорії автоматичного керування та теорії динамічних систем, які в подальшому використовуватимуться при описі об’єкта дослідження, наводяться огляд основних видів математичних моделей для опису поведінки динамічних систем та результати аналізу основного математичного інструментарію, використовуваного при аналізі розроблюваних моделей та отриманні розв’язків за основними рівняннями моделей.

1.1 Основні поняття та визначення, використовувані при описі складних керованих динамічних систем

В роботі розглядаються керовані динамічні системи, під якими, згідно до [3, 9, 16, 17, 39, 40], розуміються системи, які складаються з динамічних елементів, які можуть змінювати свій стан відповідно до вхідного сигналу та зворотного зв'язку, що забезпечує керування їх поведінкою та динамікою. Зазначеного виду динамічні системи використовуються для розв'язання різних задач у багатьох галузях, таких як автоматичне керування виробництвом, автомобільної промисловості, аерокосмічної техніки, медичних пристроїв тощо [1, 9, 16, 19-22, 33, 38-41].

При цьому потрібно зауважити, що керування динамічними системами зазвичай здійснюється за допомогою різних алгоритмів та методів, таких як PID-регулятори, керування оптимальними траєкторіями, адаптивне керування та ін. Вони дозволяють забезпечити бажану динаміку та стійкість

системи, зменшити витрати енергії, покращити точність та швидкість роботи системи [1, 4-7, 9, 21, 22, 25-29, 34].

Крім того, оскільки складні керовані динамічні системи складаються з великої кількості динамічних елементів, які можуть змінювати свій стан відповідно до вхідного сигналу та зворотного зв'язку і які взаємодіють між собою, ці системи можуть мати велику кількість взаємопов'язаних підсистем, що робить їх дуже складними для аналізу та керування. Поряд з цим, вони можуть також містити нестабільні або нелінійні елементи, які додатково ускладнюють процес керування [6, 7, 31, 32, 39].

Прикладами складних керованих динамічних систем можуть бути комп'ютерні мережі, екосистеми, фінансові ринки, транспортні системи та інші складні системи [1, 5, 9, 16, 19-22, 31, 33, 38-41].

Для керування складними керованими динамічними системами використовуються різні методи та алгоритми, такі як нейромережеве керування, адаптивне керування, керування розподіленими системами та інші. Крім того, важливо використовувати методи аналізу систем, щоб зрозуміти їх поведінку та динаміку [5-7, 32, 39].

У «складних керованих динамічних системах» термін «керовані» означає, що система містить підсистему або механізм, який здійснює керування її поведінкою. Це може бути програмне забезпечення, електронне устаткування, системи керування рухом або будь-який інший засіб, який забезпечує зміну стану системи для досягнення певної мети [1, 4-7, 9, 30-32].

Керування в таких системах може здійснюватися різними способами, включаючи зворотний зв'язок, передачу сигналів, програмування, оптимальне керування та інші [39]. Метою керування є забезпечення оптимальної поведінки системи з точки зору певної метрики ефективності, такої як мінімізація енергоспоживання, максимізація продуктивності або забезпечення стабільності системи в умовах зміни параметрів.

Отже, термін «керовані» вказує на наявність системи керування, що дозволяє змінювати поведінку системи для досягнення певної мети, що є

важливим аспектом в аналізі та керуванні складними динамічними системами.

До керованих складних динамічних систем відноситься й об’єкт дослідження кваліфікаційної роботи – складні позитивні динамічні системи, які відзначаються особливою властивістю, відомою як позитивність, за якою вхідні та вихідні змінні моделей зазнаеного об'єкту (позитивні змінні), залишаються невід'ємними протягом всього досліджуваного часу. Характеристика позитивних систем та використовуваних для опису їх руху математичних моделей разом із вимогами, що забезпечують збереження властивості системи будуть надані у 2 розділі роботи.

1.2 Огляд основних видів математичних моделей, які застосовуються для опису руху складних динамічних систем

Математичні моделі відіграють важливу роль у вивченні складних керованих динамічних систем, оскільки дозволяють описати поведінку цих систем та встановити взаємозв'язки між їх компонентами. У данному підрозділі роботи проводиться огляд різних типів математичних моделей, які використовуються для опису складних динамічних систем, в тому числі і керованих.

Починаючи з класичних диференціальних рівнянь, що базуються на законах збереження маси, речовини та енергії, існують також багато інших типів моделей. До них відносяться стохастичні моделі, які враховують випадкові фактори, логістичні моделі, що описують зміну популяцій в часі, та моделі зворотного розповсюдження, що розглядають поширення інформації в мережах [25, 29, 31, 32]. Крім того, розглядаються інтегральні моделі, які використовуються для опису накопичення величин часом, та оптимізаційні моделі, які спрямовані на пошук оптимальних рішень у складних системах [8, 27, 28]. Кожен тип моделі має свої особливості та

використовується залежно від конкретних властивостей та особливостей досліджуваної системи.

Моделювання та дослідження складних об'єктів включає їх опис у вигляді систем диференціальних, різницевих рівнянь і т. ін. великої розмірності. Зосередимось на характеристиці дискретної та неперервної моделей, як основних видах моделей, використовуваних при описі поведінки складних керованих динамічних систем. Дискретна модель при цьому ще й виступає предметом розгляду кваліфікаційної роботи при описі поведінки складної керованої позитивної динамічної системи.

Перш за все, надамо характеристику дискретної моделі. Така модель базується на дискретизації часу та стану системи, розглядає систему в окремі моменти часу та визначає значення стану системи в ці моменти [19, 20, 39]. Одним з основних елементів дискретної моделі є дискретний часовий шаг, який визначає інтервал між послідовними моментами часу, в яких відбувається оновлення стану системи. Частота оновлення може бути фіксованою або змінною в залежності від характеристик системи та вимог дослідження.

Дискретна модель використовує рекурентні співвідношення для визначення залежності між поточним та попереднім станами системи. Це дозволяє враховувати вплив попереднього стану на подальший розвиток системи. Для обчислення значень стану системи використовуються алгоритми та методи, що базуються на математичних розрахунках та апроксимаціях.

Однією з переваг дискретної моделі є її обчислювальна ефективність. Вона дозволяє проводити чисельні розрахунки та аналіз системи на комп'ютері швидше, ніж у випадку неперервної моделі. Крім того, дискретна модель може бути більш гнучкою та адаптивною до змін в системі, оскільки дозволяє змінювати часовий крок та оновлювати стан системи в потрібних точках.

Надалі розглянемо найбільш часто використовувану при математичному моделюванні складних об’єктів форму математичної моделі – неперервну математичну модель, яка базується на неперервному часі та стані системи, що змінюються плавно впродовж часу [1, 4-7, 21, 31, 32, 39]. Неперервна модель використовує диференціальні або інтегральні рівняння для опису залежностей між змінними стану системи та їх похідними. Ці рівняння враховують зміну стану системи від моменту до моменту часу, дозволяючи аналізувати динаміку системи в неперервному часовому проміжку. В результаті неперервна математична модель дозволяє отримати більш детальний та точний опис системи, але вимагає більшої обчислювальної потужності та ресурсів.

1.3 Основні методи дослідження математичних моделей та їх результатів

1.3.1 Огляд методів, які використовуються для розв'язування систем рівнянь математичних моделей динамічних систем

У даному пункті роботи наводяться результати проведеного аналізу основних методів дослідження математичних моделей складних динамічних систем та отримуваних за цими моделями результатів. Перш за все розкриємо сутність та особливості використання чисельних методів, що застосовуються для розв'язування систем рівнянь математичних моделей досліджуваних складних керованих динамічних систем.

Найбільш поширеним видом чисельних методів виступає сукупність методів чисельного розв'язування диференціальних рівнянь, які є основою багатьох математичних моделей динамічних систем [10, 11]. Ця група методів включає в себе апроксимацію рівнянь, ітераційні алгоритми та методи для знаходження значень змінних стану системи в різних часових

точках. Чисельне розв'язування диференціальних рівнянь дозволяє отримати числові результати, які відображають поведінку системи в часі. Крім того, досить часто в практиці наукових досліджень керованих динамічних систем застосовуються й інші чисельні методи, такі як методи інтегрування, оптимізації та аналізу стійкості системи. Ці методи використовуються для вивчення особливостей та властивостей математичних моделей динамічних систем, а також для знаходження оптимальних рішень та оцінки стійкості системи в різних умовах.

Для розв'язування різницевих рівнянь використовуються різні чисельні методи залежно від характеру задачі та потреб користувача. Розглянемо найбільш розповсюджені та найбільш вживані такі методи.

Метод скінченних різниць є одним з найпоширеніших і простих чисельних методів [11]. Він базується на апроксимації похідних у різницевих формулах та розбитті простору на скінченну кількість точок. За допомогою цього методу різницеве рівняння перетворюється на систему алгебраїчних рівнянь, яку можна вирішити чисельно.

Метод скінченних елементів використовує апроксимацію функцій на скінченних елементах, що дозволяє більш точно відображати поведінку системи. Простір розбивається на дискретні підобласті, і на кожній підобласті функції апроксимуються за допомогою базисних функцій. Цей метод часто використовується для задач зі складною геометрією та змінними властивостями середовища.

Методи скінченних різниць в часі використовуються для апроксимації похідних за часом. Вони особливо корисні при моделюванні динамічних систем, де час є важливим фактором.

Крім чисельних методів при дослідженні складних керованих динамічних систем, в тому числі й позитивних систем, використовується й аналітичний метод, який базується на розв'язанні різницевих рівнянь за допомогою аналітичних формул [19, 20, 39]. Аналітичний метод розв'язання різницевих рівнянь також є одним з основних підходів у математичному

моделюванні динамічних систем. Цей метод ґрунтується на застосуванні аналітичних формул для отримання точних розв'язків рівнянь, без необхідності використовувати складні чисельні апроксимації.

Головна перевага аналітичного методу полягає в тому, що він дозволяє отримувати точні розв'язки рівнянь, що спрощує інтерпретацію та аналіз результатів. Аналітичні формули дозволяють вивчити поведінку системи в різних умовах та змінних параметрах. Крім того, аналітичний підхід дозволяє ефективно виконувати параметричний аналіз та досліджувати вплив різних факторів на систему.

Використання аналітичного методу також забезпечує високу швидкість обчислень, оскільки розв'язки отримуються без застосування ітераційних алгоритмів. Це особливо корисно при моделюванні складних систем з багатьма змінними та рівняннями. Однак, варто враховувати, що аналітичний метод може бути обмежений складністю самого рівняння або відсутністю аналітичних розв'язків для деяких задач. У таких випадках може бути доцільно поєднувати аналітичний метод з чисельними методами для отримання більш загальних результатів.

Застосування аналітичного методу у розробленій програмній системі дозволяє отримати точні розв'язки різницевих рівнянь, що сприяє більш детальному та повному дослідженню складних керованих динамічних систем.

1.3.2 Обґрунтування вибору методу розв’язання системи рівнянь використовуваної математичної моделі складної системи

Обґрунтування вибору конкретного методу для використовуваної при описі поведінки досліджуваної динамічної системи математичної моделі є важливим етапом дослідження. При виборі використовуваного методу розв’язання системи рівнянь відповідної моделі необхідно враховувати

особливості самої моделі, а також поставлену задачу керування. При цьому, якщо можливо використання аналітичного методу, то, завжди перевага віддається йому, в противному випадку використовується математичнмй інструментарій чисельного розв’язання задач.

При виборі такого інструментарію виходять з наступних міркувань:

а) у випадках, де точність розв'язку не є першорядним завданням, можна використовувати простіші чисельні методи, до яких в першу чергу відносять метод Ейлера [10, 11]. Цей метод має просту структуру та обчислювальну ефективність, може бути практично застосований, коли розрахунки проводяться на невеликому кроці часу та не вимагають високої точності;

б) у випадках, де точність розв'язку є важливою і дослідник прагне отримати більш точні результати, рекомендується використовувати більш складні чисельні методи, наприклад, метод Рунге-Кутта [10]. Цей метод дозволяє отримати більш точні результати, оскільки включає в себе додаткові кроки розрахунку на кожному ітераційному кроці.

Вибір конкретного чисельного методу залежить від поставленої задачі, вимог до точності розв'язку та обчислювальних обмежень. Враховуючи ці фактори, дослідник може обґрунтувати вибір певного чисельного методу для розглянутої математичної моделі керування.

Підсумовуючі наведену у даному розділі інформацію, зазначимо, що висвітлені основні теоретичні відомості, поняття та визначення теорії автоматичного керування лежать в основі опису складної керованої динамічної системи, до якої, зокрема відноситься і об’єкт дослідження роботи – складна керована позитивна динамічна система. Крім того, наведені в даному розділі результати аналізу основних доступних методів розв'язування систем рівнянь математичних моделей динамічних систем надають можливості вибору саме того математичного інструментарію, що найбільш відповідає специфіці поставленої задачі дослідження та затребуваної обчислювальної точності розрахунків.

2 СКЛАДНА ПОЗИТИВНА ДИНАМІЧНА СИСТЕМА ЯК ОБ'ЄКТ ДОСЛІДЖЕННЯ ТА АВТОМАТИЗАЦІЇ

В даному розділі наводиться характеристика об'єкта дослідження, в якості якого обрано позитивну динамічну систему, надаються основні відомості з використовуваної для опису його поведінки дискретної математичної моделі з наведенням основних вимог та обмежень, що забезпечують виконання властивості позитивності системи, асимптотичної стійкості отримуваних за моделлю розв'язків, а також виокремлюються основні види доступних керувань, здатних як стабілізувати нестійкі й непозитивні системи, так і покращити динамічні властивості об'єкта дослідження. Крім того, за результатами аналізу об'єкта дослідження та його математичної моделі в даному розділі представляється методика проведення аналізу математичної моделі та основних властивостей складної керованої позитивної динамічної системи й отримуваних за нею результатів.

2.1 Характеристика позитивної динамічної системи та використовуваної для опису її поведінки дискретної математичної моделі

В якості об'єкта дослідження в роботі виступає складна керована позитивна динамічна система, під якою розуміється система із збереженням властивості позитивності вхідних характеристик об'єкта, змінних моделі та отримуваних вихідних характеристик об'єкта (та їх фазових траєкторій) на всьому досліджуваному проміжку часу [2, 19-22, 38, 41]. До таких систем відноситься багато складних об'єктів, наприклад, біологічні, економічні, екологічні системи [19-22, 38, 41]. Головним при йьому є володіння зазначеною системою властивістю позитивності, за якою вхідні та вихідні

змінні моделей цих об'єктів (позитивні змінні), залишаються невід'ємними. Прикладами позитивних змінних є час, гроші та товари, потоки даних у мережах, населення людей, тварин і рослин, електричні заряди, рівні інтенсивності світла, імовірності в марківських моделях та інше.

В якості математичної моделі досліджуваної позитивної моделі обрано дискретну модель, описувану лінійним неоднорідним векторно-матричним різницевим рівнянням із матрицями сталих коефіцієнтів виду [20]

$$X_{t+1} = (I - B)^{-1}(A - B)X_t + (I - B)^{-1}C_t, \qquad (2.1)$$

або

$$X_{t+1} = \tilde{A}X_t + \tilde{B}C_t, \qquad (2.2)$$

де час t обирається дискретним;

n – кількість агрегованих взаємозв'язаних між собою підсистем досліджуваної складної позитивної системи;

X_t, X_{t+1} – n – мірний вектор стану позитивної динамічної системи;

C_t – n – мірний вектор керування позитивної динамічної системи;

$\tilde{A}, \tilde{B}$ – сталі матриці відповідно стану та керування системи розмірностей $n \times n$:

$$\tilde{A} = (I - B)^{-1}(A - B),$$
$$\tilde{B} = (I - B)^{-1};$$

A, B – сталі матриці розмірностей $n \times n$:

$$A = (a_{ij})_{n \times n},\ B = (b_{ij})_{n \times n}.$$

При цьому в моделях, описуваних рівняннями (2.1) або (2.2), $n-$мірний вектор-функція керування C_t позитивної динамічної системи у роботі розглядається в одному з трьох видах [19]:

а) лінійному:

$$C_t = C^0 + Ct;$$

б) квадратичному:

$$C_t = C^0 + Ct^2;$$

в) гармонійному:

$$C_t = C^0 + C\sin\omega t,$$

де $C^0, C, \omega -$ параметри регресійної моделі, отриманої в результаті обробки експериментальних даних досліджуваного об’єкта із використанням техніки регресійного аналізу.

Для забезпечення позитивності досліджуваної системи та отримання асимптотично стійких (за Ляпуновим) розв’язків за використовуваною математичною моделлю руху системи, на коефіцієнти матриць моделі накладаються певні обмеження, найбільш ґрунтовно описані в роботі [19-22]: матриці

$$A,\ B,\ (A-B),\ \tilde{A} = (I-B)^{-1}(A-B)$$

мають бути невід’ємними, належати до класу *M*-матриць Мецлера та бути продуктивними.

При цьому критерієм продуктивності сталої квадратної матриці, наприклад, матриці S розуміється виконання однієї з наступних необхідних та достатніх умов продуктивності [19, 20, 21]:

а) існує обернена матриця

$$(I - S)^{-1} \geq 0,$$

де $I -$ одинична матриця розмірності $n \times n$;

б) матричний ряд виду

$$I + S + S^2 + S^3 + \ldots = \sum_{k=0}^{\infty} S^k$$

збігається до зворотної матриці $(I - S)^{-1}$;

в) найбільше за модулем власне значення $\rho(S)$ характеристичного рівняння $|S - \rho I| = 0$ є строго меншим за одиницю;

г) послідовні головні мінори порядку від 1 до n матриці $(I - S)$ додатні.

Для рівняння (2.1) або (2.2) ставиться задача Коші:

$$\text{при } t = t_0 = 0 \quad X(0) = X_0, \quad i = \overline{1, n}, \tag{2.3}$$

де $X(0) = X_0 - \ n -$ мірний вектор-стовпець початкових станів системи.

Описана дискретна динамічна математична модель, що подається векторно-матричним різницевим рівнянням (2.1) або (2.2), є розімкненою: в ній зміна вихідної характеристики X_{t+1} не впливає на вектор-функції X_t й C_t досліджуваної динамічної системи. Розв'язок такого рівняння шукається ітераційним шляхом [20, 39] та подається, згідно до роботи [19, 20], у вигляді

$$X_t = \left((I-B)^{-1}(A-B)\right)^t X_0 + \sum_{i=1}^{t}\left((I-B)^{-1}(A-B)\right)^{t-i}(I-B)^{-1}C_{i-1}. \quad (2.4)$$

Саме отриманий розв'язок (2.4) векторно-матричного різницевого рівняння (2.1) або (2.2) використовується в розроблюваному програмному коді при визначенні основних характеристик досліджуваної позитивної динамічної системи.

Якщо всі перераховані умови в моделі, яка описується векторно-матричним різницевим рівнянням (2.1) або (2.2), виконуються, то система (2.11) або (2.2) буде асимптотично стійкою (за Ляпуновим), а одержуваний на виході моделі вектор X_t при $t \to \infty$ буде збігатися до рівноважного (стаціонарного) значення X^*, яке визначається при $\Delta X_t = X_{t+1} - X_t = \Theta$ (де $\Theta = (\theta_{ij})_{n\times n}$, $\theta_{ij} = 0$ для $\forall i, j = \overline{1,n}$) і $C_t = C^0 - const$, є розв'язком статичної задачі В.В. Леонтьєва та надається у вигляді [20]

$$X^* = (I-A)^{-1}C^0. \quad (2.5)$$

У випадку, коли окремі вимоги до зазначених сталих матриць дискретної математичної моделі позитивної динамічної системи не задовольняються, або потрібним є покращення динамічних властивостей досліджуваної системи, застосовується математичний апарат теорії автоматичного керування й регулювання [21]. При цьому основним при такому дослідженні виступає вимога володіння позитивною системою властивістю повної керованості, за якою основні дослідження проведено у наступному підрозділі 2.2 роботи. Також у цьому підрозділі виділяються доступні керування, застосовувані до системи в окреслених випадках.

2.2 Керування у дискретній математичній моделі позитивної динамічної системи

2.2.1 Керованість позитивної динамічної системи

У даному пункті роботи надаються основні результати, на яких ґрунтується проведення перевірки однієї з найважливіших властивостей позитивної динамічної системи, розглядуваної в якості системи керування, – властивості керованості системи.

Керованість об'єкта (системи) керування означає можливість переведення об'єкта з будь-якого початкового стану (режиму роботи) $X(t_0) = X_0$ у будь-який кінцевий стан $X(t_k) = X_k$ за кінцевий дискретний час шляхом застосування допустимого керування C_t [22, 39]. При цьому для переведення системи достатньо, щоб такий кінцевий стан був асимптотично стійким для всього простору станів [22].

Система (об'єкт) керування, що володіє властивістю керованості у вказаному сенсі, вважається повністю керованою [21, 22, 39]. Оцінка керованості проводиться на основі алгебраїчного критерію Р. Калмана [39], за яким необхідною та достатньою умовою повної керованості позитивної динамічної системи з дискретним часом, що описується відповідно дискретною динамічною математичною моделлю, є рангова умова [22]

$$rang\left[\tilde{B} : \left(\tilde{A}\tilde{B}\right) : \left(\tilde{A}^2\tilde{B}\right) : \ldots : \left(\tilde{A}^{n-1}\tilde{B}\right)\right]_{n\times nn} = n,$$

за якою ранг блокової матриці керованості

$$M_{кер} = \left[\tilde{B} : \left(\tilde{A}\tilde{B}\right) : \left(\tilde{A}^2\tilde{B}\right) : \ldots : \left(\tilde{A}^{n-1}\tilde{B}\right)\right]_{n\times nn}$$

збігається з розмірністю n простору станів системи, або, інакше кажучи, в матриці керованості $M_{кер}$ є n лінійно-незалежних векторів-стовпців.

Отже, згідно з критерієм Р. Калмана, стан дискретної системи є повністю керованим, якщо і тільки якщо ранг блокової матриці керованості $M_{кер}$ дорівнює розмірності простору станів n досліджуваної системи. В противному випадку, якщо ранг матриці керованості $M_{кер}$ є меншим n, досліджувана система є не керованою (або частково керованою).

2.2.2 Типи керувань позитивною системою

У представленій дискретній математичній моделі позитивної динамічної системи існують різні компоненти, що безпосередньо впливають на керовану систему: матриці A й B коефіцієнтів моделей і функції C_t.

Залежно від цих компонентів виділяються різні типи керування [21]:

а) керування за допомогою матриці A – використовується для коригування та поліпшення вхідних параметрів та вихідних характеристик об'єкта дослідження;

б) керування за допомогою матриці B – використовується для прискорення збіжності вихідних характеристик X_t до рівноважного стану X^* об'єкта;

в) керування за допомогою функції C_t – застосовується, коли неможливо змінити матриці A і B, або їх вплив на моделі є недостатнім.

Враховуючи цілі та типи керування для дискретної моделі позитивної динамічної системи балансового типу, можна розглянути наступні задачі керування:

а) у випадку, коли матриця A вихідного рівняння (2.1) або (2.2), є продуктивною і може бути змінена, застосування керування за допомогою матриці A дозволяє покращити вихідні характеристики системи шляхом їх зменшення або збільшення;

б) якщо матриця A вихідної системи є непродуктивною, побудова замкненої моделі (моделі зі зворотним зв'язком) дозволяє отримати потрібний вигляд цієї матриці;

в) у випадку, коли необхідно отримати покращені значення вихідних характеристик системи, які не досягаються розімкненою моделлю (коли неможливо змінити матрицю A), необхідно використовувати модель зі зворотним зв'язком і задати лінійний регулятор;

г) у випадку, коли за дискретною моделлю, що описує поведінку об'єкта, при продуктивній матриці A необхідно прискорити процес збіжності вихідних характеристик та покращити їх значення, виникає потреба в зміні матриці B за допомогою керування за допомогою матриці B;

д) у випадку, коли задано бажаний вектор станів системи, застосування методів теорії програмного керування дозволяє визначити такі керування, які дозволять досягти бажаного стану системи.

Розв’язання зазначених задач керування реалізовано викладеній в наступному п. 2.3 роботи методиці проведення аналізу математичної моделі та основних властивостей досліджуваної динамічної системи й отримуваних за нею результатів та в розробленому на її основі програмному забезпечені.

2.3 Методика проведення аналізу математичної моделі та основних властивостей складної керованої позитивної динамічної системи й отримуваних за нею результатів

За результатами дослідження, проведеного у попередніх підрозділах 2.1 та 2.2 даного розділу роботи, розроблено загальну методику проведення аналізу дискретної математичної моделі та основних властивостей складної керованої позитивної динамічної системи й отримуваних за нею результатів.

Пропонована така методика визначення і дослідження змін вихідних характеристик модельованого об’єкта включає наступні етапи:

І етап: Аналіз компонентів дискретної математичної моделі об'єкта дослідження. Даний етап включає:

а) дослідження на невід'ємність та продуктивність вхідних параметрів дискретної математичної моделі об'єкта, а саме матриць коефіцієнтів моделі. При цьому на невід'ємність та продуктивність досліджуються матриці

$$A,\ B,\ (A-B),\ \tilde{A}=(I-B)^{-1}(A-B).$$

Виконання умов невід'ємності й продуктивності вказаних матриць забезпечують позитивність системи та асимптотичну стійкість одержуваних розв'язків.

Якщо матриця A є непродуктивною, то здійснюється перехід до етапу IV, підпункту б) пункту 1);

б) дослідження на невід'ємність вхідних вектор-функцій дискретної математичної моделі об'єкта C_t, які в роботі розглядаються в лінійному, квадратичному й гармонійному видах відповідно:

$$C_t = C^0 + Ct;$$

$$C_t = C^0 + Ct^2;$$

$$C_t = C^0 + C\sin\omega t,$$

де $C^0, C, \omega -$ параметри регресійної моделі, отриманої в результаті обробки експериментальних даних досліджуваного об'єкта із використанням регресійного аналізу.

Виконання умов невід'ємності вказаних вектор-функцій забезпечують позитивність системи.

Якщо вектор-функції C_t є від'ємними, то дослідникові надається можливість або корегування параметрів цих функцій або здійснюється перехід до етапу II;

в) дослідження властивості керованості об’єкта дослідження [22].

На даному кроці проводиться перевірка однієї з найважливіших властивостей позитивної динамічної системи, розглядуваної в якості системи керування, – властивості керованості системи.

Оцінка керованості проводиться на основі алгебраїчного критерію Р. Калмана, за яким необхідною та достатньою умовою повної керованості позитивної динамічної системи з дискретним часом є рангова умова

$$rang\left[\tilde{B}:\left(\tilde{A}\tilde{B}\right):\left(\tilde{A}^2\tilde{B}\right):\ldots:\left(\tilde{A}^{n-1}\tilde{B}\right)\right]_{n\times nn}=n,$$

за якою ранг матриці керованості

$$M_{кер}=\left[\tilde{B}:\left(\tilde{A}\tilde{B}\right):\left(\tilde{A}^2\tilde{B}\right):\ldots:\left(\tilde{A}^{n-1}\tilde{B}\right)\right]_{n\times nn}$$

збігається з розмірністю n простору станів системи.

В результаті аналізу системи стан дискретної системи вважатиметься повністю керованим, якщо і тільки якщо ранг матриці керованості $M_{кер}$ дорівнює розмірності простору станів n досліджуваної системи. В такому випадку здійснюється перехід до ІІ етапу.

В противному випадку, якщо ранг матриці керованості $M_{кер}$ є меншим кількості n підсистем позитивної системи, досліджувана система є не керованою (або частково керованою). В такому випадку здійснюється перехід до етапу IV, підпункту б) пункту 1).

ІІ етап: Перевірка завдання бажаного стану об’єкта дослідження.

Якщо такий стан не заданий, то здійснюється перехід на етап ІІІ.

Якщо ж бажаний стан

$$X_t^*=\varphi_t\geq 0$$

є відомим, то розв'язується задача визначення програмних керувань C_t, які виводять систему на бажану траєкторію руху, після чого здійснюється перехід до V етапу.

Шуканий вектор C_t керувань визначаються з рівняння для дискретного часу у вигляді

$$C_t = (I - B)(\Delta\varphi_t - (A - I)\varphi_t) \geq 0,$$

де

$$\Delta\varphi_t = \varphi_{t+1} - \varphi_t .$$

Знайдені таким чином керуючі функції $C_t^1,...,C_t^n$ є програмними, оскільки вони змінюються відповідно з заданою вектор функцією

$$X_t^* = \varphi_t, \quad 0 < t < T, \quad 0 < T < \infty .$$

Система керування в такому випадку є розімкненою.

З урахуванням знайденого керування C_t процес в розглядуваній системі описується векторно-матричним різницевим рівнянням виду

$$X_{t+1} = (I - B)^{-1}(A - B)X_t + f_t,$$

де f_t знаходиться залежно від заданої траєкторії φ_t у вигляді

$$f_t = \Delta\varphi_t - (A - I)\varphi_t .$$

Розв'язок такого різницевого рівняння за заданих початкових умов $X(0) = X_0$ в точності співпадає з назначеною траєкторією φ_t руху системи.

Даний підхід розповсюджується на замкнені моделі, які описується на етапі IV, підпункті б) пункту 1), шляхом заміни матриці A у вказаних системах матрицею $(A-k)$, де $k-$ матриця коефіцієнтів посилення зворотного зв'язку.

Зазначимо, що за емпіричними даними реальної досліджуваної системи шляхом застосування викладеного підходу до визначення програмних керувань можливе знаходження вхідних вектор-функцій C_t розімкненої та замкненої дискретних моделей позитивної динамічної системи балансового типу (для яких зазначена вектор-функція C_t вважається відомою).

III етап: Визначення вихідних характеристик об'єкта моделювання X_t при наявних вхідних змінних C_t і початкових умовах $X(0) = X_0$ об'єкта за розімкненою дискретною моделлю позитивної динамічної системи балансового типу.

Для цього використовуються наступні співвідношення:

$$X_t = \left((I-B)^{-1}(A-B)\right)^t X_0 + \sum_{i=1}^{t}\left((I-B)^{-1}(A-B)\right)^{t-i}(I-B)^{-1}C_{i-1}$$

або

$$X_t = \tilde{A}^{\,t-1}\tilde{B}\left((A-B)X_0 + (I-A)X^*\right) + \tilde{A}^{\,t-1}\sum_{i=1}^{t-1}\tilde{A}^{\,-i}\tilde{B}C_i\,,$$

де

$$\tilde{A} = \tilde{B}(A-B), \quad \tilde{B} = (I-B)^{-1}.$$

IV етап: Дослідження результатів, одержаних на етапі III. Якщо одержані вихідні характеристики об'єкта моделювання не задовольняють дослідника (замовника), тоді з множини можливих керувань обирається такий вплив на систему, який забезпечує досягнення поставленої мети. Даний етап включає наступне:

а) при необхідності покращення вхідних параметрів і вихідних характеристик об'єкта дослідження проводиться керування за допомогою початкової матриці A, причому:

1) якщо матриця $A \geq 0$ є продуктивною і при цьому її можливо змінювати, то покращення вихідних характеристик об'єкта здійснюється безпосереднім вибором елементів матриці A так, щоб матриця A залишалася продуктивною:

– для збільшення значень кожного з елементів вектора X_t до граничних значень $C_0^i \quad \left(i = \overline{1,n}\right)$ вектора $C^0 = \left(C_0^1, C_0^2, \ldots, C_0^n\right)^T$ елементи матриці A повинні наближуватися до одиниці (причому дана ситуація справедлива, якщо $C_0^i \neq 0$);

– для зменшення значень кожного з елементів вектора X_t до граничних значень $C_0^i \quad \left(i = \overline{1,n}\right)$ вектора $C^0 = \left(C_0^1, C_0^2, \ldots, C_0^n\right)^T$ елементи матриці A повинні наближуватися до нуля.

Далі переходимо до пункту б) етапу IV;

2) якщо матриця A є непродуктивною або продуктивною, але її неможливо змінювати, то шляхом переходу до замкненої моделі за допомогою введення зворотного зв'язку

$$C_t = C_t^0 - kX_t,$$

де $k=(k_{ij})_{n\times n}$ – матриця сталих коефіцієнтів k_{ij} $(i,j=\overline{1,n})$ посилення зворотного зв'язку, за вибором яких є можливим зробити матрицю A продуктивною або покращити динамічні властивості систем.

В даному випадку за допомогою функцій C_t здійснюється регулювання в розімкненій дискретній моделі, що описується рівнянням

$$X_{t+1}=(I-B)^{-1}(A-B)X_t+(I-B)^{-1}C_t,\quad X(0)=X_0.$$

Відповідно до введеного зворотного зв'язку $C_t=C_t^0-kX_t$, побудована замкнена система керування опишеться наступним векторно-матричним рівнянням стану:

$$X_{t+1}=(I-B)^{-1}(A-k-B)X_t+(I-B)^{-1}C_t^0.$$

Вигляд замкненої моделі є аналогічним вигляду розімкненої моделі. Відмінність полягає в тому, що матриця A в розімкненій моделі замінена в замкненій моделі матрицею $(A-k)$.

У замкненій моделі C_t^0 задається як деяка функція часу t. У цьому випадку одержуємо систему лінійних різницевих рівнянь, розв'язок якої залежить від виду C_t^0 й надається у вигляді

$$X_t=\tilde{A}^{t-1}\tilde{B}((A-k-B)X_0+(I-A-k)X^*)+\tilde{A}^{t-1}\sum_{i=1}^{t-1}\tilde{A}^{-i}\tilde{B}C_i^0,$$

де

$$\tilde{A}=\tilde{B}(A-k-B),\quad \tilde{B}=(I-B)^{-1},\quad X^*=(I-(A-k))^{-1}C^0.$$

Оскільки асимптотична стійкість розв'язків X_t означає їх асимптотичну збіжність до стаціонарних розв'язків $X^* \geq 0$, визначимо основні співвідношення, що встановлюються між елементами вектора X^* й коефіцієнтами k_{ij} [21]:

– для збільшення значень елементів вектора X^*, а отже, значень елементів вектора X_t, достатнє виконання умов

$$\sum_{j=1}^{n} a_{ij} - 1 < \sum_{j=1}^{n} k_{ij} < \sum_{j=1}^{n} a_{ij}$$

або

$$\sum_{i=1}^{n} a_{ij} - 1 < \sum_{i=1}^{n} k_{ij} < \sum_{i=1}^{n} a_{ij},$$

причому у випадку продуктивної матриці A коефіцієнти k_{ij} можуть приймати як додатні, так і невід'ємні значення. У цьому випадку зворотний зв'язок може бути як невід'ємним (якщо k_{ij} обираються додатними), так і додатним (якщо k_{ij} обираються невід'ємними). У випадку непродуктивної матриці A коефіцієнти k_{ij} приймають додатні значення (зворотний зв'язок є невід'ємним);

– для зменшення значень елементів вектора X^*, а отже, значень елементів вектора X_t, достатнє виконання умов

$$\sum_{j=1}^{n} k_{ij} > \sum_{j=1}^{n} a_{ij}$$

або

$$\sum_{i=1}^{n} k_{ij} > \sum_{i=1}^{n} a_{ij},$$

причому, якщо матриця A є продуктивної, коефіцієнти $k_{ij} > 0$ (зворотний зв'язок є невід'ємним). Якщо матриця A непродуктивна, то $k_{ij} \geq 1$ (зворотний зв'язок є невід'ємним);

– для забезпечення рівності значень елементів вектора X^*, а отже, значень елементів вектора X_t, значенням елементів вектора C^0, достатньо, щоб

$$\sum_{j=1}^{n} k_{ij} = \sum_{j=1}^{n} a_{ij}$$

або

$$\sum_{i=1}^{n} k_{ij} = \sum_{i=1}^{n} a_{ij}.$$

б) при необхідності прискорення процесу збіжності вихідних характеристик X_t до рівноважного стану $X*$ об'єкта здійснюється керування за допомогою початкової матриці B.

При цьому, виходячи із продуктивності матриць B, $(A-B)$, встановлено, що для збільшення швидкості збіжності та покращення значень розв'язків X_t до рівноважного стану

$$X^* = (I - A)^{-1} C^0$$

елементи матриці B повинні задовольняти нерівності

$$0 \le b_{ij} \le a_{ij},\ i, j = \overline{1,n}\ (\text{при}\ n \ge 2)$$

або

$$0 \le B \le A\ (\text{при}\ n = 1).$$

З урахуванням вказаного, покращення вихідних характеристик об'єкта (обумовлене метою дослідження) шляхом безпосереднього вибору елементів матриці B визначається наступним:

– для збільшення значень кожного з елементів вектора X_t до відповідних «граничних» значень елементів вектора X^*, елементи матриці B повинні наближуватися до елементів матриці A (причому дана ситуація справедлива, якщо $C_i^0 \ne 0\ \left(i = \overline{1,n}\right)$);

– для зменшення значень кожного з елементів вектора X_t елементи матриці B повинні наближуватися до нуля.

Даний підхід може бути розповсюджений і на замкнену дискретну модель позитивної системи шляхом заміни матриці A (у відповідній розімкненій моделі) матрицею $(A - k)$, де $k -$ матриця коефіцієнтів посилення зворотного зв'язку, елементи якої визначаються відповідно до підходу, описаних на етапі IV, підпункті б) пункту 1).

V eman: Отримання в табличній формі та графічне представлення одержаних результатів. В даному випадку і в табличному, і в графічному виглядах надається інформація про статичний розв’язок X^* та отриманий за моделлю розв’язок X_t, впорядковані в хронологічному порядку. Завдяки отримуваній за таким виглядом отримуваних результатів візуалізації, здійснення аналізу прогнозованих за моделлю даних проводиться в більш зручний спосіб.

Зауважимо, що на основі пропонованої методики проведення аналізу математичної моделі та основних властивостей складної керованої позитивної динамічної системи й отримуваних за нею результатів в наступному розділі роботи проводиться розробка програмного забезпечення, яке дозволяє автоматизувати процес виконання широкого спектру трудомістких обчислень, проведення аналізу основних характеристик моделі та властивостей складної системи.

3 АВТОМАТИЗАЦІЯ ПРОЦЕСУ МОДЕЛЮВАННЯ ТА АНАЛІЗУ ОСНОВНИХ ВЛАСТИВОСТЕЙ СКЛАДНОЇ КЕРОВАНОЇ ПОЗИТИВНОЇ ДИНАМІЧНОЇ СИСТЕМИ

В даному розділі наводиться огляд існуючих програмних засобів та інструментів для автоматизації математичного моделювання динамічних систем та представляється характеристика розроблюваного в роботі програмного забезпечення для автоматизації процесу математичного моделювання та аналізу основних властивостей складної керованої позитивної динамічної системи. Крім того, з метою апробації пропонованого програмного забезпечення в даному розділі наводяться практичні приклади проведених обчислювальних експериментів.

3.1 Огляд існуючих програмних засобів та інструментів для автоматизації процесу математичного моделювання динамічних систем

Огляд існуючих програмних засобів та інструментів для автоматизації математичного моделювання динамічних систем є важливим кроком у дослідженні. На сьогоднішній день існує широкий вибір програмних засобів та інструментів [12, 14, 15, 17, 18, 23, 24], що допомагають в розробці, візуалізації та аналізі математичних моделей динамічних систем.

Одним із найпопулярніших програмних пакетів для математичного моделювання виступає MATLAB [14, 17, 18], що забезпечує потужні інструменти для здійснення числових розрахунків, визначення та відображення функцій, графіків та діаграм, а також має велику базу готових функціональних блоків та модулів для моделювання різних типів систем.

Ще одним популярним інструментом є Simulink [17, 23], що входить до складу пакету MATLAB. Simulink дозволяє розробляти блок-схеми для

моделювання та симуляції динамічних систем. Він має велику бібліотеку готових блоків, що спрощує процес побудови складних моделей.

Крім MATLAB та Simulink, існують інші популярні програмні засоби, такі як Python з бібліотеками NumPy та SciPy, R, Octave та деякі комерційні пакети, спеціалізовані на моделюванні конкретних типів систем [13, 35-37].

Вибір певного програмного засобу залежить від специфіки дослідження, області застосування та особистих вподобань дослідника. Важливо враховувати можливості програмного забезпечення, його функціональність та зручність використання при розв'язуванні конкретних задач математичного моделювання динамічних систем.

Оскільки для позитивної системи висуваються специфічні обмеження [19-22], виконання яких досить важко зберегти із використанням стандартних методів у існуючому програмному забезпеченні, виникла потреба в розробці нового програмного забезпечення за представленою у попередньому розділі роботи методикою проведення аналізу математичної моделі та основних властивостей складної керованої позитивної динамічної системи й отримуваних за нею результатів.

3.2 Основні характеристики розробленого програмного забезпечення для автоматизації процесу моделювання та аналізу основних властивостей складної керованої позитивної динамічної системи

На основі пропонованої в попередньому розділі методики дослідження в роботі проводиться розробка програмного забезпечення (програмної системи), яке дозволяє автоматизувати процес виконання широкого спектру трудомістких обчислень, проведення аналізу основних характеристик моделі та властивостей досліджуваної складної керованої системи. Наведемо стислу характеристику розробленого програмного забезпечення для автоматизації

процесу моделювання та аналізу основних властивостей складної керованої позитивної динамічної системи.

Розроблена програмна система написана з використанням мови програмування Python [13, 35], в середовищі розробки IDLE для Python версії 3.10.11.

Вибір Python для реалізації програми має кілька переваг. Сформулюємо їх та наведемо стислу характеристику кожної.

По-перше, Python є високорівневою мовою програмування, що робить його дуже зручним і легким у використанні. Крім того, Python має простий і зрозумілий синтаксис, що сприяє швидкому розробленню програм та полегшує супровід і розширення коду.

По-друге, Python має велику кількість стандартних бібліотек і сторонніх модулів, що дозволяє ефективно виконувати різноманітні завдання. Наявність готових інструментів спрощує реалізацію складних алгоритмів та обробку даних.

Крім того, Python є платформонезалежною мовою, що означає, що програми, написані на Python, можуть запускатись на різних операційних системах без змін. Це забезпечує переносимість та зручність використання програмної системи на різних пристроях та платформах.

У результаті вибір Python як мови програмування для реалізації розроблюваної в роботі програмної системи дозволив забезпечити зручність розробки, швидкість виконання та переносимість, що є важливими аспектами для успішної автоматизації дослідження складних керованих динамічних систем.

Розроблювана програмна система включає в себе використання деяких додаткових бібліотек, зокрема NumPy і matplotlib.pyplot. Кожна з цих бібліотек дозволяє розширити можливості Python та забезпечити виконання поставлених в роботі завдань дослідження.

Так, бібліотека NumPy використовується для обробки та маніпуляцій з числовими даними у Python. Вона надає широкий набір функцій та методів

для роботи з масивами чисел, включаючи математичні операції, лінійну алгебру, статистику та інші обчислювальні операції. NumPy дозволяє зручно та ефективно виконувати розрахунки, що є необхідним у процесі обробки та аналізу даних динамічних систем.

В розроблюваній програмній системі з бібліотеки NumPy були використані такі функції як [13, 36, 37]:

а) np.array – функція бібліотеки NumPy у Python, яка використовується для створення та роботи з масивами чисел, приймає послідовність чисел або вкладені списки й перетворює їх у масив NumPy, а також дозволяє здійснювати різноманітні операції над масивами чисел, такі як арифметичні операції (додавання, віднімання, множення, ділення), доступ до елементів масиву за допомогою індексів, зміна розмірності масиву, виконання математичних функцій тощо. Одна з головних переваг np.array полягає у тому, що вона дозволяє виконувати ефективні операції на великих масивах чисел, що робить її корисною для обробки даних, наукових обчислень та моделювання. Наприклад, за допомогою np.array можна створити масив чисел, виконати математичні операції над ними та отримати результат;

б) np.zeros – функція бібліотеки NumPy у Python, яка використовується для створення масиву з нулів, приймає в якості аргументу кортеж або список, який визначає розміри масиву, і повертає масив з усіма елементами, що дорівнюють нулю. Цей метод часто використовується для ініціалізації масиву перед заповненням його даними або обчисленнями. Він дозволяє швидко створювати масиви з великою кількістю елементів, що мають однакове значення;

в) np.dot – функція бібліотеки NumPy в Python, яка використовується для обчислення скалярного (внутрішнього) добутку двох масивів або матриць. Цей метод виконує матричне множення для двох масивів або матриць різних розмірів. Якщо вхідні аргументи є одномірними масивами, то np.dot виконує звичайний скалярний добуток векторів. У випадку, коли вхідні аргументи є двовимірними матрицями, np.dot виконує матричне множення;

г) np.add – функція бібліотеки NumPy в Python, яка використовується для елементарного поелементного додавання двох масивів або матриць. Цей метод виконує поелементне додавання для відповідних елементів двох масивів або матриць однакового розміру. Якщо вхідні аргументи є одномірними масивами, то np.add виконує додавання елементів векторів поелементно. У випадку, коли вхідні аргументи є двовимірними матрицями, np.add виконує поелементне додавання елементів матриць;

д) np.linalg.inv – функція бібліотеки NumPy в Python, яка використовується для обернення квадратної матриці. Цей метод дозволяє обчислити обернену матрицю для заданої квадратної матриці. Обернена матриця має властивість, що добуток поелементних множин квадратної матриці на її обернену матрицю дає одиничну матрицю;

е) np.identity – функція бібліотеки NumPy в Python, яка використовується для створення квадратної одиничної матриці (одиничної матриці або матриці одиничного оператора), під якою розуміється квадратна матриця, в якій всі елементи на головній діагоналі рівні одиниці, а всі інші елементи рівні нулю. Функція np.identity(n) створює одиничну матрицю розмірністю n × n, при цьому кожен елемент головної діагоналі матриці буде дорівнювати 1, а всі інші елементи – нулю;

ж) np.eye – функція бібліотеки NumPy в Python, яка використовується для створення квадратної одиничної матриці або матриці з деякими ненульовими елементами на головній діагоналі. Функція np.eye(N, M=None, k=0) створює матрицю розміром N × M, де всі елементи на головній діагоналі дорівнюють одиниці, а всі інші елементи – нульові. Аргумент N визначає кількість рядків у матриці, аргумент M (необов'язковий) – кількість стовпців. Аргумент k (необов'язковий) визначає зсув головної діагоналі, де k=0 відповідає головній діагоналі, k>0 – діагоналі вище головної, а k<0 – діагоналі нижче головної;

з) LA.eigvals – функція з пакету numpy.linalg в Python для обчислення власних значень квадратної матриці. Синтаксис функції LA.eigvals(A)

приймає один аргумент A, який представляє квадратну матрицю. Функція LA.eigvals повертає масив, що містить власні значення матриці *A*.

Надалі, надамо стислу характеристику використовуваної при написанні програмного коду бібліотеки matplotlib.pyplot. Перш за все, потрібно зауважити, що при програмуванні вона зазвичай використовується для візуалізації даних у вигляді графіків та діаграм. Саме завдяки використанню функцій бібліотеки matplotlib.pyplot можна будувати різноманітні типи графіків, включаючи лінійні графіки, стовпчикові діаграми, точкові графіки та багато інших. Таким чином, використання зазначеної бібліотеки надає розширені можливості для відображення результатів аналізу динамічних систем у зрозумілій та графічній формі, які і в проміжному дослідженні в даній роботі мають велике значення.

В розроблюваній програмній системі з бібліотеки matplotlib.pyplot були використані такі функції як [13, 35-37]:

а) plt.grid – функція в Python, яка використовується для відображення сітки на графіку, створеному з використанням бібліотеки matplotlib.pyplot. Синтаксис функції plt.grid() досить простий. Ця функція не приймає жодних аргументів. Виклик функції plt.grid() додає сітку до поточного графіку;

б) plt.xlim – функція в Python, яка використовується для встановлення меж діапазону осі x на графіку, створеному з використанням бібліотеки matplotlib.pyplot. Синтаксис функції plt.xlim() дозволяє вказати значення початку та кінця діапазону осі x. Функція приймає два аргументи: початкове значення та кінцеве значення діапазону осі x;

в) plt.ylim – функція в Python, яка використовується для встановлення меж діапазону осі y на графіку, створеному з використанням бібліотеки matplotlib.pyplot. Синтаксис функції plt.ylim() дозволяє вказати значення початку та кінця діапазону осі y. Функція приймає два аргументи: початкове значення та кінцеве значення діапазону осі y;

г) plt.xticks – функція в Python, яка використовується для налаштування відміток (позначок) на осі x графіку, створеного з використанням бібліотеки

matplotlib.pyplot. Синтаксис функції plt.xticks() дозволяє вказати позиції відміток на осі x та їх відповідні підписи. Функція приймає два аргументи: список позицій відміток та список відповідних підписів;

д) plt.scatter – функція в Python, яка використовується для створення графіку розсіювання (scatter plot) з використанням бібліотеки matplotlib.pyplot. Функція plt.scatter приймає два аргументи, які відповідають координатам точок на графіку розсіювання. Зазвичай ці аргументи представлені списками або масивами числових значень. Крім того, для даної функції можна вказати додаткові аргументи, такі як кольори, розміри та маркери точок;

е) plt.axhline – функція в Python, яка використовується для додавання горизонтальної прямої лінії на графік з використанням бібліотеки matplotlib.pyplot. Функція plt.axhline приймає один аргумент, який відповідає значенню y-координати горизонтальної прямої лінії. Також можна вказати додаткові аргументи, такі як кольори, стиль лінії та ширина, що також розширює можливості аналізу досліджуваних систем;

ж) plt.legend – функція в Python, яка використовується для додавання легенди (пояснювального тексту) на графік з використанням бібліотеки matplotlib.pyplot. Функція plt.legend дозволяє вказати мітки для різних елементів графіку, що допомагає ідентифікувати їх на графіці. Легенда зазвичай розташовується в куті графіка або поза ним;

з) plt.show – функція в Python, яка використовується для відображення графіка або фігури, створеної з використанням бібліотеки matplotlib.pyplot. Після побудови графіка або налаштування фігури за допомогою різних функцій pyplot, виклик функції plt.show дозволяє відобразити цю графіку або фігуру на екрані.

Підводячи підсумок наведеній характеристиці використовуваних в розроблюваному програмному забезпеченні бібліотек Python, можна сказати, що їх використання є досить виправданим, оскільки надає широкий набір функцій та методів для роботи з великими масивами чисел, дозволяє провести

візуалізацію проміжних та кінцевих результатів моделювання й аналізу, а, отже, значно розширює можливості аналізу досліджуваних систем.

3.3 Опис розробленої програмної системи та її можливостей для автоматизації дослідження складних керованих динамічних систем

Розроблена програмна система, спрямована на автоматизацію дослідження складних керованих динамічних систем, складається з двох програмних компонентів (лістинги програмного коду для кожного з компонентів наведено відповідно у Додатках А, Б кваліфікаційної роботи). Наведемо стислу характеристику зазначених програмних компонентів.

Перша програма виконує перевірку продуктивності систем шляхом проведення різних тестів та аналізу їх результатів. Вона дозволяє оцінити ефективність розглянутої системи та здійснити попередню фільтрацію систем, які не відповідають заданим критеріям продуктивності, та / або перейти до розв’язання перерахованих у роботі задач керування для досягнення поставлених до системи вимог.

Друга програма використовує складену формулу для обчислення значень X_t відповідно до вхідних параметрів. Вона забезпечує проведення числових обчислень та розрахунок значень X_t для обчислювальних експериментів за досліджуваними динамічними системами. Отримані остаточні результати за роботою даної програми виводяться на екран у вигляді графіка, що дозволяє зробити візуальну оцінку поведінки системи та аналізувати її динаміку.

Загалом, розроблена в роботі програмна система надає дослідникам можливість автоматизувати процес дослідження складних керованих динамічних систем, спрощує аналіз та обробку даних, а також дозволяє швидше отримати результати дослідження. Завдяки окресленим можливостям, дослідники можуть зосередитися на аналізі отриманих результатів та

виявленні закономірностей в поведінці досліджуваної системи, що в свою чергу сприяє й подальшому розвитку та удосконаленню складних керованих динамічних систем.

3.4 Основні кроки з використання розробленої програмної системи

Для ефективного використання розроблених програм рекомендується дотримуватися певних кроків.

У першій програмі необхідно спочатку задати послідовно вихідні матриці A та B досліджуваної системи та її розмірності. Після цього програма перевірить, чи виконується умова продуктивності для всіх матриць досліджуваної системи. Якщо всі умови продуктивності виконуються, можна переходити до другої програми та здійснювати відповідні розрахунки для знаходження модельованих значень вектор-функції X_t та, за необхідності, покращувати динамічні властивості системи. В противному випадку потрібно використати відповідне керування, реалізоване у частині другої програми, та привести матриці системи до продуктивних.

Розглянемо описаний процес роботи першої програми на конкретному прикладі, при проведенні обчислювального експерименту.

Після відкриття вікна введення даних (рис. 3.1), необхідно спочатку задати розмірності й ввести поелементні значення вихідних матриць A та B досліджуваної системи (рис. 3.2). Також на даному кроці перевіряється, чи виконується умова невід’ємності та продуктивності матриць моделі системи, та задається кількість виконуваних ітерацій для знаходження суми ряду $\sum_{k=0}^{\infty} S^k$ для відповідної матриці системи S (рекомендовано проведення 40 таких ітерацій) (найнижчий рядок у вікні програми на рис. 3.2).

```
Введите розмірність матриці A:
```

Рисунок 3.1 – Скріншот відкриття програми

```
Введите розмірність матриці A:
2
введіть (1,1) елемент
0.6
введіть (1,2) елемент
0.2
введіть (2,1) елемент
0.3
введіть (2,2) елемент
0.4
Введена матрица A:
 [[0.6 0.2]
 [0.3 0.4]]
інвертирована матриця(E-A)^(-1):

 [[3.33333333 1.11111111]
 [1.66666667 2.22222222]]

(E-A)^(-1) >= 0

Матриця A є невід'ємною

Всі диагональні елементи матриці A меньше одиниці

Введіть кількість ітерацій для ряду
```

Рисунок 3.2 – Скріншот вікна введення вихідної матриці *A* системи та перевірки її продуктивності

Випадки, за якими умова продуктивності матриці не виконується, представлено відповідно на рис. 3.3, 3.4.

```
Введена матрица A:
 [[1.  0.2]
 [0.3 0.4]]
Введите розмірність матриці B:
2
введіть (1,1) елемент
1
введіть (1,2) елемент
0.2
введіть (2,1) елемент
0.25
введіть (2,2) елемент
0.3
Введена матрица B:
 [[1.   0.2 ]
 [0.25 0.3 ]]
(I-B)^-1*(A-B)=
 [[-0.2 -0.4]
 [ 0.   0. ]]

Введіть кількість ітерацій для ряду
40
Розрахуємо продуктивність та невід'ємність для матриці A
Всі елементи >= 0

інвертирована матриця(E-матриця)^(-1):

 [[-10.          -3.33333333]
 [ -5.          -0.        ]]

(E-Матриця)^(-1) < 0

Матриця є невід'ємною
Не всі диагональні елементи матриці меньше одиниці
Отже,матриця не є позитивною
```

Рисунок 3.3 – Скріншот роботи програми у першому випадку незадовільних умов функціонування позитивної системи

```
Введите розмірність матриці A:
2
введіть (1,1) елемент
1
введіть (1,2) елемент
2
введіть (2,1) елемент
3
введіть (2,2) елемент
4
Введена матрица A:
 [[1. 2.]
 [3. 4.]]
інвертирована матриця(E-A)^(-1):

 [[ 0.5        -0.33333333]
 [-0.5        -0.        ]]

(E-A)^(-1) < 0

Матриця A є невід'ємною

Не всі диагональні елементи матриці A меньше одиниці
```

Рисунок 3.4 – Скріншот роботи програми у другому випадку незадовільних умов функціонування позитивної системи

Після введення всіх матриць системи й перевірки умов її функціонування здійснюється перехід до другої програми розробленої програмної системи. У цій програмі після її відкриття необхідно спочатку задати кількість прогнозованих кроків *t* (рис. 3.5).

```
Введіть t
1
```

Рисунок 3.5 – Введення кількості прогнозованих кроків *t*

Надалі потрібно ввести матрицю *A*, яка буде використовуватися для обчислень, вектор вільних членів X_0, початкові значення коефіцієнтів C_i^0 та C_i $(i=\overline{1,n})$ вектор-функції C_t.

Розглянемо цей процес на конкретному числовому прикладі. Перш за все у програмі потрібно ввести розмірність та коефіцієнти C_i^0 та C_i $(i=\overline{1,n})$ вектор-функції C_t (рис. 3.6).

```
Введіть розмірність вектора C:
2
введіть C0[0]:
0.1
введіть шаг C_i[0]
0.0002
введіть шаг C_i[0]
введіть C0[1]:
0.6
введіть шаг C_i[1]
0.00015
```

Рисунок 3.6 – Скріншот введення розмірності та коефіцієнтів C_i^0 та C_i $(i=\overline{1,n})$ вектор-функції C_t

В результаті отримаємо на екрані всі значення впорядкованої за часом t вектор-функції C_t (у програмі вона позначається через C), причому спочатку у квадратних дужках (до коми) розташовані значення вектор-функції C_t для першої підсистеми, після цього у наступних квадратних дужках виводяться значення для другої підсистеми (рис. 3.7). В даному випадку представлено значення вектор-функції C_t при її завданні у лінійному вигляді.

```
C=
 [[0.1, 0.1002, 0.1004, 0.1006, 0.1008, 0.101, 0.1012, 0.1014, 0.1016, 0.1018],
[0.6, 0.60015, 0.6003, 0.60045, 0.6006, 0.60075, 0.6009, 0.60105, 0.6012, 0.6013
5]]
```

Рисунок 3.7 – Скріншот виведення значень впорядкованої за часом t вектор-функції C_t

Далі вводимо значення вхідних матриць A й B системи та визначаємо їх розмірності (рис. 3.8), причому програма спочатку виводить на екран вигляд потрібної матриці для наочності та вірного розташування її елементів, а далі

користувачеві надається можливість послідовного введення значення кожного з елементів матриці (спочатку запонюються елементи першого рядка – на рис. 3.8 це елементи (1,1) та (1,2) відповідно, потім – другого рядка – на рис. 3.8 це елементи (2,1) та (2,2) відповідно, і т.д.). Після введення елементів матриці програма виводить на екран і остаточний її вигляд.

```
Введіть розмірність матриці A:
2
Ваша матриця:
(
0.0,0.0
0.0,0.0
)
введіть (1,1) елемент
0.6
введіть (1,2) елемент
0.2
введіть (2,1) елемент
0.3
введіть (2,2) елемент
0.4
Введена матриця A:
 [[0.6 0.2]
 [0.3 0.4]]
Введіть розмірність матриці B:
2
(
0.0,0.0
0.0,0.0
)
введіть (1,1) елемент
0.35
введіть (1,2) елемент
0.1
введіть (2,1) елемент
0.2
введіть (2,2) елемент
0.25
Введена матриця B:
 [[0.35 0.1 ]
 [0.2  0.25]]
```

Рисунок 3.8 – Скріншот вікна введення розмірностей та значень елементів вихідних матриць *A* та *B*

Далі вводимо розмірність та по елементні значення вектору X_0 (рис. 3.9).

```
ведіть розмірність X0
2
Введите 1 элемент x0
8
Введите 2 элемент x0
5
Ви ввели:
 [[8.]
 [5.]]
```

Рисунок 3.9 – Скріншот введення розмірності та початкових значень елементів вектору стану X_0 системи

В результаті чого отримуємо шукані значення вектору результатів X_t системи (рис. 3.10).

```
X1=
 [[4.63101604]
 [4.10160428]]
X2=
 [[3.03515519]
 [3.04736438]]
X3=
 [[2.15836566]
 [2.39012441]]
X4=
 [[1.6617827 ]
 [2.00954902]]
```

Рисунок 3.10 – Скріншот виведення результатів X_t

У даній програмі при цьому слід звернути увагу на функцію func_c, яка описує залежність системи. За необхідності, залежно від умови, функцію можна змінити з лінійної на квадратичну або гармонічну, враховуючи вимоги досліджуваної системи. Це можна зробити шляхом внесення відповідних змін у функцію func_c (рис. 3.11).

```
def func_c(C_given,c,t):
    #omega=80          тут треба вводити омегу,при гармонічн.функ. C
    C=[0]*c
    for i in range(c):
        C[i]=[0]*t

    for i in range(c):

        for j in range(t):
            C[i][j]=round((C_given[i][0]+(C_given[i][1]*j)),5)          #round
            #тут замінити функцію C на
            #квадратичну:
            #C[i][j]=round((C_given[i][0]+(C_given[i][1]*j*j)),5)

            #гармоничну:
            ##C[i][j]=round((C_given[i][0]+(C_given[i][1]*sin(omega*j))),5)

    return(C)
```

Рисунок 3.11 – Функція func_c

Також у цій програмі перевіряється, чи потрібно здійснити керування у системі для покращення її динамічних властивостей (рис. 3.12).

```
Чи необхідно покращення вхідних параметрів?так/ні.
так
збільшемо значення матриці A
Введена матриця A:
 [[0.8  0.6 ]
 [0.65 0.7 ]]
```

Рисунок 3.12 – Скріншот введення даних з потрібного користувачу керування системою

Попередньо у програмі також перевіряється виконання властивості повної керованості системи. Приклад цієї перевірки надається у наступному п. 3.5 роботи. Надалі у програмі здійснюються окремі види здійснюваного керування та для кожного випадку отримуються траєкторії X_t системи, представлені в табличному й графічному виглядах. За отриманими результатами роботи програми надаються рекомендації для здійснення ефективного функціонування досліджуваної позитивної системи.

3.5 Приклад практичного використання програмного забезпечення. Проведення обчислювального експерименту на моделі

Використання програмних засобів та інструментів є важливим етапом в дослідженні складних керованих динамічних систем. Написання та використання програм дозволяє не тільки автоматизувати процес моделювання, але й здійснювати широкий аналіз системи, вивчати різні варіанти вхідних даних та оцінювати їх вплив на поведінку системи.

У даному пункті роботи представлено результати проведення обчислювального експерименту на розроблених двох програмах програмної системи, які дозволяють здійснювати розрахунки та проводити аналіз складних динамічних систем. Обчислювальний експеримент в роботі проводився для позитивної динамічної системи із двома підсистемами ($n = 2$) при наступних початкових умовах та вхідних даних моделі:

$$A = \begin{pmatrix} 0{,}6 & 0{,}2 \\ 0{,}3 & 0{,}4 \end{pmatrix};\ B = \begin{pmatrix} 0{,}35 & 0{,}1 \\ 0{,}2 & 0{,}25 \end{pmatrix};\ X_0 = \begin{bmatrix} 8 \\ 5 \end{bmatrix};$$

$$C^0 = \begin{pmatrix} 0{,}1 \\ 0{,}6 \end{pmatrix};\ C = \begin{pmatrix} 0{,}0002 \\ 0{,}00015 \end{pmatrix};\ \omega = 45,\ t \in [0{,}1,\ldots,10].$$

Розглянемо роботу першої програми, призначеної для перевірки продуктивності матриць системи. За допомогою неї можна задати довільну матрицю та її розмірність, а потім виконати перевірку на відповідність критеріям продуктивності. Дана програма отримує від користувача матриці A, B (рис. 3.13), обчислює матриці $(A - B)$, $\tilde{A} = (I - B)^{-1}(A - B)$ та визначає, чи є всі отримані матриці позитивними, невід'ємними, чи їх елементи менше за одиницю, та видає найбільші за модулем значення характеристичного рівняння (рис. 3.14 та рис. 3.15).

```
Введите розмірність матриці A:
2
введіть (1,1) елемент
0.6
введіть (1,2) елемент
0.2
введіть (2,1) елемент
0.3
введіть (2,2) елемент
0.4
Введена матрица A:
 [[0.6 0.2]
 [0.3 0.4]]
Введите розмірність матриці B:
2
введіть (1,1) елемент
0.35
введіть (1,2) елемент
0.1
введіть (2,1) елемент
0.2
введіть (2,2) елемент
0.25
Введена матрица B:
 [[0.35 0.1 ]
 [0.2  0.25]]
(I-B)^-1*(A-B)=
 [[0.42245989 0.19251337]
 [0.2459893  0.2513369 ]]

Введіть кількість ітерацій для ряду
40
Розрахуємо продуктивність та невід'ємність для матриці A
Всі елементи >= 0
```

Рисунок 3.13 – Скріншот роботи програми 1 – введення вихідних матриць A, B та визначення матриці $\tilde{A}$ системи

```
Розрахуємо продуктивність та невід'ємність для матриці A
Всі елементи >= 0

інвертирована матриця(E-матриця)^(-1):

 [[3.33333333 1.11111111]
 [1.66666667 2.22222222]]

(E-Матриця)^(-1) >= 0

Матриця є невід'ємною
Всі диагональні елементи матриці меньше одиниці
матричний ряд збігається до:
 [[3.33328474 1.11108445]
 [1.66662668 2.22220029]]
Найбільше за модулем власне занчення хар.рівняння:
 [0.76457513 0.23542487]

Розрахуємо продуктивність та невід'ємність для матриці B
Всі елементи >= 0

інвертирована матриця(E-матриця)^(-1):

 [[1.60427807 0.21390374]
 [0.42780749 1.39037433]]

(E-Матриця)^(-1) >= 0

Матриця є невід'ємною
Всі диагональні елементи матриці меньше одиниці
матричний ряд збігається до:
 [[1.60427807 0.21390374]
 [0.42780749 1.39037433]]
Найбільше за модулем власне занчення хар.рівняння:
 [0.45 0.15]
```

Рисунок 3.14 —Скріншот роботи програми 1 – перевірка продуктивності вихідних матриць *A*, *B* системи

```
Розрахуємо продуктивність та невід'ємність для матриці A-B
Всі елементи >= 0

інвертирована матриця(E-матриця)^(-1):

 [[1.35458167 0.15936255]
 [0.15936255 1.19521912]]

(E-Матриця)^(-1) >= 0

Матриця є невід'ємною
Всі диагональні елементи матриці меньше одиниці
матричний ряд збігається до:
 [[1.35458167 0.15936255]
 [0.15936255 1.19521912]]
Найбільше за модулем власне занчення хар.рівняння:
 [0.3118034 0.0881966]

Розрахуємо продуктивність та невід'ємність для матриці (I-B)^-1*(A-B)
Всі елементи >= 0

інвертирована матриця(E-матриця)^(-1):

 [[1.94444444 0.5       ]
 [0.63888889 1.5       ]]

(E-Матриця)^(-1) >= 0

Матриця є невід'ємною
Всі диагональні елементи матриці меньше одиниці
матричний ряд збігається до:
 [[1.94444444 0.5       ]
 [0.63888889 1.5       ]]
Найбільше за модулем власне занчення хар.рівняння:
 [0.57072953 0.10306726]

Якщо всі умови виконуються, можете переходити до програми 2.
```

Рисунок 3.15 —Скріншот роботи програми 1 – перевірка продуктивності розрахованих матриць $(A-B)$ та $\tilde{A}=(I-B)^{-1}(A-B)$

Надалі здійснюється перехід до другої програми, яка для здійснення своєї роботи запитує у користувача кількість прогнозних моментів часу *t*, розмірність, вид завдання функції C_t та значення її коефіцієнтів, матрицю A та B (вони можуть змінюватись при здійсненні замкненого керування у

системі) та значення елементів вектору вільних членів X_0 (рис. 3.16) та (рис. 3.17).

```
Введіть t
10
Введіть розмірність вектора C:
2
введіть C0(1):
0.1
введіть коефіцієнти C_i(1)
0.0002
введіть коефіцієнти C_i(1)
введіть C0(2):
0.6
введіть коефіцієнти C_i(2)
0.00015
введіть коефіцієнти C_i(2)
виберіть вид функції C(1-лінійний,2-квадратичний,3-гармонічний
1
C=
 [[0.1, 0.1002, 0.1004, 0.1006, 0.1008, 0.101, 0.1012, 0.1014, 0.1016, 0.1018],
[0.6, 0.60015, 0.6003, 0.60045, 0.6006, 0.60075, 0.6009, 0.60105, 0.6012, 0.6013
5]]
Введіть розмірність матриці A:
2
Ваша матриця:
(
0.0,0.0
0.0,0.0
)
введіть (1,1) елемент
0.6
введіть (1,2) елемент
0.2
введіть (2,1) елемент
0.3
введіть (2,2) елемент
0.4
Введена матриця A:
```

Рисунок 3.16 – Скріншот роботи програми 2 – введення початкових умов та вхідних для програми даних

Після введення початкових умов та вхідних для програми даних, програма перевіряє володіння досліджуваною системою властивості повної керованості. Для цього на екрані вона видає матрицю керованості, а також знаходить її ранг. Якщо ранг співпадає з розмірністю матриць *A* або *B*, то система вважається повністю керованою та доступною для здійснення обраного користувачем певного виду керування (рис. 3.18).

```
Введена матриця A:
 [[0.6 0.2]
 [0.3 0.4]]
Введіть розмірність матриці B:
2
(
0.0,0.0
0.0,0.0
)
введіть (1,1) елемент
0.35
введіть (1,2) елемент
0.1
введіть (2,1) елемент
0.2
введіть (2,2) елемент
0.25
Введена матриця B:
 [[0.35 0.1 ]
 [0.2  0.25]]
ведіть розмірність X0
2
Введите 1 элемент x0
8
Введите 2 элемент x0
5
Ви ввели:
 [[8.]
 [5.]]
```

Рисунок 3.17 – Скріншот роботи програми 2 – продовження введення початкових умов та вхідних для програми даних

```
матриця керованості:
 [[1.60427807 0.21390374 0.7601018  0.3580314 ]
 [0.42780749 1.39037433 0.50215906 0.40207041]]
ранг матриці дорівнює 2
```

Рисунок 3.18 – Скріншот роботи програми 2 – перевірка володіння досліджуваною системою властивості повної керованості

Далі програма виводить на екран значення векторів X_t в табличному (рис. 3.19) та графічному виглядах (рис. 3.20).

```
t  | X(t)1          | X(t)2          |
1  | [4.63101604]   | [4.10160428]   |
2  | [3.03515519]   | [3.04736438]   |
3  | [2.15836566]   | [2.39012441]   |
4  | [1.6617827]    | [2.00954902]   |
5  | [1.37908341]   | [1.79203641]   |
6  | [1.21813316]   | [1.66812058]   |
7  | [1.12663562]   | [1.59767803]   |
8  | [1.07477338]   | [1.55775993]   |
9  | [1.04553184]   | [1.53526359]   |
10 | [1.02920056]   | [1.52271045]   |
```

Рисунок 3.19 – Скріншот виведення результативних значень X_t в табличному вигляді

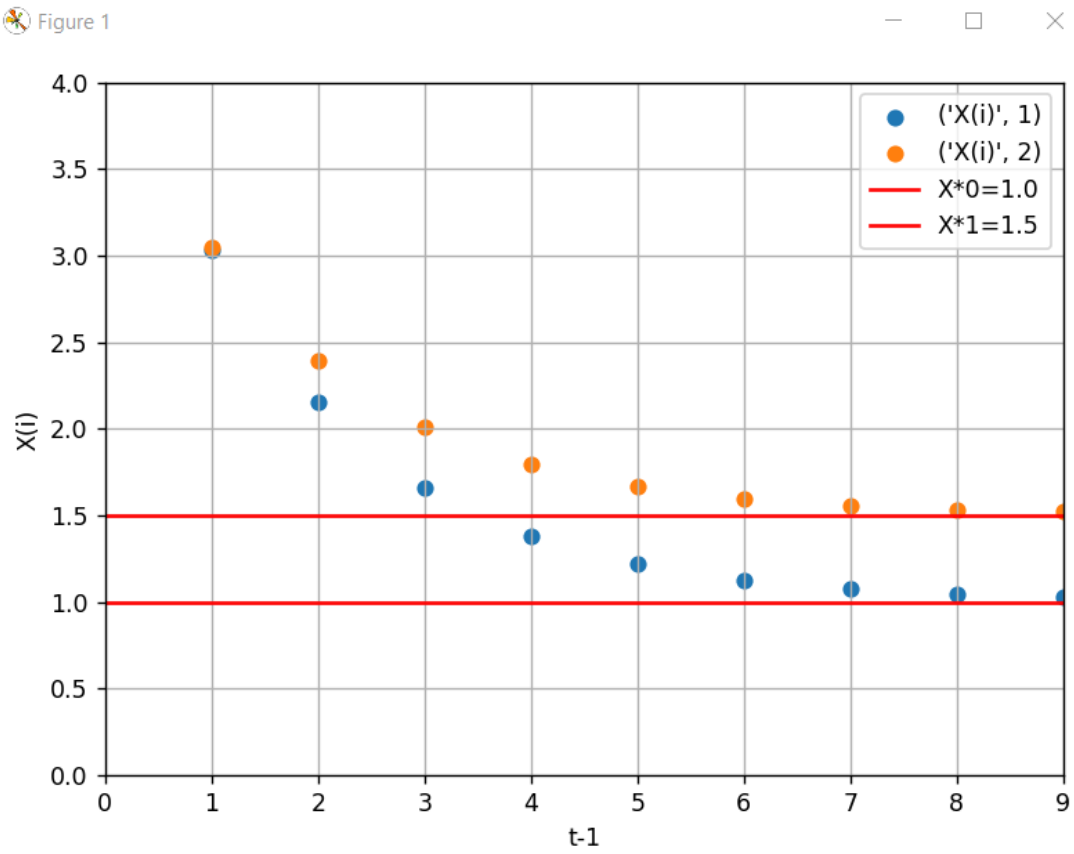

Рисунок 3.20 – Скріншот виведення результативних значень X_t в графічному вигляді

Далі, при необхідності покращення вхідних параметрів і вихідних характеристик об’єкта дослідження, проводиться керування за допомогою початкової матриці A (рис. 3.21).

```
Чи необхідно покращення вхідних параметрів?так/ні.
так
матриця А є продуктивною та можна її змінювати?так/ні
так
збільшемо чи зменьшимо значення матриці А?

збільшемо-1
зменьшимо-2
1
Введена матриця А:
 [[0.625   0.25   ]
 [0.34375 0.4375 ]]
```

Рисунок 3.21 – Скріншот здійснення покращення вхідних параметрів

І далі, аналогічно, отримуємо результати вже для збільшеної матриці A в табличному (рис. 3.22) та графічному виглядах (рис. 3.23).

```
Введена матриця А:
 [[0.625   0.25   ]
 [0.34375 0.4375 ]]
inv*dif:
 [[0.47192513 0.28074866]
 [0.31751337 0.32486631]]
матриця керованості:
 [[1.60427807 0.21390374 0.87720552 0.49129229]
 [0.42780749 1.39037433 0.64835998 0.51960308]]
ранг матриці дорівнює 2
отже,система є повністю керованою
inv*dif:
 [[0.47192513 0.28074866]
 [0.31751337 0.32486631]]
 t | X(t)1         | X(t)2         |
 1 | [5.46791444] | [5.04144385] |
 2 | [4.28494787] | [4.25123066] |
 3 | [3.50517786] | [3.61920344] |
 4 | [2.96009693] | [3.1665858] |
 5 | [2.57614069] | [2.84676921] |
 6 | [2.30550695] | [2.62125445] |
 7 | [2.11482806] | [2.46235659] |
 8 | [1.98058448] | [2.35048705] |
 9 | [1.88617728] | [2.2718144] |
 10| [1.81988984] | [2.21657487] |
```

Рисунок 3.22 – Скріншот виведення результативних значень X_t в табличному вигляді для покращеної матриці A

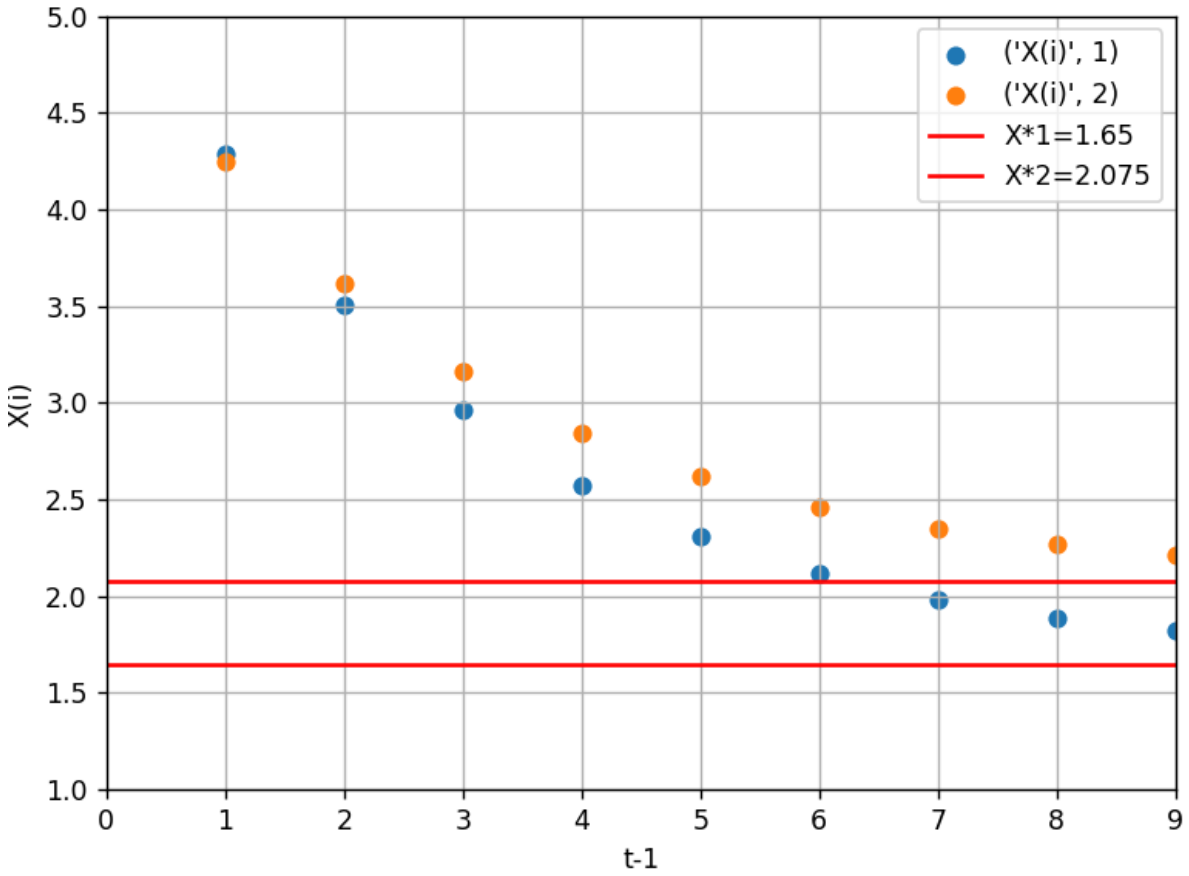

Рисунок 3.23 —Скріншот виведення результативних значень X_t в графічному вигляді для покращеної матриці A

За необхідності здійснення додаткового керування певного виду, зазначеного у розділі 2 кваліфікаційної роботи, програма продовжує свою роботу аналогічно останньому кроку.

Отримані в результаті використання розробленої програмної системи результати проведених обчислювальних експериментів вказують на ефективність та придатність розробленої методики та програмного забезпечення для подальших досліджень та застосування в практичних задачах керування складними динамічними системами.

ВИСНОВКИ

Кваліфікаційна робота присвячена розробці методики дослідження та реалізуючої її програми для автоматизації процесу моделювання та аналізу основних властивостей складної керованої позитивної динамічної системи, поведінка якої описується дискретною математичною моделлю, що представляється лінійним неоднорідним векторно-матричним різницевим рівнянням із матрицями сталих коефіцієнтів та може включати в себе елементи з різних галузей, такі як техніка, економіка, біологія тощо. Дослідження проведено з метою полегшення трудомістких обчислень при аналізі та оцінюванні ключових характеристик моделі та основних властивостей зазначеної складної системи (позитивність, асимптотична стійкість розв’язків та керованість системи).

Структурно кваліфікаційна робота складається з 3 розділів.

Перший розділ роботи присвячено висвітленню основних теоретичних відомостей, розкриттю понять та визначень теорії автоматичного керування, використовуваних при описі складної керованої динамічної системи. Крім того, в даному розділі наведено результати аналізу основних доступних методів розв'язування систем рівнянь математичних моделей динамічних систем, на окремі з яких спиралася пропонована у другому розділі роботи методика проведення аналізу математичної моделі та основних властивостей складної керованої позитивної динамічної системи й отримуваних за нею результатів.

У другому розділі наведено основні характеристики об’єкта дослідження – складної керованої позитивної динамічної системи – та використовуваної для опису її поведінки дискретної математичної моделі, висвітлено основні вимоги до математичної моделі та засоби їх досягнення, розкрито доступні керування для позитивної системи та виникаючі на їх основі задачі керування, представлено основні етапи пропонованої в роботі

методики проведення аналізу математичної моделі та властивостей позитивності й керованості складної позитивної динамічної системи й отримуваних за нею результатів, на якій ґрунтувалась реалізація розробленого в роботі програмного забезпечення, що дозволяє автоматизувати процес здійснення відповідних досліджень.

Третій розділ кваліфікаційної роботи присвячено висвітленню особливостей програмної реалізації розробленого програмного продукту – програмної системи автоматизації процесу моделювання та аналізу основних властивостей складної керованої позитивної динамічної системи – за представленою методикою здійснення дослідження, виділенню основних переваг та особливостей використання обраної мови програмування, наданню детального опису інтерфейсу та інструкції з використання створеного продукту для кінцевого користувача. Також в даному розділі наведено результати проведеного обчислювального експерименту на моделі із використанням розробленої програмної системи.

Отримані в кваліфікаційній роботі результати вказують на ефективність та придатність розробленої методики та програмного забезпечення для подальших досліджень та застосування в практичних задачах керування складними динамічними системами, а, отже, є важливими з точки зору наукових досліджень і практичного застосування. Розроблене програмне забезпечення може бути використано при дослідженні промислових, економічних, біологічних, екологічних процесів, у робототехніці, системах керування й регулювання і т. ін.

ПЕРЕЛІК ПОСИЛАНЬ

1. Абраменко І. Г., Абраменко Д. І. Теорія автоматичного керування. Харків : ХНАМГ, 2008. 178 с.

2. Алілуйко А. М., Мазко О. Г. Інваріантні конуси та стійкість лінійних динамічних систем. *Український математичний журн*ал, 2006. Т. 58, № 11. С. 1446–1461.

3. Блохін Л. М., Буриченко М. Ю. Статистична динаміка систем управління. Київ : НАУ, 2003. 208 с.

4. Бурау Н. І., Півторак Д. О. Теорія автоматичного управління : практикум. Київ : КПІ ім. Ігоря Сікорського, 2021. 57 с.

5. Гапак О. М. Теорія керування в технічних системах . конспект лекцій. Ужгород : «АУТДОР-ШАРК», 2021. 70 с.

6. Гоголюк П. Ф., Гречин Т. М. Теорія автоматичного керування : підручник. Львів : Видавництво Національного університету «Львівська політехніка», 2008. 285 с.

7. Гуров А. П., Ольшевський С. І., Черно О. О., Бугрім Л. І. Теорія автоматичного керування : навч. посіб. у 2 ч. Ч. 1. Миколаїв : НУК ім. адмірала Макарова, 2018. 111 с.

8. Жалдак М. І., Триус Ю. В. Основи теорії і методів оптимізації : навч. посіб. Черкаси : Брама-Україна, 2005. 608 с.

9. Жуковська О.А., Новицький В. В. Неокласичні динамічні керовані системи. Лінійні моделі. *Вопросы аналитической механики и ее применений*: *Праці Інституту математики НАН України*. Київ : Ін-т математики НАН України, 1999. Т. 26. С. 84–93.

10. Задачин В. М., Конюшенко І. Г. Чисельні методи : навч. посіб. Харків : Вид. ХНЕУ ім. С. Кузнеця, 2014. 180 с.

11. Колесницький О. К., Арсенюк І. Р., Месюра В. І. Чисельні методи : навч. посіб. Вінниця : ВНТУ, 2017. 130 с.

12. Кондратьева Н. А., Леонтьева В. В., Чопоров С. В. К вопросу построения автоматизированной системы исследования сложных систем. *Матеріали міжнародної конференції «Dynamical System Modeling and Stability Investigation» – DSMSI-2007* (22-25 травня 2007 р., м. Київ). Київ : КНУ ім. Т. Шевченко, 2007. С. 377.

13. Копей В. Б. Мова програмування Python для інженерів і науковців : навч. посіб. Івано-Франківськ : ІФНТУНГ, 2019. 274 с.

14. Коржик М. В. Моделювання об’єктів та систем керування засобами MatLab : навч. посіб. для студ. вищ. навч. закл. Київ : НТУУ «КПІ», 2016. 174 с.

15. Кундрат А. М., Кундрат М. М. Науково-технічні обчислення засобами MathCAD та MS Excel : навч. посіб. Рівне : НУВГП, 2014. 252 с.

16. Лазарєв Ю. Ф. Математичні моделі та методи теоретичного дослідження стаціонарних лінійних динамічних систем. Конспект лекцій. Київ : КПІ, 1991. 156 с.

17. Лазарєв Ю. Ф. Моделювання динамічних систем у Matlab. Електронний навчальний посібник. Київ : НТУУ «КПІ», 2011. 421 с.

18. Лазарєв Ю. Ф. Matlab 5.x. Київ : Вид. група BHV, 2000. 384 с.

19. Леонтьева В. В. Математическая модель динамики функционирования позитивных систем балансового типа. *Зб. наук. праць. Вісник ЗНУ*. Запоріжжя : ЗНУ, 2008. №1 С. 118–124.

20. Леонтьева В. В., Кондратьева Н. А. Разомкнутая дискретная математическая модель позитивных динамических систем балансового типа и ее анализ. *Зб. наук. праць. Вісник ЗНУ*. Запоріжжя : ЗНУ, 2009. №1. С. 132–137.

21. Леонтьева В. В., Кондратьева Н. А. Управление в непрерывной математической модели позитивной динамической системы балансового типа. *Вестник Херсонского национального технического университета: Сб. научных статей*. Херсон : ХНТУ, 2009. Вып. 2 (35). С. 273–278.

22. Леонтьева В.В., Кондратьева Н.А. Управляемость позитивной динамической системы балансового типа. *Зб. наук. праць. Вісник ЗНУ*. Запоріжжя : ЗНУ. 2011. № 1. С. 58–66.

23. Лиходєєв О. С., Сидоренко І. І. Математичне моделювання мехатронних систем за допомогою Simulink та SimPoweRSystem. *Збірник наукових праць Національної академії Національної гвардії України*, 2012. № 1(19), С. 88-92.

24. Нікітенко О. М. Maple. Розв’язання інженерних та наукових задач : навч. посіб. Харків : ХНУРЕ, 2014. 289 с.

25. Новицький В. В. Декомпозиція та керування в лінійних системах. Київ : Ін-т математики НАН України, 1995. 150 с.

26. Олешко М. І., Соломко Н. О. Теорія автоматичного керування : навч. посіб. Ніжин: НДУ ім. М. Гоголя, 2019. 284 с

27. Перестюк М. О., Станжицький О. М., Капустян О. В. Екстремальні задачі : навч. посіб. Київ : ВПЦ «Київський університет», 2004. 50 с.

28. Перестюк М. О., Станжицький О. М., Капустян О. В. Задачі оптимального керування : навч. посіб. Київ : ТВіМС, 2004. 55 с.

29. Попович М. Г., Ковальчук О. В. Теорія автоматичного керування : підручник. Київ : Либідь, 1997. 544 с.

30. Сорока К. О. Теорія автоматичного керування : навч. посіб. Харків : ХНАМГ, 2006. 187 с.

31. Сорока К. О. Теорія автоматичного керування і комп’ютерне моделювання (неперервні лінійні системи). Ч. 1. Основи теорії систем автоматичного керування : навч. посіб. Харків : ФОП Тімченко, 2010. 218 с.

32. Сорока К. О. Теорія автоматичного керування і комп’ютерне моделювання (неперервні лінійні системи). Ч. 2. Аналіз систем автоматичного керування засобами комп’ютерного моделювання : навч. посіб. Харків : ФОП Тімченко, 2010. 156 с.

33. Стенцель Й. І. Математичне моделювання технологічних об'єктів керування : навч. посіб. Київ: ІСДО, 1993. 328 с.

34. Штіфзон О. Й., Новіков П. В., Бунь В. П. Теорія автоматичного управління : навч. посіб. Київ : КПІ ім. Ігоря Сікорського, 2020. 144 с.

35. Юрченко І. В., Сікора В. С. Програмування мовою Python: навч. посіб. Чернівці: Чернівецький національний університет, 2022. 104 с.

36. Яковенко А. В. Основи програмування. Python. Ч. 1 : підручн. для студ. спеціальності 122 «Комп'ютерні науки», спеціалізації «Інформаційні технології в біології та медицині». Київ : КПІ ім. Ігоря Сікорського, 2018. 195 с.

37. Joshi Prateek. Artificial Intelligence with Python. Birmingham : Packt, 2017. 437 p.

38. Kaczorek T. Some recent developments in positive systems. *Proc. 7th Conf. on Dynamical Systems: Theory and Applications*. Łydź, 2003. P. 25–35.

39. Kvakernaak H., Siwan R. Linear Optimal Control Systems. New York: Wiley-lnterscience, 1972. 575 pp.

40. Luenberger D. G. Introduction to Dynamic Systems: Theory, Models and Applications. New York: John Wiley & Sons, 1979. 446 pp.

41. Schuppen J. H. Control and System Theory of Positive Systems. Amsterdam : The Vrije Universitei, 2007. 245 p.

ДОДАТОК А

Лістинг програмного коду для введення вхідних даних моделі та перевірки основних вимог моделі

```
        import numpy as np
from numpy import linalg as LA
import sys

def vvod_matrix(a):   # введення будь-якої матриці. A - розмірність.
   #A=[[0]*a]*a
   A=np.zeros((a, a))
   for i in range(a):
      for j in range(a):

      #a=1
         A[i][j]=float(input(f'введіть ({i+1},{j+1}) елемент\n'))
         #print(A,i,j)
   return(A)

def check_for_non_negative_elements(A):
   # Перевіряємо, що всі елементи матриці A невід'ємні
   if np.all(A >= 0):
      print("Всі елементи >= 0\n")
   else:
      print("Всі елементи < 0\n")
      print("Отже,матриця не є позитивною")
      sys.exit()

def reverse_matrx(A,a):
   # Створюємо одиничну матрицю
   E = np.eye(a)
   # Обчислюємо зворотну матрицю (E-A)^(-1)
   B = np.linalg.inv(E - A)
   print('інвертирована матриця(E-матриця)^(-1):\n\n',B,'\n')
   # Перевіряємо, що всі елементи матриці B невід'ємні
   if np.all(B >= 0):
      print("(E-Матриця)^(-1) >= 0\n")
   else:
      print("(E-Матриця)^(-1) < 0\n")

def check_for_non_negativity(A):
   # Перевіряємо, чи є матриця невід'ємною
   if np.all(A >= 0):
      print("Матриця є невід'ємною")
   else:
      print("Матриця не є невід'ємною")
```

```
        sys.exit()

def diagon_elem_less_than_one(A):
    # Перевіряємо, що всі діагональні елементи матриці менше одиниці
    if np.all(np.diag(A) < 1):
        print("Всі диагональні елементи матриці меньше одиниці")
    else:
        print("Не всі диагональні елементи матриці меньше одиниці")
        print('Отже,матриця не є позитивною')
        sys.exit()

def matrix_ryad(A,n):
    Ak = np.identity(A.shape[0])  # одинична матриця розмірності A
    S = Ak
    for k in range(1, n+1):
        Ak = np.dot(Ak, A)  # обчислення A^k
        S = np.add(S, Ak)  # підсумовування A^k з попередніми доданками
    print('матричний ряд збігається до:\n',S)


def vlasne_znach(A):
    #Метод .eigvals() из модуля numpy.linalg в
    #Python обчислює власні значення квадратної матриці.

    l=LA.eigvals(A)

    print('Найбільше за модулем власне значення хар.рівняння:\n',l)


a=int(input('Введите розмірність матриці A:\n'))

A=(vvod_matrix(a))


print('Введена матрица A:\n',A)


I = np.eye(a)


b=int(input('Введите розмірність матриці B:\n'))
B=(vvod_matrix(b))
print('Введена матрица B:\n',B)
A_tild=np.dot(np.linalg.inv(I-B),(A-B))
print(A-B)
print('(I-B)^-1*(A-B)=\n',np.dot(np.linalg.inv(I-B),(A-B)))

input_txt=["A","B","A-B","(I-B)^-1*(A-B)"]
input_matrx=[A,B,A-B,A_tild]
n=int(input('\nВведіть кількість ітерацій для ряду\n'))
```

```
for i in range(4):
    print("Розрахуємо продуктивність та невід'ємність для матриці",input_txt[i])
    check_for_non_negative_elements(input_matrx[i])
    reverse_matrx(input_matrx[i],a)
    check_for_non_negativity(input_matrx[i])
    diagon_elem_less_than_one(input_matrx[i])
    matrix_ryad(input_matrx[i],n)
    vlasne_znach(input_matrx[i])
    print()

print('Якщо всі умови виконуються, можете переходити до програми 2.')
```

ДОДАТОК Б

Лістинг програмного коду для розв’язання основних задач за моделлю

```
import numpy as np
import matplotlib.pyplot as plt
import math
from numpy import linalg as LA
import sys

def print_matrx_2(A):   #виводить двовимірну матрицю на екран
    print('(\n',end='')
    for i in range(2):
        for j in range(2):
            if j!=1:
                print(A[i][j],end=',')
            else:
                print(A[i][j],end='')
        print('\n',end='')
    print(')')

def print_matrx(A,a):   #виводить матрицю на екран.a-це розмірність матриці
    print('(\n',end='')
    for i in range(a):
        for j in range(a):
            if j!=a-1:
                print(A[i][j],end=',')
            else:
                print(A[i][j],end='')
        print('\n',end='')
    print(')')



def vvod_matrix_2():   # введення двовимірної матриці.
    A=[[0,0],
      [0,0]]

    for i in range(2):
        for j in range(2):

        #a=1
            A[i][j]=float(input(f'введіть ({i+1},{j+1}) елемент\n'))
    return(A)

def vvod_matrix(a):   # введення будь-якої квадратної матриці. a- розмірність.
    #A=[[0]*a]*a
    A=np.zeros((a, a))
```

```
    for i in range(a):
        for j in range(a):

        #a=1
            A[i][j]=float(input(f'введіть ({i+1},{j+1}) елемент\n'))
            #print(A,i,j)
    return(A)

def A_tild(A,B,b):#b- розмірність матриці b
    #print(A-B)

    return(np.dot(B_tild(B,b),(A-B))    )

def B_tild(Matrx,b):#b- розмірність матриці b
    I=np.eye(b)
    #print(I)
    #print('i-b:',I-Matrx)
    return(np.linalg.inv(I-Matrx))

def vvod_c(c):
    C=np.zeros((1,c))
    for i in range(c):
        C[0][i]=float(input(f'введіть ({i+1},{1}) елемент\n'))

    return(C)

def stepen_t(A,t,a):
    if t==1:
        return(A)
    A_1=A
    if t==0:
        A_1=np.eye(a)
        a=A.shape[0]
        A_1=np.eye(a)
    else:
        for i in range(t-1):

            A_1=np.dot(A_1,A)

    return(A_1)

def vvod_func_c(c):

    C_1=[[0]*2]*c
    C=np.zeros((c,2))
    for i in range(c):
        print(f'введіть C0({i+1}):')
        for j in range(2):
            C_1[i][j]=input()
            print(f'введіть коефіцієнти C_i({i+1})')
            C[i][j]=C_1[i][j]
```

```
    return(C)

def func_c(C_given,c,t):
    #omega=80          тут треба вводити омегу,при гармонічн.функ. C
    C=[0]*c
    for i in range(c):
        C[i]=[0]*t
    ans=input(str('виберіть вид функції C(1-лінійний,2-квадратичний,3-гармонічний\n'))
    if ans=='3':
        omega=int(input('Введіть омегу(Sin(omega *C))\n'))


    for i in range(c):

        for j in range(t):
                if ans=='1':
                    C[i][j]=round((C_given[i][0]+(C_given[i][1]*j)),5)          #round(C_0+C_step*i,3)
            #тут замінити функцію C на
            #квадратичну:
                else:
                    if ans=='2':
                        C[i][j]=round((C_given[i][0]+(C_given[i][1]*j*j)),5)
                    else:
                        if ans=='3':
                            C[i][j]=round((C_given[i][0]+(C_given[i][1]*math.sin(omega*j))),5)
            #C[i][j]=round((C_given[i][0]+(C_given[i][1]*j*j)),5)

            #гармоничну:
            ##C[i][j]=round((C_given[i][0]+(C_given[i][1]*sin(omega*j))),5)

    return(C)

def vvod_x_0(x_0_raz):
    #x0=[0]*x_0_raz
    x0=np.zeros((x_0_raz,1))
    for i in range(x_0_raz):
        x0[i]=float(input(f'Введіть {i+1} елемент x0\n'))
    return(x0)




#тут модуль з першої програми:

def check_for_non_negative_elements(A):
    # Перевіряємо, що всі елементи матриці A невід'ємні
    check=0
    if np.all(A >= 0):
        print("Всі елементи >= 0\n")
    else:
        print("Має елементи < 0\n")
        print("Отже,матриця не є позитивною")
```

```
        check+=1
    return(check)

def reverse_matrx(A,a):
    # Створюємо одиничну матрицю
    E = np.eye(a)
    # Обчислюємо обернену матрицю (E-A)^(-1)
    B = np.linalg.inv(E - A)
    check=0
    print('інвертована матриця(E-матриця)^(-1):\n\n',B,'\n')
    # Перевіряємо, що всі елементи матриці B невід'ємні
    if np.all(B >= 0):
        print("(E-Матриця)^(-1) >= 0\n")
    else:
        print("(E-Матриця)^(-1) < 0\n")
        print("Отже,матриця не є продуктивною\n")
        check+=1
    return(check)

def check_for_non_negativity(A):
    # Перевіряємо, чи є матриця невід'ємною
    if np.all(A >= 0):
        print("Матриця є невід'ємною")
    else:
        print("Матриця не є невід'ємною")
        #sys.exit()

def diagon_elem_less_than_one(A):
    #Перевіряємо, що всі діагональні елементи матриці менше одиниці
    check=0
    if np.all(np.diag(A) < 1):
        print("Всі диагональні елементи матриці меньше одиниці")
    else:
        print("Не всі диагональні елементи матриці меньше одиниці")
        print('Отже,матриця не є позитивною')
        check+=1
    return(check)
        #sys.exit()

def matrix_ryad(A,n):
    Ak = np.identity(A.shape[0])  # одинична матриця розмірності A
    S = Ak
    for k in range(1, n+1):
        Ak = np.dot(Ak, A)  # обчислення A^k
        S = np.add(S, Ak)  # підсумовування A^k з попередніми доданками
    print('матричний ряд збігається до:\n',S)


def vlasne_znach(A):
    #Метод .eigvals() з модуля numpy.linalg в
    #Python обчислює власні значення квадратної матриці.
```

```
    l=LA.eigvals(A)

    print('Найбільше за модулем власне занчення хар.рівняння:\n',l)


def dimploma_1(A,B,a):
    I = np.eye(a)
    A_minus_B=(A-B)
    print('A-B\n',A-B)
    A_tild=np.dot(np.linalg.inv(I-B),(A_minus_B))
    print('A_tild=\n',A_tild)
    #reverse_matrx
    # Обчислюємо обернену матрицу (E-A)^(-1)
    #INVERT = np.linalg.inv(E - A)
    input_txt=["A","B","A-B","(I-B)^-1*(A-B)"]
    input_matrx=[A,B,A-B,A_tild]
    n=40
    check=0
    for i in range(4):
        print("Розрахуємо продуктивність та невід'ємність для матриці",input_txt[i])
        check+=check_for_non_negative_elements(input_matrx[i])
        check+=reverse_matrx(input_matrx[i],a)
        check_for_non_negativity(input_matrx[i])
        diagon_elem_less_than_one(input_matrx[i])
        matrix_ryad(input_matrx[i],n)
        vlasne_znach(input_matrx[i])
        if check>0:
            print('одна з вимог продуктивності не виконується')
            return(int(1))
            break

    return(int(0))   #1 якщо не продукт,0 якщо продукт.
###закінчився


t=int(input('Введіть t\n'))


c=int(input('Введіть розмірність вектора C:\n'))
C_given=vvod_func_c(c)   #given- дано.

C=func_c(C_given,c,t)
print('C=\n',C)



a=int(input('Введіть розмірність матриці A:\n'))
A_1 = np.zeros((a, a))
print('Ваша матриця:')

print_matrx(A_1,a)
A=(vvod_matrix(a))              #A-основна введена матриця A.Так само і B
```

```
print('Введена матриця A:\n',A)

b=int(input('Введіть розмірність матриці B:\n'))
B_1 = np.zeros((b, b))
print_matrx(B_1,b)

B=vvod_matrix(b)            #тут

print('Введена матриця B:\n',B)

x_0_raz=int(input('ведіть розмірність X0\n'))#raz=размірність
x0=vvod_x_0(x_0_raz)
print('Ви ввели:\n',x0)




while True:

    summa=0



    X_mas = [[0] * t for _ in range(x_0_raz)]

    I=np.eye(a)

    dif=(A-B)

    inv=np.linalg.inv(I-B)



    left_bracket=np.dot(inv,dif)#left_bracket-ліві дужки
    print('inv*dif:\n',left_bracket)#inv-інвертована,dif-різниця A-B




    def sewing_c(C,c,j):#sewing-(с англ.) шити. Зшиваємо два значення для вектора C
        vect=np.zeros((c,1))
        for i in range(c):
            vect[i]=C[i][j]
        return(vect)

    def concatenate_matrices(matrix1, matrix2):
        # Перевіряємо, що матриці мають однакову кількість рядків
        if matrix1.shape[0] != matrix2.shape[0]:
            raise ValueError("Матриці повинні мати однакову кількість рядків.")
```

```
    # Об'єднуємо матриці
    result_matrix = np.concatenate((matrix1, matrix2), axis=1)

    return result_matrix

B_tild_np=B_tild(B,b)
A_tild_np=A_tild(A,B,b)
#print("B_tild=",B_tild_np)
#print("A_tild*B_tild",np.dot(A_tild_np,B_tild_np))
result = concatenate_matrices(B_tild_np,np.dot(A_tild_np,B_tild_np))
print('матриця керованості:\n',result)
rank = np.linalg.matrix_rank(result)

if rank==a:
    print('ранг матриці дорівнює',rank)
    print('отже,система є повністю керованою')

left_bracket=np.dot(inv,dif)#left_bracket-левые скобки
print('inv*dif:\n',left_bracket)#inv-інвертована,dif-різниця A-B

print(f' t |',end=' ')
for h in range(x_0_raz):
    print(f'X(t){h+1}        | ',end='')
print()
for k in range(1,t+1):
    left_side=np.dot(stepen_t(left_bracket,k,a),x0)
    for i in range(1,k+1):
        right_bracket=stepen_t(left_bracket,k-i,a)
        summa=np.dot(np.dot(right_bracket,inv),sewing_c(C,c,i-1))+summa
        right_bracket=0
   # print(f'X{k} для {m+1}-го графіка=\n',
    if k<10:
        print(f' {k} | ',end='')
    else:
        print(f' {k}| ',end='')
    for m in range(x_0_raz):
        print((summa+left_side)[m],end=' | ')
    print()

    for i in range(x_0_raz):

        X_mas[i][k-1]=float((summa+left_side)[i])

    summa=0


plt.grid(True)  #сетка
plt.xlim(0, None)#старт координат
#plt.ylim(min(X_mas_1)-0.3,max(X_mas_1)+0.3)#(4, 5)
#plt.ylim(min(X_mas[0])-1,max(X_mas[0])+1)
#Y_mas = [1,2,3,4,5]
#Y_mas = [1,2,3,5,6]
```

```
    #Y_mas = [0.5,4]
    #plt.ylim(Y_mas[0], Y_mas[1])
    plt.ylim(round(min(X_mas[0]))-1,round(max(X_mas[0])))
    #Y_mas = [4,5,6,7,8,10,11]
    plt.xticks(range(0, t))  #ціна поділу 1


    j=1

    for i in range(x_0_raz):
        plt.scatter(range(len(X_mas[j-1])),X_mas[i],label=('X(i)',j))
        j+=1
    X_star=[0]*x_0_raz
    for i in range(x_0_raz):
        X_star[i]=round(float((np.dot(np.linalg.inv(I-A),sewing_c(C,c,0))[i])),3)
        plt.axhline(X_star[i], color='r', linestyle='-',label=f'X*{i+1}={X_star[i]}')
    plt.xlabel('t-1')
    plt.ylabel('X(i)')
    plt.legend()
    plt.tight_layout()
    plt.show()

    ansv1=input(str('Чи необхідно покращення вхідних параметрів?так/ні.\n'))
    if ansv1=='ні':
        break
    else:
        if ansv1=='так':
            ansv2=input(str('матриця А є продуктивною та можна її змінювати?так/ні\n'))

            if ansv2=='так':



                    ansv3=input(str('"збільшемо чи зменьшимо значення матриці А?
\nзбільшемо-1\nзменьшимо-2\n"))

                    A_new=np.copy(A)
                    #print('Введена матриця А:\n',A)
                    #print('Введена матриця A_new:\n',A_new)
                    if ansv3=='1':
                        for i in range(a):
                            for j in range(a):
                                A_new[i][j]=round(   ((A[i][j]+0.99999999)/2),5)

                    else:
                        if ansv3=='2':
                            for i in range(a):
                                for j in range(a):
                                    A_new[i][j]=round(   ((A[i][j]+0.0000001)/2),5)
                    #print('Введена матриця А:\n',A)
                    #print('Введена матриця A_new:\n',A_new)
                    Sum_i=np.sum(A_new, axis=1)#суми рядків
```

```
            Sum_j=np.sum(A_new, axis=0)#суми столбців
            count_=0
            for p in range(a):
                if (Sum_i[p]>=1).any() or (Sum_j[p]>=1).any():
                    count_+=1
            while count_>=1:



                if count_>=1:
                    if ansv3=='1':
                        for i in range(a):
                            for j in range(a):
                                #A_new[i][j]=round(   ((A[i][j]+A_new[i][j])/2),5)
                                A_new[i][j]=((A[i][j]+A_new[i][j])/2)

                    else:
                        if ansv3=='2':
                            for i in range(a):
                                for j in range(a):
                                    #A_new[i][j]=round(   ((A[i][j]+A_new[i][j])/2),5)
                                    A_new[i][j]=((A[i][j]+A_new[i][j])/2)
                    #print('Введена матриця A:\n',A)
                    #print('Введена матриця A_new:\n',A_new)
                    Sum_i=np.sum(A_new, axis=1)#суми строк
                    Sum_j=np.sum(A_new, axis=0)#суми столбців
                    #print('Sum_i',Sum_i)
                    #print('Sum_j',Sum_j)
                    count_=0
                    for p in range(a):
                        #if (Sum_i[p]>=1).any() or (Sum_j[p]>=1).any():
                        #print(f'Sum_i[{p}]',Sum_i[p])
                        #print(f'Sum_j[{p}]',Sum_j[p])
                        if (Sum_i[p] > 1) or (Sum_j[p] > 1):
                            count_+=1


                A=A_new
                print('Введена матриця A:\n',A)

    else:
        if ansv2=='ні':

            A_k=np.zeros((a,a))
            print('"Якщо матриця A не продуктивна або ми не можемо"',
            '"її змінювати,то "')
            print('Переходимо до замкненої моделі')
            print('Що ви хочете зробити з значеннями елементів вектора X(t)?')
            ansv4=input(str('Збільшити-1,зменьшити-2\n'))
            if ansv4=='1':
                if a==1:
                    A = A.reshape((1, 1))
```

```
            for j in range(a):
                #print(np.sum(A, axis=0))
                column_sums = np.sum(A, axis=0)
                #print(column_sums)
                for i in range(a):
                    A_k[i][j]=column_sums[j]+0.02

        else:
            for j in range(a):
                #print(np.sum(A, axis=0))
                column_sums = np.sum(A, axis=0)
                #print(column_sums)
                for i in range(a):
                    A_k[i][j]=(column_sums[j]+column_sums[j]-1)/a
                    #if A_k[i][j]-A[i][j]<=0:
                      # A_k[i][j]=column_sums[j]/2
        A=A-A_k
        print('Матриця k=\n',A_k)
        # print('Матриця A=\n',A)



else:
    if ansv4=='2':
        for j in range(a):
                #print(np.sum(A, axis=0))
                column_sums = np.sum(A, axis=0)
                #print(column_sums)
                for i in range(a):
                    A_k[i][j]=(column_sums[j]+1)/a
                    #if A_k[i][j]-A[i][j]<=0:
                      # A_k[i][j]=column_sums[j]/2
        A=A-A_k
        print('Матриця k=\n',A_k)
```

Декларація
академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Старовойтов Олександр Вікторович

студент 4 курсу, денної форми навчання, математичного факультету, спеціальності 113 прикладна математика, адреса електронної пошти mowniw@gmail.com,

– підтверджую, що написана мною кваліфікаційна робота бакалавра на тему «Автоматизація дослідження основних властивостей складних керованих динамічних систем» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений/ознайомлена;

– заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

– згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи а також на архівування моєї роботи в базі даних цієї системи.

Студент

(дата)	(підпис)	(прізвище, ініціали)

Науковий керівник

(дата)	(підпис)	(прізвище, ініціали)