МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра фундаментальної та прикладної математики

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«Автоматизація процесу моделювання поведінки та дослідження змін основних характеристик складної керованої позитивної динамічної системи»**

Виконав: студент 2 курсу, групи 8.1130

Спеціальності 113 прикладна математика
(шифр і назва спеціальності)

освітньої програми Прикладна математика
(назва освітньої програми)

Я. В. Чайкін
(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 – 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра фундоментальної та прикладної математики

Рівень вищої освіти магістр

Спеціальність 113 прикладна математика
(шифр і назва)

Освітня програма прикладна математика

ЗАТВЕРДЖУЮ

Завідувач кафедри фундоментальної та прикладної математики, д.т.н., доцент

Гребенюк С.М.

(підпис)

« » 2021 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Чайкіну Ярославу Володимировичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проєкту) Автоматизація процесу моделювання поведінки та дослідження змін основних характеристик складної керованої позитивної динамічної системи

керівник роботи (проєкту) Леонтьєва Вікторія Володимирівна, к.ф.-м.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 09 » червня 2021 року № 851-с

2. Строк подання студентом роботи

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) Аналіз і даних
2) Реалізація
3) Перевірка працездатності програмного продукту

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
1			
2			
3			

7. Дата видачі завдання 10.06.2021

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	15.06.2021	
2.	Збір вихідних даних.	25.06.2021	
3.	Обробка методичних та теоретичних	03.09.2021	
	джерел.		
4.	Розробка першого розділу.	15.10.2021	
5.	Розробка другого розділу.	14.11.2021	
6.	Розробка третього та четвертого розділу.	20.11.2021	
7.	Оформлення та нормоконтроль	02.12.2021	
	кваліфікаційної роботи.		
8.	Захист кваліфікаційної роботи.	16.12.2021	

Студент _____ Я. В. Чайкін
(підпис) (ініціали та прізвище)

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

Нормоконтроль пройдено

Нормоконтролер _____ В.В. Леонтьєва
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Автоматизація процесу моделювання поведінки та дослідження змін основних характеристик складної керованої позитивної динамічної системи»: 66 с., 6 рис., 22 джерел, 2 додатки.

РОЗІМКНЕНА МОДЕЛЬ ПОЗИТИВНОЇ СИСТЕМИ, ЗАМКНЕНА МОДЕЛЬ ПОЗИТИВНОЇ СИСТЕМИ, МОДЕЛЮВАННЯ, АВТОМАТИЗАЦІЯ, АЛГОРИТМІЗАЦІЯ.

Мета роботи – розробка алгоритму та програмного продукту моделювання поведінки складної позитивної динамічної системи та розробці програмного продукту на основі отриманого алгоритму.

Об’єктом дослідження виступає розімкнена та замкнена дискретна та неперервна математична модель позитивної динамічної системи.

Структурно робота складається з чотирьох розділів. В першому розділі проводиться огляд сучасного стану позитивних систем. У другому розділі розкривається особливості побудови математичної моделі позитивної динамічної системи, надається представлення про властивості позитивних систем, а також розкривається особливості керування даними системами. В третьому розділі розглядається процес алгоритмізації моделювання позитивної системи на основі інформації наведеної в другому розділі. В четвертому розділі наведені результати роботи розробленого програмного продукту на основі алгоритму отриманого в розділі 3, а також описані особливості реалізації і структура даного програмного продукту.

Результатом виконання роботи є розроблений програмний продукт математичного моделювання та керування.

SUMMARY

"Automation of the process of modeling behavior and research of changes in the main characteristics of a complex controlled positive dynamical system": 66 p., 6 figs., 22 sources, 2 applications.

OPEN POSITIVE SYSTEM MODEL, CLOSED POSITIVE SYSTEM MODEL, MODELING, AUTOMATION, ALGORITHMIZATION.

The purpose of the work is to develop an algorithm and software product for modeling the behavior of a complex positive dynamic system and to develop a software product based on the obtained algorithm.

The object of research is an open and closed discrete and continuous mathematical model of a positive dynamical system.

Structurally, the work consists of four sections. The first section reviews the current state of positive systems. The second section reveals the features of building a mathematical model of a positive dynamic system, provides an idea of the properties of positive systems, as well as features of the management of these systems. The third section considers the process of algorithmization of positive system modeling based on the information presented in the second section. The fourth section presents the results of the developed software product based on the algorithm obtained in section 3, as well as describes the features of the implementation and structure of this software product.

The result of the work is a developed software product of mathematical modeling and control.

ЗМІСТ

ВСТУП

Завданням кваліфікаційної роботи є розробка алгоритму та реалізуючої його програми процесу моделювання поведінки, визначення, дослідження й покращення основних вихідних характеристик складної керованої позитивної динамічної системи, яка функціонує в умовах дискретного та неперервного часу.

Для досягнення поставленого завдання кваліфікаційної роботи є необхідність у вирішені наступних завдань:

а) аналіз існуючого стану розвитку предметної області дослідження та застосовуваного до цього математичного апарату;

б) засновуючись на результатах аналізу вхідних параметрів та характеристик математичної моделі визначити тип позитивної динамічної системи – функціонуюча в умовах повної/неповної інформації про стан об'єкта; із заданим/не заданим бажаним станом об'єкта; потребує/не потребує стабілізації; потребує/не потребує покращення динамічних властивостей системи та її вихідних характеристик;

в) проведення аналізу існуючих розімкнених дискретних та неперервних моделей позитивної динамічної системи з позиції володіння властивостями та визначення основних підходів й методів керування та регулювання, за використанням яких досліджувана система буде позитивна, керована, спостережувана, індефікована та параметрично чутлива;

г) за результатами аналізу визначених властивостей позитивної динамічної системи сформулювати алгоритм та реалізуючу його програму.

Структурно робота складається з 4 розділів. В першому розділі проводиться огляд сучасного стану позитивних систем. У другому розділі розкривається особливості побудови математичної моделі позитивної

динамічної системи, надається представлення про властивості позитивних систем, а також розкривається особливості керування даними системами. В третьому розділі розглядається процес алгоритмізації моделювання позитивної системи на основі інформації наведеної в другому розділі. В четвертому розділі наведені результати роботи розробленого програмного продукту на основі алгоритму отриманого в розділі 3, а також описані особливості реалізації і структура даного програмного продукту.

1 АНАЛІТИЧНИЙ ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ МОДЕЛЮВАННЯ ПОВЕДІНКИ, ДОСЛІДЖЕННЯ, КЕРУВАННЯ Й РЕГУЛЮВАННЯ ПОЗИТИВНИМИ ДІНАМІЧНИМИ СИСТЕМАМИ, АЛГОРИТМІЗАЦІЇ ТА АВТОМАТИЗАЦІЇ ПРОЦЕСУ ЇХ МОДЕЛЮВАННЯ, АНАЛІЗУ Й ВИЗНАЧЕННЯ ОСНОВНИХ ХАРАКТЕРИСТИК

Даний розділ присвячено огляду аналітичного стану проблеми моделювання позитивних динамічних систем. В ньому буде розглянуто, що таке позитивні системи і в який сферах вони використовуються в даний час.

При моделюванні об’єктів та явищ реального світу виникає необхідність в моделюванні процесів, які мають невід’ємну природу, тобто які будуть залишатися невід’ємними протягом усього часу. Для вирішення даної проблеми використовують клас складних систем відомих як позитивні.

Позитивні системи представляють собою клас складних систем, стани яких залишаються невід’ємними протягом усіх майбутніх часів, якщо розпочати з невід’ємних початкових умов.

Позитивні системи мають широкий спектр використання в різних галузях науки та техніки, як наприклад: біологія, медицина, екологія, економіка чи системи комунікації [1-3].

Починаючи з робот Альфреда Лотки [1], в процесах пов’язаних з екологією та динамікою росту популяції позитивні системи активно використовуються. Причиною є те, що в екологічних системах, зазвичай, станами системи є популяції різних видів, які оцінюються невід’ємно.

Найбільш значну роль в сфері економіки позитивні системи відіграють при аналізі моделі “витрат-випусків”, яка була запропонована Леонтьєвим В.В. Модель “витрат-випусків” розглядає конкретне питання рівня

виробництва різних галузей економіки, які б задовольняло попит на певний продукт у конкретній економічній ситуації [2].

Не менш важливу роль в економіці, позитивні системи відіграються при досліджені моделі фон Неймана. Дана модель вводить концепцію рівноважного зростання і надає більш детальну інформацію про застосування теорії позитивних систем у моделі введення-виведення [2].

В системах комунікації позитивні системи, здебільшого, використовуються у якості керування явищами, які виникають в мережах. У якості приклада можна навести контроль перевантаження в мережах TCP та в аналізі стабільності алгоритмів розподіленого керування потужністю для стільникового зв'язку [3].

З вище наведених прикладів можна зрозуміти, що позитивні системи мають дуже широкий спектр використання, який обумовлюється не тільки необхідністю працювати з невід’ємними даними, а також через більш простий аналіз. Дана простота забезпечується зв’язком теорії позитивних систем з теорією невід’ємних матриць.

Через необхідність в більш точних рішеннях та автоматизації процесу в цілому, виникає необхідність в алгоритмізації процесу моделювання та автоматизації отримання результатів, що буде більш детально розглянуто в подальших розділах даної роботи.

2 АНАЛІЗ ОБ’ЄКТА ТА ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ ЯК ОБ’ЄКТА АЛГОРИТМІЗАЦІЇ ТА ЇЇ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Об'єктом дослідження даної роботи виступає клас складних динамічних систем, де відповідні матриці коефіцієнтів у моделях що описують поведінку складних систем, є невід'ємними та задовольняють певним умовам, які забезпечують невід'ємність рішень на нескінченному інтервалі часу. Такі системи входять до класу позитивних динамічних систем.

До таких систем відносяться багато реальних технічних, біологічних, демографічних, економічних, екологічних та ін. системи. В якості прикладу можна навести статичну та динамічну модель В. Леонтьєва, які являють собою економіко-математичну балансову модель, що характеризують міжгалузеві виробничі взаємозв'язки в економіці країни.

Предметом дослідження виступають шляхи автоматизації даного класу складних динамічних систем. Визначаються дані, які потрапляють на вхід, функції, які мають виконуватися в системі, та вихідні дані, які в результаті має отримувати кінцевий користувач.

Під автоматизацією розуміється процес побудови алгоритму розв'язання позитивної динамічної системи з наступним застосуванням даного алгоритму для розробки програми.

У наступному розділі більш детально будуть розглянуті особливості, що виникають при моделюванні позитивних систем.

2.1 Поняття, основні підходи та особливості моделювання позитивних систем

Даний розділ присвячений розгляду понять, підходів та особливостей, які пов’язанні з процесом моделювання позитивних систем.

Методи математичного моделювання широко використовуються для опису і подальшого вивчення та дослідження властивостей, структур та законів взаємодій, об’єкта або явища реального світу.

У випадку моделювання позитивних систем, об’єктом моделювання є позитивні зміні, тобто вхідні та вихідні змінні моделі, які залишаються невід’ємними в часі. До позитивних змінних можна віднести час, гроші, одиниці товарів, потоки даних в мережах чи популяцію тварин, людей або рослин.

Слід враховувати, що позитивна система, яка моделюється, складається з багатьох взаємозалежних і взаємодіючих між собою, різнорідних елементів та підсистем, які забезпечують виконання системою деякої досить складної функції, тобто дана система є складною системою [4, 5].

З вище наведеного визначення випливає, що при моделюванні позитивної системи слід враховувати взаємопов’язаність підсистем досліджуваної моделі. Кількість взаємозв'язків між n підсистемами визначається числом n^2 [4, 5].

Оскільки позитивна система не є статичною, а описує процес на часовому інтервалі, то виникає необхідність в представлені часу t в дискретному або неперервному вигляді, а саме як рівняння моделі, яке в залежності від виду часу, буде представлене неоднорідним різницевим (для дискретного часу) чи диференційним (для неперервного часу) рівняння.

Окремим випадком позитивних систем є позитивні системи балансового типу, при моделюванні яких слід враховувати наступні властивості:

- балансові співвідношення моделей, в яких описується поведінка складних систем;
- обмеження моделей на матричні коефіцієнти, які забезпечують отримання невід’ємних розв’язків, при невід’ємних вхідних даних та початкових станів системи на нескінченному інтервалі часу.

В наступному підрозділі наочно буде розглянута розімкнена модель позитивної динамічної системи, при дискретному та неперервному часі та буде проведено її аналіз.

2.2 Особливості побудови, аналіз та умови застосування розімкнених дискретної та неперервної математичних моделей позитивної системи. Напрями аналізу та покращення вхідних характеристик моделей щодо визначення типу та математичного апарату дослідження позитивної динамічної системи

Даний підрозділ присвячений особливостям побудови, аналізу та умовам застосування розімкненої математичної моделі позитивної систем.

Неоднорідна дискретна та неперервна математична модель позитивної динамічної системи може бути представлена в наступному вигляді [6 – 10]:

$$X_{t+1} = (I - B)^{-1}(A - B)X_t + (I - B)^{-1}C_t, X(0) = X_0, \quad (2.1)$$

$$\dot{X} = (I - B)^{-1}\big((A - I)X(t) + C(t)\big), X(0) = X_0, \quad (2.2)$$

або

$$X_{t+1} = \hat{A}X_t + \hat{B}C_t, \quad (2.3)$$

$$\dot{X} = \hat{A}X(t) + \hat{B}C(t). \quad (2.4)$$

Тут маємо, що час t, в залежності від виду моделі є дискретним чи неперервним; X – є вектором стану системи; C – вектор керування системи; $X(0) = X_0$ – n-мірні вектор-стовпець початкових умов системи; A,B – матриці сталих коефіцієнтів розмірності $n \times n$.

В представлених моделях (2.1) – (2.4) вихідні характеристики моделі ніяк не впливають на вхідні характеристики, тобто дані моделі є розімкненими [6 – 8].

Оскільки бажаний розв'язок моделей (2.1) – (2.4) повинен бути невід'ємним, невід'ємністю повинні характеризуватися матриці коефіцієнтів даних моделей.

Також бажаний розв'язок повинен бути асимптотично стійким, що досягається при продуктивності матриці A.

Вигляд розв'язку моделей (2.1) – (2.4), під впливом вид вектору керування C, має набувати наступного вигляду [8 – 10]:

- якщо, вектор представлений у вигляді $C = \Theta$ (де $\Theta = \left(\theta_{ij}\right)_{n\times n}$, $\theta_{ij} = 0$ при $\forall i,j = \overline{1,n}$), то розв'язок рівнянь (2.1) та (2.2) набуває вигляду:

$$X_t = \left((I-B)^{-1}(A-B)\right)^t X_0, \tag{2.5}$$

$$X(t) = e^{(I-B)^{-1}(A-I)}X_0. \tag{2.6}$$

- у випадку, коли вектор дискретної моделі $C_t = C^0 > 0$ чи неперервної моделі $C(t) = C^0 > \Theta$, то розв'язок рівнянь (2.1) та (2.2) набуває вигляду:

$$X_{t+1} = \left((I-B)^{-1}(A-B)\right)^{t+1}X_0 + X - (I-A)^{-1}(A-B)\left((I-B)^{-1}(A-B)\right)^t(I-B)^{-1}C^0\,, \tag{2.7}$$

$$X(t) = e^{(I-B)^{-1}(A-B)t}(X_0 - \mathrm{X}) + \mathrm{X}. \tag{2.8}$$

- у випадку, коли вектор C представлений, як деяка функція часу, то розв'язок рівнянь (2.1) та (2.2) набуває виду:

$$X_t = A^{t-1}B\left((A-B)X_0 + (I-A)X^*\right) + A^{t-1}\sum_{i=1}^{t-1}A^{-i}BN_i, \tag{2.9}$$

$$X(t) = e^{B(A-I)t}\left(X_0 + \int_0^t e^{-B(A-I)\tau}BC(\tau)d\tau\right). \tag{2.10}$$

В даному підрозділі була приведена загальна характеристика розімкненої моделі позитивної системи при неперервному чи дискретному часі. В наступному підрозділі буде більш детально розглянуті основні властивості позитивної системи.

2.3 Основні властивості позитивної динамічної системи – позитивність, керованість, спостережуваність, індетифікацією та параметрична чутливість: сутність та критерії їх перевірки й задоволення для досліджуваних моделей

Даний підрозділ присвячений огляду основних властивостей позитивної динамічної системи, які представлені наступними властивостями: позитивність, керованість, спостережуваність, індетифікацією та параметричною чутливістю. В даному підрозділі будуть розглянуті сутність даних властивостей, критерії їх перевірки та задоволення для досліджуваної моделі.

Для забезпечення невід'ємності розв'язків системи рівнянь, необхідним є виконання властивості невід'ємності для матриць сталих коефіцієнтів, тобто матриця A вважається невід'ємною тоді і тільки тоді, коли невід'ємними є елементи даної матриці, тобто $(a_{ij}) \geq 0$ [9].

В той же час, окрім невід'ємності, розв'язок повинен характеризуватися асимптотичною стійкістю, яка досягається при продуктивності матриці сталих коефіцієнтів.

Матриця A вважатиметься продуктивною у разі виконання одного з еквівалентних достатніх та необхідних умов [9]:

- існування оберненої матриці $(I - A)^{-1} \geq 0$;

- збіжність матричного ряду $I + A + A^2 + A^3 + \ldots = \sum_{k=0}^{\infty} A^k$ до оберненої матриці $(I - A)^{-1}$;

- у випадку, коли спектральний радіус матриці A менший за одиницю: $p(A) < 1$;

- якщо послідовні головні мінори матриці $(I - A)$ є позитивними.

В тому випадку, коли виникає необхідність в перекладі об'єкта моделювання з будь-якого початкового стану $X(t_0) = X_0$ до будь-якого кінцевого стану $X(t_k) = X_k$ за кінцевий час $t \in [t_0, t_k]$, то мова їде про керованість моделі.

Необхідною і достатньою умовою існування керованості моделі є виконання наступної рангової умови.

У разі перекладу об'єкта з будь-якого початкового стану $X(t_0) = X_0$ до будь-якого кінцевого стану $X(t_k) = X_k$ за кінцевий час $t \in [t_0, t_k]$, маємо справу з керованістю моделі, необхідною і достатньою умовою якої є виконання ранговою умови, при продуктивності матриці B [11 – 12]:

$$rang[B \vdots (AB) \vdots (A^2B) \vdots \cdots \vdots (A^{n-1}B)]_{n \times nn} = n, \quad (2.11)$$

$$rang[B \vdots (AB) \vdots (A^2B) \vdots \cdots \vdots (A^{n-1}B)]_{n \times nn} = n. \quad (2.12)$$

Проте в тому випадку, коли про стан вектору керування об'єктом недостатньо інформації, то виникає необхідність у відновленні стану вектора керування за спостережуваними вимірами змінної. Дане відновлення відоме як властивість спостережуваності, і необхідною та достатньою умовою її існування є виконання наступної рангової умови [13]:

$$rang M_y = n, \quad (2.13)$$

де матриці $M_y = [\tilde{B} \vdots (\tilde{A}\tilde{B}) \vdots (\tilde{A}^2\tilde{B}) \vdots \cdots \vdots (\tilde{A}^{n-1}\tilde{B})]_{n\times nn}$ (для системи з дискретним часом) та $M_y = [\tilde{B} \vdots (\tilde{A}\tilde{B}) \vdots (\tilde{A}^2\tilde{B}) \vdots \cdots \vdots (\tilde{A}^{n-1}\tilde{B})]_{n\times nn}$ (для системи з неперервним часом), називаються матрицями спостережуваності.

Отримана рангова умова (2.13) є аналогічною до рангових умов (2.11) та (2.12), тоді на основі праці\праць [13], можливо стверджувати, що властивість спостережуваності симетрична властивості керованості.

Виходячи з симетричності спостереження до керованості, можна стверджувати про дуальність спостереження та керованості. Дана дуальність полягає в тому, що система вважається повністю спостережуваною в тому випадку, коли система спостереження буде цілком керованою.

Необхідно зазначити, що вхідною системою може виступати система керування (спостереження), тоді дуальною виступатиме система спостереження (керування).

Для уникнення проблеми зміни динамічних властивостей досліджуваної системи та отримання некоректних результатів функціонування, використовується властивість ідентифікації щодо змінених параметрів.

Ідентифікація полягає в тому, що за допомогою вектору виходу, при існуванні вхідного сигналу, матриця параметрів системи може бути відновлена з точністю до інваріантів канонічної форми спостереження за кінцевий відрізок часу та по часовій послідовності $Y = Y(t)$, проте, якщо не всі, або жодна з матриць параметрів не може бути відновлена за по часовій послідовності, то досліджувана система є неідентифікованою.

Згідно з [16, 17], існування повної ідентифікованості можливо лише при виконані рангової умови для відповідної матриці ідентифікованості

$$Rank[W_0 W_B] = n, Rank W_c = n,$$

де W_0, W_B – системні матриці ідентифікованості та керованості, W_c – системная матрица спостережуваності $W_c = [D^T, A^T D^T, \dots, (A^T)^{n-1} D^T]$ (для системи з дискретним часом) або $W_c = [D^T, A^T D^T, \dots, (A^T)^{n-1} D^T]$.

У випадку, коли на динамічні властивості системи впливають відхилення параметрів і характеристик системи керування, то мова йде про вирішення проблем, пов'язаних з параметричною чутливістю.

Сама параметрична чутливість представляє собою залежність динамічних властивостей системи керування від будь-яких відхилень її параметрів і характеристик, від тих значень, які взяті у якості початкових.

Згідно з [18], знаходження розв'язків параметричної чутливості відбувається шляхом обчислення частинних похідних функції чутливості та побудови рівнянь чутливості. Також на розв'язок впливає вид функції C(t), яка може бути задана:

а) лінійно : $C(t) = C^0 + Ct$;

б) квадратично: $C(t) = C^0 + Ct^2$;

в) гармонійно: $C(t) = C^0 + Csin\omega t$;

де C, C^0, ω.

В тому випадку, коли позитивна система складається з однієї підсистеми, то маємо, що розв'язок параметричної чутливості подається у вигляді [18]:

$$X(t) = e^{\frac{(A-1)}{(1-B)}t}\left(\frac{1}{(1-B)}\int e^{\frac{-(A-1)}{(1-B)}t} C(t)dt + \bar{C}\right), \qquad (2.14)$$

для кожного параметра якого складається рівняння чутливості, які при урахуванні того, що всі всі функції чутливості є нульовими, набувають виду:

$$U_A^X(t) = e^{\frac{A-1}{1-B}t}\left[\frac{1}{1-B}\int X(t)e^{-\frac{A-1}{1-B}t}dt + c\right], \qquad (2.15)$$

$$U_B^X(t) = e^{\frac{A-1}{1-B}t}\left[\frac{A-1}{(1-B)^2}\int X(t)e^{-\frac{A-1}{1-B}t}dt + \frac{1}{(1-B)^2}\int C(t)e^{-\frac{A-1}{1-B}t}dy + C\right], \qquad (2.16)$$

$$U_{C^0}^X(t) = -\frac{1}{A-1} + ce^{\frac{A-1}{1-B}t}, \tag{2.17}$$

$$U_C^X(t) = e^{\frac{A-1}{1-B}t}\left[\frac{1}{1-B}\int F(t)e^{-\frac{A-1}{1-B}t}dt + c\right], \tag{2.18}$$

$$U_\omega^X(t) = e^{\frac{A-1}{1-B}t}\left[\frac{1}{1-B}\int G(t)e^{-\frac{A-1}{1-B}t}dt + c\right], \tag{2.19}$$

Стаціонарний же розв'язок рівнянь чутливості (2.15) — (2.19) при фіксації часу набувають виду [18]:

$$U_A^* = \frac{C^0}{(1-A)^2} \geq 0, \quad U_B^* = 0, \quad U_{C^0}^* = \frac{1}{1-A} \geq 0, \quad U_C^* = 0, \quad U_\omega^* = 0. \tag{2.20}$$

Проте, коли ми маємо систему, яка складається з n підсистем, то маємо, що розв'язок набуває наступного виду [18]:

$$X(t) = e^{\mathcal{A}(t-t_0)} + \int_{t_0}^t e^{\mathcal{A}(t-\tau)} f(\tau)d\tau,$$

Як і у випадку з однією підсистемою, для кожного параметру розв'язку складається рівняння чутливості, які при урахуванні, що вони нульові, набувають наступного виду [18]:

$$U_{a_{ij}}^X = \int_{t_0}^t e^{\mathcal{A}(t-\tau)} \frac{d\mathcal{A}}{da_{ij}} X(\tau)d\tau, \tag{2.21}$$

$$U_{b_{ij}}^X = \int_{t_0}^t e^{\mathcal{A}(t-\tau)} \left(\frac{d\mathcal{A}}{db_{ij}} X(\tau) + \frac{df(t)}{db_{ij}}\right) d\tau, \tag{2.22}$$

$$U_{c_i^0}^X = \int_{t_0}^t e^{\mathcal{A}(t-\tau)} \frac{df(\tau)}{dc_i^0} d\tau, \tag{2.23}$$

$$U_{c_i}^X = \int_{t_0}^t e^{\mathcal{A}(t-\tau)} \frac{df(\tau)}{dc_i} d\tau, \tag{2.24}$$

$$U_{\omega_i}^X = \int_{t_0}^t e^{\mathcal{A}(t-L\overline{o})} \frac{df(\tau)}{d\omega_i} d\tau. \tag{2.25}$$

Стаціонарний розв’язок рівнянь (2.21) – (2.25) при фіксації часу набуває виду [18]:

$$\begin{cases} U^*_{a_{ij}} = A^{-1}\dfrac{dA}{da_{ij}}(I-B)^{-1}C^0 \geq 0, \\ U^*_{b_{ij}} = 0, \\ U^*_{c_i^0} = A^{-1}\dfrac{(I-B)^{-1}C^0}{dc_i^0} \geq 0, \\ U^*_{c_i} = 0, \\ U^*_{\omega_i} = 0. \end{cases} .$$

Даний підрозділ був присвячений огляду основних властивостей позитивної динамічної системи та їх впливу на систему. В наступному підрозділі більш детально буде розкрита тема керування та регулювання в моделях позитивної системи при дискретному чи неперервному часі.

2.4 Керування та регулювання в дискретній та неперервній моделях позитивної динамічної системи

Даний підрозділ присвячений загальному огляду керування та регулювання в моделях позитивної системи при дискретному та неперервному часі.

У випадку виникнення необхідності в зміні досліджуваного процесу на більш прийнятний до певних вимог, використовується апарат теорії керування та регулювання.

Основним обмеженням керування є повний вектор станів, який може бути виміряний в будь-який момент на часовому інтервалі, а також використаний для зворотного зв’язку.

Виходячи з того, що система керування повинна бути асимптотично стійкою, то при не стійкому об’єкті, вона може бути стабілізована лише за допомогою замкненого керування.

Саме замкнене керування представляє собою вимірювання зворотного зв’язку керованої змінної на замкненому контурі, що дозволяє порівнювати бажану величину з дійсною та миттєво реагувати на будь які відхилення, які виникають між двома величинами.

Наступний підрозділ даної роботи буде присвячений можливим керуванням, які можуть бути використані в моделі позитивної динамічної системи при неперервному та дискретному часі, та можливим задачам, які можуть бути вирішенні в результаті використання даних керувань.

2.4.1 Визначення спектру можливих керувань

Даний підрозділ присвячений можливим керуванням, які можуть бути використані в моделі позитивної динамічної системи при неперервному та дискретному часі, та можливим задачам, які можуть бути вирішенні в результаті використання даних керувань.

Згідно з [15, 17, 20], класифікація керувань формується на основі компонентів системи, які безпосередньо на неї впливають. У випадку позитивної динамічної системи даними компонентами є матриці коефіцієнтів A та B, а також вектор керування C.

На основі вище наведеного, класифікація керувань для позитивних систему наступна [15, 17, 20]:

- технологічне керування або керування за допомогою вихідної матриці;
- інноваційне керування або керування за допомогою вихідної матриці;
- керування за допомогою вектору керування C;

Наведену класифікацію керувань позитивної динамічної системи можливо направити на розв’язок наступних задач [20]:

- направлення технологічного керування на покращення вихідних характеристик моделі;

- використання зворотнього зв’язку для отримання необхідного виду матриці;

- побудову моделі зі зворотнім зв’язком за допомогою завдання лінійного регулятора;

- використання інноваційного керування для прискорення процесу збіжності та покращення характеристик;

- використання програмного керування для досягнення бажаного стану системи.

Оскільки даний розділ присвячений виключно загальному опису можливих керувань та їх використання, в подальших розділах більш детально буде розглянуті дані керування та їх особливості.

2.4.2 Замкнені дискретна та неперервні моделі позитивної динамічної системи та їх використання

Даний підрозділ присвячений побудові замкненої моделі позитивної динамічної системи з неперервним чи дискретним часом, а також особливостям та властивостям даної моделі.

Для побудови замкненої моделі позитивної динамічної системи, використовується розімкнена позитивна динамічна модель виду [6 – 10]:

$$X_{t+1} = (I - B)^{-1}(A - B)X_t + (I - B)^{-1}C_t, X(0) = X_0, \quad (2.26)$$

$$\dot{X} = (I - B)^{-1}\big((A - I)X(t) + C(t)\big), X(0) = X_0. \quad (2.27)$$

Використовуючи зворотній зв'язок на рівняннях (2.26) та (2.27), представимо вектор керування C, як $C_t = C_t^0 - kX_t$ чи $C(t) = C_0(t) - kX(t)$, тоді замкнена модель позитивної динамічної системи матиме вигляд [11, 12]:

$$X_{t+1} = (I - B)^{-1}(A - k - B)X_t + (I - B)^{-1}C_t^0, \quad (2.28)$$

$$\dot{X}(t) = (I - B)^{-1}(A - k - I)X(t) + (I - B)^{-1}C_0(t). \quad (2.29)$$

Оскільки, замкнена модель є аналогічною до розімкненої моделі, то на її розв'язки накладаються аналогічні умови невід'ємності та асимптотичної стійкості.

Розв'язок замкненої моделі визначається станом моделі в теперішній момент часу величиною C_t^0 чи $C_0(t)$, та подається у вигляді [11, 12]:

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

$$X(t) = e^{At}X_0 + e^{At}\int_0^t e^{-At}BC_0(\tau)d\tau.$$

В томи випадку, коли вектор C_t^0 чи $C_0(t)$ є сталим, розв'язок рівняння замкненої моделі набуває наступного вигляду [11, 12]:

$$\Delta X_t = (I - B)^{-1}(A - k - I)X_t + (I - B)^{-1}C^0,$$

$$\dot{X}(t) = (I - B)^{-1}(A - k - I)X(t) + (I - B)^{-1}C^0.$$

Збереження асимптотичної стійкості розв'язків досягається шляхом визначення границі змінювання елементів матриці k.

В даному підрозділі було розглянуто формування замкненої моделі позитивної динамічної системи з неперервним та дискретним часом, в подальших підрозділах буть розглянуті решта керувань.

2.4.3 Здійснення керування через окремі матриці моделей позитивної системи

Даний підрозділ присвячений огляду керування більш відомого, як інноваційне, його особливостям та використанню.

Оскільки маємо, що через зв’язок зі стаціонарним розв'язком $X^* = (I - A)^{-1}C^0$, на вхідні характеристики X_t і $X(t)$ впливають елементи матриці $A = \left(a_{ij}\right)_{n\times n}$, існує необхідність в зміні керуючого впливу.

Так як, згідно з [20], існує можливість прискорити процес збіжності вихідних характеристик об’єкта до стаціонарного розв’язку, шляхом зміни елементів матриці $B = \left(b_{ij}\right)_{n\times n}$, то матриця B приймається у якості керуючого впливу, за тої умови, що елементи матриці задовольняють нерівності $0 \le b_{ij} \le a_{ij},\ i,j = \overline{1,n}$.

На основі вищенаведеного, покращення вихідних характеристик об'єкта, шляхом безпосереднього вибору елементів матриці B, визначається наступним [20, 21]:

- для збільшення значень кожного з елементів вектору X_t або $X(t)$ до відповідних значень елементів вектора X^*, елементи матриці B повинні наближуватися до елементів матриці A, за умовою, що $C_i^0 \neq 0\left(i = \overline{1,n}\right)$;

- для зменшення значень кожного з елементів вектору X_t або $X(t)$ елементи матриці B повинні наближуватися до нуля.

Виходячи з того, що замкнена позитивна модель є аналогом розімкненої моделі, то з урахуванням зміни матриці A, даній підхід застосовується і до неї.

Даний підрозділ був присвячений інноваційному керуванню, в наступному підрозділі буде детально розглянуто керування рухом позитивної системи більш відоме, як програмне керування.

2.4.4 Здійснення програмного керування рухом позитивної системи

Даний підрозділ присвячено огляду керування рухом позитивної системи більш відомого, як програмне керування. В даному розділі буде розглянуто особливості даного керування, а також його використання.

В тому випадку, коли задача керування полягає в знаходженні вектору-функції керування, який би реалізовував рух досліджуваної системи за призначеною траєкторією, бажана траєкторія руху системи буде задаватися в просторі станів наступним чином [20, 21]:

$$X_t^* = \varphi_t \geq 0,\ 0 < t < T,\ 0 < T < \infty, \tag{2.30}$$

$$X^*(t) = \varphi(t) \geq 0,\ 0 < t < T,\ 0 < T < \infty. \tag{2.31}$$

Виходячи з роботи [20, 21], з (2.30) та (2.31) випливає, що розв'язком задачі керування буде знаходження такого керування C, при якому рух замкненої системи точно проходив по траєкторії φ_t або $\varphi(t)$.

Оскільки, об'єктом керування виступає позитивна динамічна система, то виникає необхідність в накладані властивостей позитивності на бажані траєкторії руху системи φ_t та $\varphi(t)$. З цього випливатиме, що шуканий вектор C також буде позитивним.

Так як, поставлена задача має точне рішення, то шуканий вектор C матиме наступний вигляд та буде змінюватися відповідно до заданої векторної функції [20, 21]:

$$C_t = (I - B)\varphi_{t+1} + (A - B)\varphi_t, \tag{2.33}$$

$$C(t) = (I - B)\dot{\varphi}(t) + (A - B)\varphi(t). \tag{2.34}$$

При знайденому керуванні C процес в системі описується векторно-матричним різницевим рівнянням або диференціальним рівнянням відповідного виду

$$X_{t+1} = (I - B)^{-1}(A - B)X_t + f_t\ , \tag{2.35}$$

$$\dot{X}(t) = (I - B)^{-1}(A - I)X(t) + f(t). \tag{2.36}$$

де f_t або $f(t)$ визначається залежно від заданих траєкторій φ_t чи $\varphi(t)$ у вигляді

$$f_t = \varphi_{t+1} - (I - B)^{-1}(A - B)\varphi_t,$$

$$f(t) = \dot{\varphi}(t) - (I - B)^{-1}(A - I)\varphi(t).$$

Розв'язки рівнянь (2.35) та (2.36) за заданими початковими умовами $X(t_0) = X_0$ повинні точно співпадати із заданими траєкторіями руху системи $\varphi_t \geq 0$ або $\varphi(t) \geq 0$.

В даному підрозділі було розглянуто програмне керування, його особливості та використання. В наступному підрозділі буде детально розглянуто процес побудови алгоритму на основі усієї інформації наведеної в даному розділі

3 АЛГОРИТМІЗАЦІЯ ПРОЦЕСУ МОДЕЛЮВАННЯ ПОВЕДІНКИ, ВИЗНАЧЕННЯ, ДОСЛІДЖЕННЯ Й ПОКРАЩЕННЯ ВИХІДНИХ ХАРАКТЕРИСТИК КЕРОВАНОЇ ПОЗИТИВНОЇ ДИНАМІЧНОЇ СИСТЕМИ З n ПІДСИСТЕМАМИ

Даний розділ присвячений процесу побудови алгоритму моделювання позитивної динамічної системи. В даному розділі будуть відділені основні вхідні та вихідні дані системи, розроблен алгоритм моделювання позитивної динамічної системи та виділені особливості, які можуть виникнути при розробці алгоритму.

3.1 Вхідні та вихідні дані складної системи

Даний підрозділ присвячений виділенню основних вхідних та вихідних даних позитивної системи.

Оскільки об’єктом алгоритмізації виступає розімкнена динамічна позитивна модель з дискретним або неперервним часом, то на основі її формалізованого виду можна зробити висновки про елементи, які виступають її вхідними та вихідними даними.

Розімкнена динамічна позитивна модель з дискретним чи неперервним часом представлена в наступному вигляді [6 – 10]:

$$X_{t+1} = (I - B)^{-1}(A - B)X_t + (I - B)^{-1}C_t, X(0) = X_0, \quad (3.1)$$

$$\dot{X} = (I - B)^{-1}\big((A - I)X(t) + C(t)\big), X(0) = X_0. \quad (3.2)$$

Виходячи з виду моделей (3.1) та (3.2), в якості вхідних даних виступають наступні елементи:

- матриці коефіцієнтів A та B;

- вектор керування C;

- початкові стан системи X_0;

- розмірність матриці та векторів n.

Також в якості вихідних даних приймається тип системи (дискретна чи неперервна), кількість бажаних користувачем ітерації, крок ітерації, у випадку задання вектору керування гармонійно змінна ω та при необхідності побудови замкненої моделі матриця k.

В якості вихідних змінних виступатиме вектор отриманих результатів X.

В даному підрозділі буди виділені основні вхідні та вихідні дані позитивної системи. Наступний розділ буде присвячений побудові алгоритму моделювання позитивної системи.

3.2 Алгоритм визначення й покращення основних характеристик позитивної динамічної системи

Даний підрозділ присвячений процесу побудови алгоритму моделі позитивної системи з неперервним чи дискретним часом.

Алгоритм визначення та покращення основних характеристик позитивної динамічної системи складається з наступних етапів:

Першим етапом є введення вхідних даних, а саме, коефіцієнтів матриць A та B, елементів вектору C, початкової умови $X(0) = X_0$, тип моделі, кількість ітерації та крок ітерації (у випадку дискретної моделі крок за замовчуванням дорівнює 1).

На другому етапі відбувається перевірка даних на невід'ємність та продуктивність вхідних матриць, у разі непродуктивності матриці, відбувається перехід до керування, яке проводить поліпшення характеристик матриці до бажаного результату.

Третій етап представляє собою отримання вектору *C*, в залежності від заданого виду: лінійного, квадратичного або гармонійного.

На четвертому користувачеві надається можливість вибрати можливі керування, які направленні на прискорення процесу збіжності вихідних характеристик.

П’ятий етап представляє собою знаходження та отримання чисельних результатів.

Для більш наглядного зображення алгоритму у додатку А наведена абстрактна блок-схема.

В даному розділі було наведено алгоритм моделювання позитивної системи. Наступним розділ буде присвячений програмній реалізації алгоритму наведеного в даному розділі.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ МОДЕЛЮВАННЯ ПОВЕДІНКИ, ВИЗНАЧЕННЯ, ДОСЛІДЖЕННЯ Й ПОКРАЩЕННЯ ВИХІДНИХ ХАРАКТЕРИСТИК КЕРОВАНОЇ ПОЗИТИВНОЇ ДИНАМІЧНОЇ СИСТЕМИ З n ПІДСИСТЕМАМИ

Даний розділ присвячений реалізації алгоритму моделювання позитивної системи на основі алгоритму наведеного в розділі 3. В даному розділі буде обґрунтовано використання мови С++, для реалізації алгоритму, особливості розробленого програмного продукту, а також наведені результати роботи.

4.1 Обґрунтування вибору мови та середовища програмування для проведення автоматизації процесу моделювання поведінки, визначення, дослідження й покращення вихідних характеристик позитивної системи

Даний підрозділ присвячено обґрунтуванню використання мови С++, для реалізації алгоритму наведеного в розділі 3.

Враховуючи вище розглянуті особливості моделі та алгоритм реалізації визначення й покращення основних характеристик позитивної динамічної системи, було прийнято рішення для реалізації програмного продукту використати мову програмування С++.

Мова С++ представляє собою компілювану, статично типізовану мова програмування загального призначення. Дана мова програмування підтримує наступні парадигми: процедурне програмування, об'єктно-орієнтоване програмування та узагальнене програмування [22].

Мова має багату стандартну бібліотеку, яка включає поширені контейнери і алгоритми, введення-виведення, регулярні висловлювання,

підтримку багатопоточності та інші можливості. C++ поєднує властивості як високорівневих, так і низькорівневих мов [22].

В даному підрозділі було розглянуто обґрунтуванню вибору мови C++ для реалізації алгоритму наведеного в розділі 3. В наступному підрозділі буде описано реалізація програмного продукту.

4.1.1 Реалізація програмного продукту для кінцевого користувача

Даний підрозділ присвячено опису реалізації розробленого програмного продукту, на основі алгоритму наведеного в розділі 3, для кінцевого користувача.

Виходячи з описаного у підрозділі 3.2 алгоритму, був розроблений програмний продукт призначений для автоматизації процесу побудови, керування та розв’язку складної позитивної динамічної системи з n підсистемами.

На першому етапі її роботи користувачеві пропонується ввести кількість підсистем складної системи.

Оскільки в програмі, за замовченням, вибрана дискретна система, то на другому етапі вводу даних користувачеві пропонується змінити з дискретної на неперервну модель.

На третьому етапі вводу даних користувачеві пропонується ввести кількість бажаних їм ітерацій роботи програми. У випадку вибору неперервної системи користувачеві пропонується ввести крок ітерації, у випадку дискретної системи він за замовченням рівний одиниці.

Четвертий етап вводу даних представляє собою заповнення елементів матриць і векторів.

Наступний етап представляє собою перевірку невід’ємності та продуктивності матриці, у разі її не продуктивності користувачеві дається

можливість, використовуючи програмне керування, довести матрицю до відповідного вигляду. Також користувачеві пропонується задати вектор керування C за наступними законами: лінійним, квадратичним або гармонійним.

На останньому етапі користувачеві пропонується використати програмне керування для поліпшення або прискорення результатів, після цього відбувається розрахунок результатів та їх вивід на екран користувачеві.

В даному підрозділі було описана реалізація розробленого програмного продукту для кінцевого користувача, на основі алгоритму наведеного в розділі 3. В наступному підрозділі буде більш детально розглянута структура розробленого програмного продукту.

4.1.2 Структура розробленого програмного продукту

Даний розділ присвячено огляду розробленого програмного продукту з точки зору його структурної реалізації.

Розроблений програмний продукт використовує парадигму ООП, тому його структура представляє собою взаємозв’язок між класами.

Класом, який забезпечує створення та описує операції роботи з позитивною динамічною системою, виступає клас Matrixx.

Клас Matrixx представляє собою опис реалізації функціоналу для робот з матрицями, тобто визначає тип матриці, розмірність, а також необхідні методи для роботи з матрицями.

Основними методами, які забезпечують його роботу є:

У класі dynamicSystem визначаються також об’єкти типу «матриці», які описує клас matrix. Даний клас представляє собою реалізацію функціоналу для робот з матрицями, тобто визначає тип матриці, розмірність, а також

необхідні методи для роботи з матрицями. Наведемо деякі методи, присутні у класі matrix:

- setElement – метод, що дозволяє задавати окремий елемент матриці. Отримує у якості вхідні параметри число рядків, стовпців та елемент, яким треба замінити поточній елемент матриці;
- getElement – дозволяє отримати значення елеманта матриці за рядком та стовпцем;
- getInverse – функції отримання оберненої матриці методом Гауса-Жордана;
- getTranspose – метод реалізації транспонування матриці;
- getPow – метод піднесення матриці до степеня;
- setSingle – метод ініціалізації поточної матриці, шляхом заміни її на одиничну матрицю;
- isProductive – метод для перевірки матриці на продуктивність. Повертає значення true якщо матриця продуктивна, false – якщо не продуктивна.

Також клас Matrixx перевизначає основні арифметичні операції для роботи з матрицями, такі як:

- додавання, віднімання, множення матриць;
- додавання, віднімання, множення та ділення матриці на число;
- перевизначає оператора «дорівнює».

Даний підрозділ був присвячений огляду програмного продукту с точки зору його структурної реалізації. В наступному підрозділі будуть наведені результати роботи розробленого програмного продукту.

4.2 Приклади роботи розробленого програмного продукту

В піддaному розділі будуть розглянуті деякі приклади роботи розробленого програмного продукту для визначення й дослідження вихідних характеристик позитивної динамічної системи.

4.2.1 Визначення основних характеристик системи за математичними моделями системи без керування

Даний підрозділ присвячений наведенню результатів роботи розробленого програмного продукту без використання керування.

Проведемо обчислювальний експеримент за наступних початкових умовах та вхідних параметрах моделі:

$$n = 2\ A = \begin{pmatrix} 0.08 & 0.03 \\ 0.13 & 0.22 \end{pmatrix} B = \begin{pmatrix} 0.05 & 0.06 \\ 0.18 & 0.11 \end{pmatrix} C_0 = \begin{pmatrix} 0.6 \\ 0.8 \end{pmatrix} C_1 = \begin{pmatrix} 1.1 \\ 2 \end{pmatrix} X_0 = \begin{pmatrix} 6 \\ 8 \end{pmatrix}$$

Результати роботи програми для дискретної та неперервної системи з вхідними параметрами та початковими умовами наведені на рис. 4.1 та 4.2

Рисунок 4.1 – Результатів для дискретної моделі з вхідними параметрами та початковими умовами

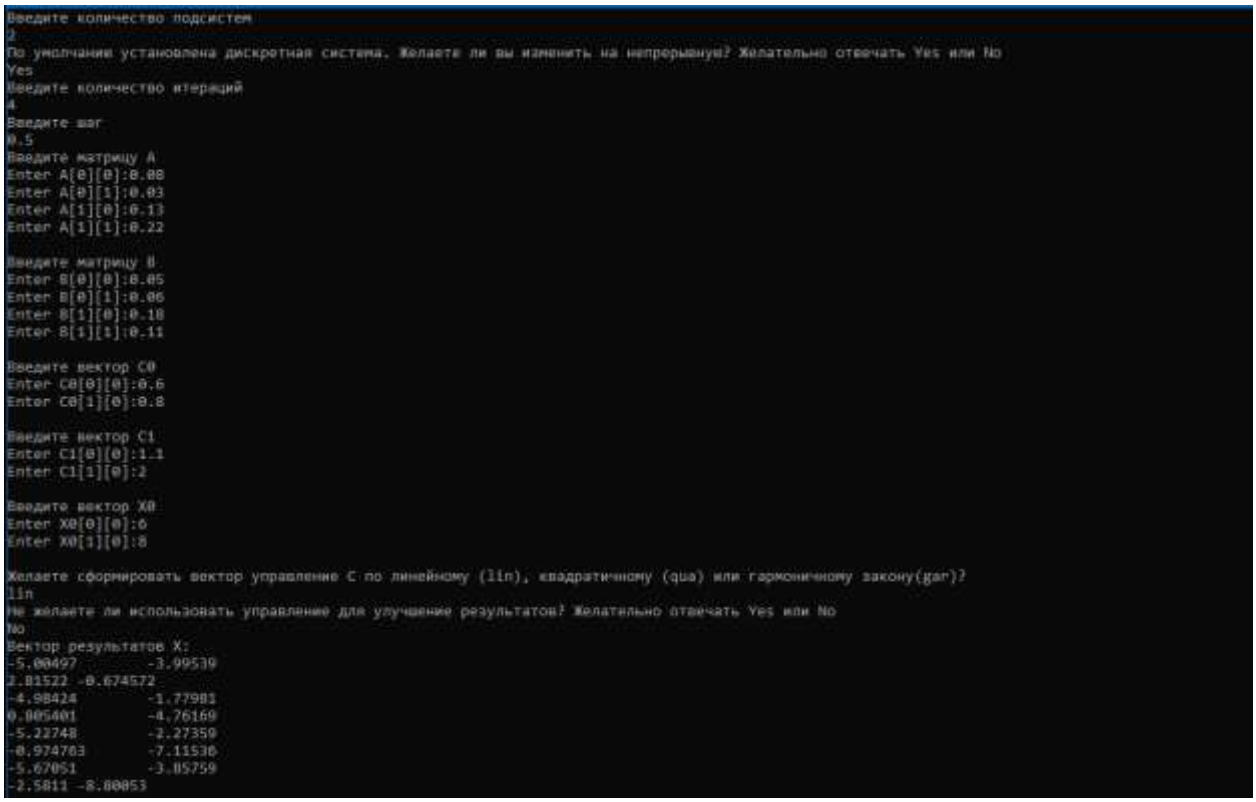

Рисунок 4.2 – Результатів для неперервної моделі з вхідними параметрами та початковими умовами

Даний підрозділ був присвячений результатам роботи розробленого програмного продукту без використання керування. В наступному підрозділі будуть наведені результати програмного продукту при використанні керування.

4.2.2 Визначення основних характеристик системи за математичними моделями системи із застосуванням різних видів керувань

Даний розділ присвячено результатам роботи розробленого програмного продукту з використанням керування.

Проведемо обчислювальний експеримент за наступних початкових умовах та вхідних параметрів моделі:

$$n = 2, A = \begin{pmatrix} 1.4 & 0.4 \\ 0.5 & 0.6 \end{pmatrix}, B = \begin{pmatrix} 0.05 & 0.06 \\ 0.18 & 0.11 \end{pmatrix}, C_0 = \begin{pmatrix} 0.6 \\ 0.8 \end{pmatrix}, C_1 = \begin{pmatrix} 1.1 \\ 2 \end{pmatrix} X_0 = \begin{pmatrix} 6 \\ 8 \end{pmatrix},$$

В даному випадку, матриця A непродуктивна, тому для того, щоб існували невід'ємні та асимптотично стійкі розв'язки, необхідно скорегувати значення матриці A, через ведення матриці k посилення зворотного зв'язку та переходу до замкненої моделі, де матриця k обрана наступним чином

$$k = \begin{pmatrix} 1 & 0{,}02 \\ 0{,}02 & 0{,}02 \end{pmatrix}$$

Результати корегування для неперервної системи представлені нижче на рис. 4.3

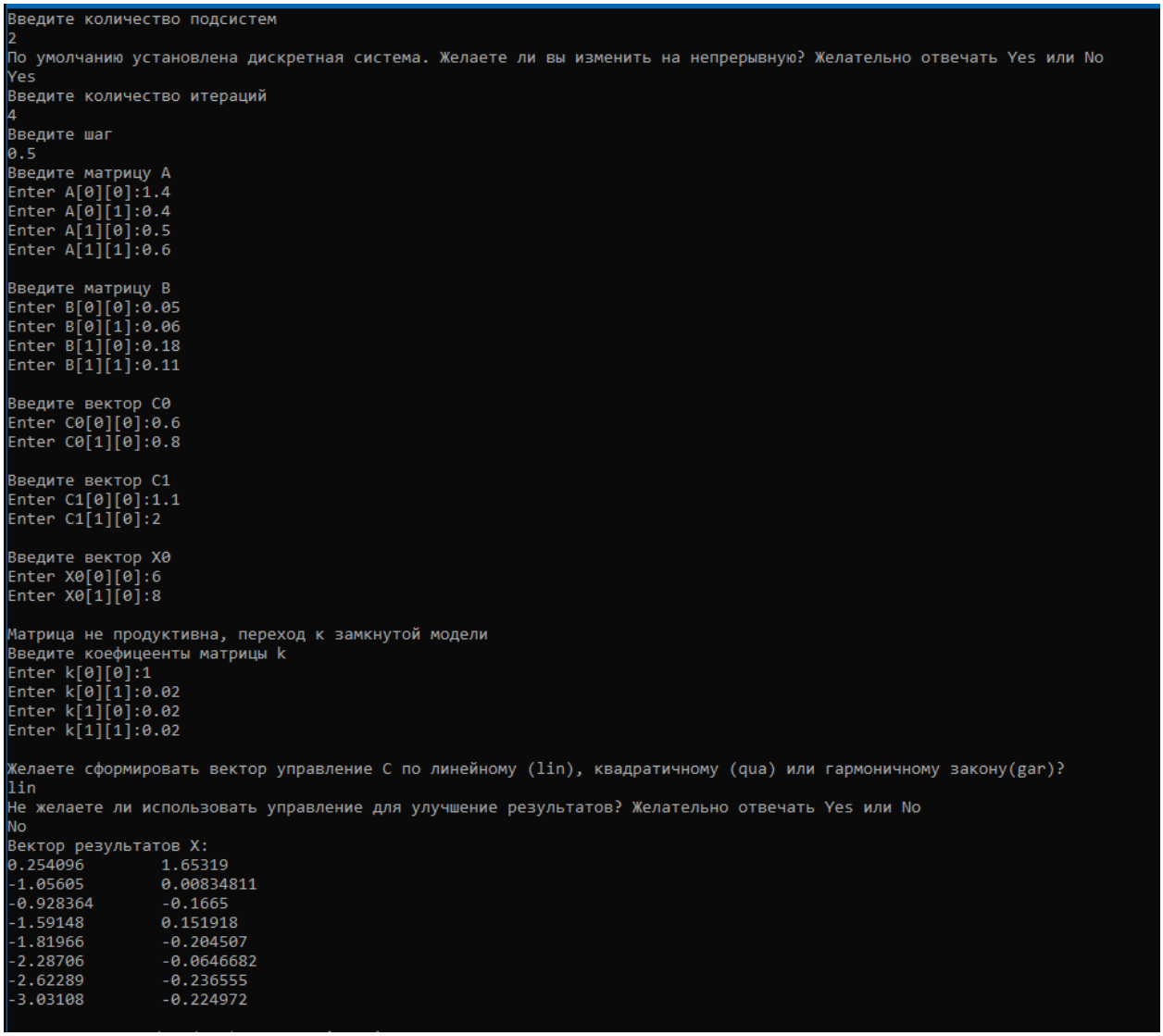

Рисунок 4.3 – Результатів для моделі з використанням матриці посилення зворотного зв’язку.

Припустимо, що матрицю A не можна змінювати, проте матриця B підлягає зміні, тоді для покращення вихідних значень X_t (чи $X(t)$ для непервного випадку) використаємо матрицю B, причому згідно з 2.4.3, для збільшення швидкості збіжності та збільшення значень розв’язкі матриця B повинна наближуватися до матриці , з цього матриця B має наступний вигляд:

$$B = \begin{pmatrix} 0{,}5 & 0{,}15 \\ 0{,}25 & 0{,}3 \end{pmatrix}$$

Результати покращення значень X_t наведені на рис. 4.4.

```
Вектор результатов X:
2.02889 4.25333
0.638054        3.05526
0.588289        3.24061
0.767996        4.1634
0.986968        5.36928
1.21266 6.67275
1.4395  8.00833
1.66654 9.35423
```

Рисунок 4.4 – Результати для дискретної моделі з використанням інноваційного керування за допомогою матриці B.

Результати покращення вектору $X(t)$ для неперервного аналогу наведені на рис 4.5.

```
Вектор результатов X:
-2.49556        -2.47333
-0.111422       -1.17589
-2.26295        -2.02723
-2.05593        -3.58261
-3.07536        -3.71547
-3.45704        -5.20077
-4.17034        -5.62742
-4.7112 -6.81068
```

Рисунок 4.5 – Результати для неперервної моделі з використанням інноваційного керування за допомогою матриці B.

Даний підрозділ був присвячений результатам роботи розробленого програмного продукту з використанням керування.

ВИСНОВКИ

Дана робота присвячена алгоритмізації процесу моделювання поведінки складної динамічної системи, яка характеризується невід’ємністю, та розробку на основі отриманого алгоритму програмного продукту. Під час виконання роботи були розглянуті основні етапи побудови позитивної системи, були розглянуті властивості системи, та їх взаємозв’язок, розкрите питання визначення необхідних видів керувань із множин можливих для даної системи та здійснення відповідного керування з урахуванням різних цілей дослідника.

Під час роботи був сформульований алгоритм визначення основних характеристик позитивної динамічної системи балансового типу, була надана його словесна та графічна форма, у вигляді блок-схеми.

В рамках проведеного дослідження моделі та створення на його основі алгоритму, була досягнуто мети дослідження, а саме був розроблений програмний продукт для визначення й дослідження вихідних характеристик позитивної динамічної системи балансового типу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Bokharaie V.S. Stability Analysis of Positive Systems with Applications to Epidemiology. Maynooth: National University of Ireland Maynooth, 2012. 178 p.

2. Ашманов С.А. Введение в математическую экономику. Москва.: Наука, 1984. 296 с.

3. Rami A. Solvability of static output-feedback stabilization for LTI positive systems. Systems & Control Letters №60, 2011. P.704-708

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

5. Гантмахер Ф.Р. Теория матриц. Москва.: Наука, 1988. — 552 с.

6. Давидов П.Л. Алгоритмізація процесу моделювання та керування складною позитивною динамічною системою балансового типу. Дев'ята регіональна наукова конференція молодих дослідників «Актуальні проблеми математики та інформатики» (Запоріжжя 28-29 квітня 2011). Збірка тез доповідей. Запоріжжя: ЗНУ, 2008. С. 71-74.

7. Леонтьева В.В. Наблюдаемость отдельного класса сложных динамических систем. Зб. наук. праць. Проблеми інформаційних технологій. Херсон: ХНТУ, 2015. - №2 - С. 89-95.

8. Леонтьева В.В. Разомкнутая дискретная математическая модель позитивных динамических систем балансового типа и ее анализ. Зб. наук. праць. Вісник ЗНУ. Запоріжжя: Запорізький національний університет, 2009. – №1. — С. 132-137.

9. Леонтьева В.В. Разомкнутая непрерывная математическая модель позитивных динамических систем балансового типа и ее анализ. Зб. наук. праць. Вісник ЗНУ. Запоріжжя: Запорізький національний університет, 2010. – №1. — С. 81-88.

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

11. Леонтьева В.В. Построение и анализ замкнутых дискретной и непрерывной математических моделей позитивных динамических систем балансового типа. Шоста регіональна наукова конференція молодих дослідників «Актуальні проблеми математики та інформатики» (Запоріжжя 24-25 квітня 2008). Збірка тез доповідей. Запоріжжя: ЗНУ, 2008. – С. 34-37.

12. Леонтьева В.В. Управляемость позитивной динамической системы балансового типа. Зб. наук. праць. Вісник ЗНУ. Запоріжжя: Запорізький національний університет, 2011. – №1. — С. 58-66.

13. Леонтьєва В.В. Керування у неперервній математичній моделі позитивної динамічної системи балансового типу. Вісник Херсонського національного технічного університету: Зб. наукових статей. Херсон: ХНТУ, 2009. - Вип. 2 (35). - С.273-278.

14. Квакернаак Х. Линейные оптимальные системы управления. Москва: Мир, 1977. 656 с.

15. Атанс М. Оптимальное управление / под ред. Ю.И. Топчеева; пер с англ. Москва: Машиностроение, 1968. 764 с.

16. Н. А. Балонин. Теореми ідентифікованості. Санкт-Петербург, 2010. 45 с.

17. Н. А. Балонин. Новий курс теорий управления движением. Санкт-Петербург: Санкт-Петербуржсикй государственный университет, 2000. 160 с.

18. Кокотович, П. В. Чувствительность систем автоматического управления. Автоматика и телемеханика, 1965. – Т. 26, Вып. 4. – С. 730–750.

19. Рубан, А. І. Ідентифікація та чутливість складних систем. Томск: Вид-во ТГУ, 1982. 304 с.

20. Андреев Ю.Н. Управление конечномерными линейными объектами. Москва: Наука, Главная редакция физико-математической литературы, 1976. 424 с.

21. Красовский Н.Н. Теория управления движением. Москва: Наука, 1968. 476 с.

22. C++ [Електронний ресурс] // Вільна енциклопедія ВікіпедіЯ. — Режим доступу: http://uk.wikipedia.org/wiki/C++. — Назва з екрана.

Додаток А

Блок схема алгоритму визначення й дослідження вихідних характеристик позитивної динамічної системи балансового типу

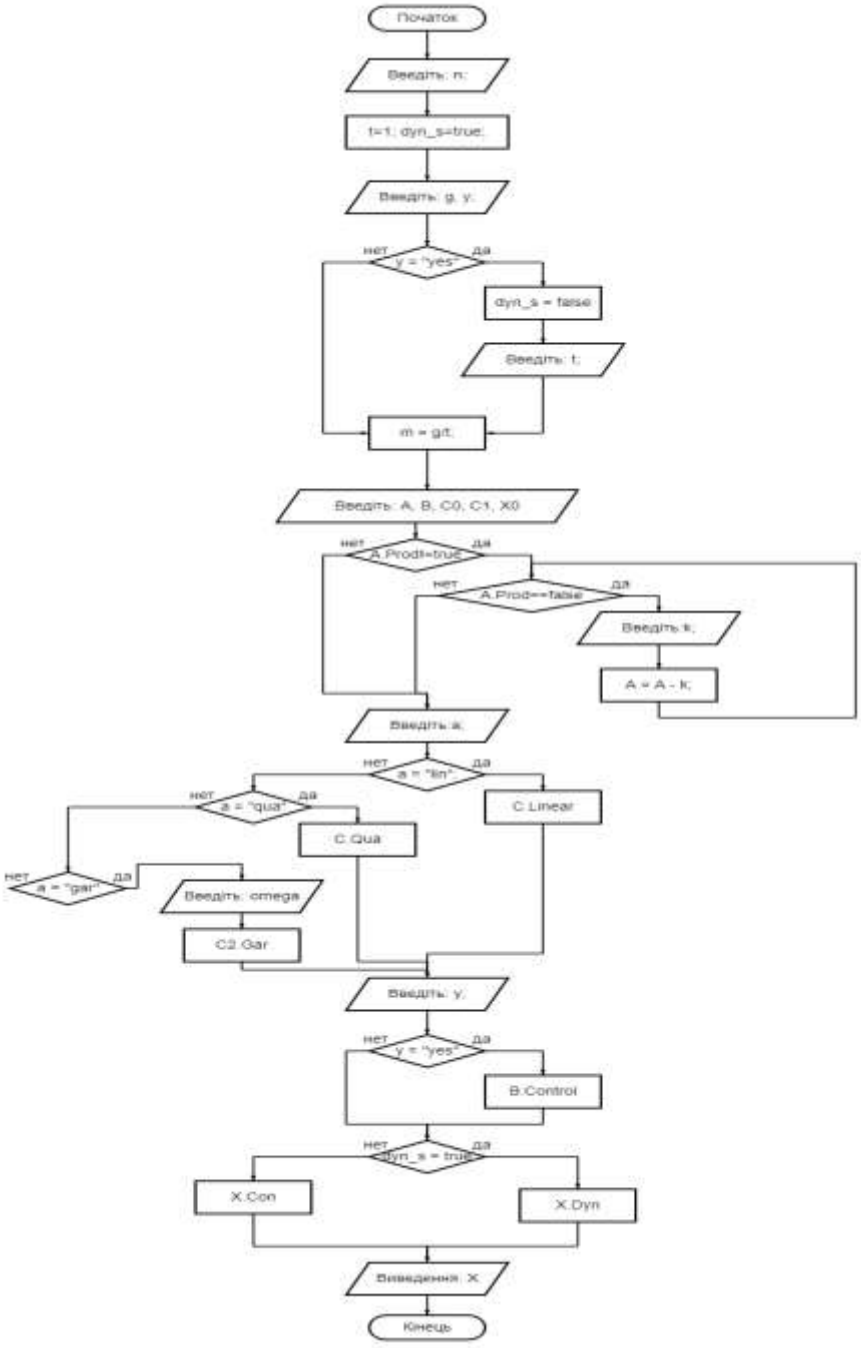

Рисунок А.1 – Блок схема алгоритму

Додаток Б

Лістинг вихідного коду програмного продукту для визначення й дослідження вихідних характеристик позитивної динамічної системи

Файл main.cpp:

```
#include <iostream>
#include "matrixx.h"

using namespace std;


int main()
{
   setlocale(LC_ALL, "Russian");

   int n;
   float t;
   t = 1;
   cout << "Введите количество подсистем" << endl;
   cin >> n;

   bool dyn_s = true;

   string y;
   cout << "По умолчанию установлена дискретная система. Желаете ли вы изменить на непрерывную? Желательно отвечать Yes или No" << endl;
   cin >> y;

   int g;
   cout << "Введите количество итераций" << endl;
   cin >> g;

   if (y == "Yes" || y == "yes")
   {
      dyn_s = false;
      cout << "Введите шаг" << endl;
      cin >> t;
```

```
}


int m;
m = g / t;

matrixx A(n, n);
matrixx B(n, n);

cout << "Введите матрицу A" << endl;
A.Creat("A");

cout << "Введите матрицу B" << endl;
B.Creat("B");

matrixx C0(n, 1);
matrixx C1(n, 1);

cout << "Введите вектор C0" << endl;
C0.Creat("C0");

cout << "Введите вектор C1" << endl;
C1.Creat("C1");

matrixx C2(m, n);



matrixx X0(n, 1);
cout << "Введите вектор X0" << endl;
X0.Creat("X0");

matrixx X(m, n);

if (A.isProductive() != true)
{
   printf("Матрица не продуктивна, переход к замкнутой модели");
   cout << endl;
   matrixx k(n, n);
   while (A.isProductive() == false)
   {
```

```
            printf("Введите коефицеенты матрицы k");
            cout << endl;
            k.Creat("k");
            A = A - k;
        }
    }
    cout << "Желаете сформировать вектор управление C по линейному (lin), квадратичному (qua) или гармоничному закону(gar)?" << endl;
    string a;
    cin >> a;

    if (a == "lin")
    {
        C2.C_Linear(C0, C1, n, m);
    }
    else if (a == "qua")
    {
        C2.C_Qua(C0, C1, n, m);
    }
    else if (a == "gar")
    {
        cout << "Введите значение омега";
        float omega;
        cin >> omega;
        C2.C_Gar(C0, C1, n, m, omega);
    }

    cout << "Не желаете ли использовать управление для улучшение результатов? Желательно отвечать Yes или No" << endl;
    string b;
    cin >> b;

    if (b == "Yes")
    {
        if(B.Control(A) == true)
        {
            string p;
            cout<<"Не желаете ли вы изменить матрицу B. Отвечать Yes или No"<<endl;
            cin>>p;
            if(p == "Yes")
```

```
            B.Creat("B");
        }
        else
        {
            cout<<"Матрица не соотвествует условию. Необходимо
изменить"<<endl;
            B.Creat("B");
        }
    }

    if (dyn_s == true)
    {
        X.Dyn_X(A, B, X0, C2, C0, m, n);
    }
    else X.Cont_X(A, B, C2, C0, X0, n, m);

    cout << "Вектор результатов X:" << endl;
    X.Print();

}
```

Файл matrixx.cpp:

```
#include "matrixx.h"
#include <math.h>
#include <stddef.h>
#include <iostream>

#define EPS 0.1e-15
#define Max(A,B) (((A)>(B))?(A):(B))
#define Min(A,B) (((A)<(B))?(A):(B))
#define Abs(A) (((A) > 0.)?(A): -(A))

// конструктор за замовчуванням
matrixx::matrixx()
{
        array = NULL;
        sizeRow = 0;
        sizeCol = 0;
}
```

```
// конструктор копіювання
matrixx::matrixx(matrixx& m)
{
	array = NULL;
	sizeRow = 0;
	sizeCol = 0;
	if (m.sizeRow > 0 && m.sizeCol > 0)
	{
		sizeRow = m.sizeRow;
		sizeCol = m.sizeCol;
		array = new float* [sizeRow];
		for (int ic = 0; ic < sizeRow; ic++) array[ic] = new float[sizeCol];
		for (int i = 0; i < sizeRow; i++)
			for (int j = 0; j < sizeCol; j++)
				array[i][j] = m.array[i][j];
	}
}

// конструктор із завданням розміру
matrixx::matrixx(int row, int col)
{
	array = NULL;
	sizeRow = 0;
	sizeCol = 0;
	if (row > 0 && col > 0)
	{
		sizeRow = row;
		sizeCol = col;
		array = new float* [sizeRow];
		for (int i = 0; i < sizeRow; i++) array[i] = new float[sizeCol];
	}
}

// деструктор
matrixx::~matrixx()
{
	if (array != NULL)
	{
		for (int i = 0; i < sizeRow; i++) if (array[i] != NULL)	delete[]
array[i];
		delete[] array;
```

```
        }
}

// перевантаження оператора '='
matrixx& matrixx::operator = (matrixx& m)
{
        if (this != &m)
        {
                if (sizeRow == 0 && sizeCol == 0)
                {
                        sizeRow = m.sizeRow;
                        sizeCol = m.sizeCol;
                        array = new float* [sizeRow];
                        for (int i = 0; i < sizeRow; i++) array[i] = new float[sizeCol];
                }
                if (sizeRow == m.sizeRow && sizeCol == m.sizeCol)
                {
                        for (int i = 0; i < sizeRow; i++)
                                for (int j = 0; j < sizeCol; j++)
                                        array[i][j] = m.array[i][j];
                }
        }
        return *this;
}

// перевантаження оператора '()'
float& matrixx::operator () (int row, int col)
{
        if (row > 0 && row <= sizeRow && col > 0 && col <= sizeCol)
        {
                return array[row - 1][col - 1];
        }
        else
                return array[0][0];
}

// перевантаження оператора '[]'
float& matrixx::operator [] (int k)
{
        if (sizeRow == 1 && k > 0 && k <= sizeCol)
                return array[0][k - 1];
```

```
	else
	{
		if (sizeCol == 1 && k > 0 && k <= sizeRow)
			return array[k - 1][0];
		else
			return array[0][0];

	}
}

// отримання кількості рядків
int matrixx::getSizeRow()
{
	return sizeRow;
}

// отримання кількості столбців
int matrixx::getSizeCol()
{
	return sizeCol;
}

// встановлення розміру
void matrixx::setSize(int row, int col)
{
	if (row == 0 && col == 0)
	{
		if (array != NULL)
		{
			for (int i = 0; i < sizeRow; i++) if (array[i] != NULL)	delete[]
array[i];
			delete[] array;
			array = NULL;
		}
		sizeRow = 0;
		sizeCol = 0;
		return;
	}
	if (row > 0 && col > 0)
	{
		if (array != NULL)
```

```
		{
			for (int i = 0; i < sizeRow; i++) if (array[i] != NULL) delete[] array[i];
			delete[] array;
		}
		sizeRow = row;
		sizeCol = col;
		array = new float* [sizeRow];
		for (int i = 0; i < sizeRow; i++) array[i] = new float[sizeCol];
	}
}

// зробити одиничною
matrixx& matrixx::setSingle(int n)
{
	if (n > 0)
	{
		setSize(n, n);
		init(0);
		for (int i = 0; i < n; i++)
		{
			array[i][i] = 1;
		}
	}
	return *this;
}

// ініціалізація числом
void matrixx::init(float c)
{
	for (int i = 0; i < sizeRow; i++)
	{
		for (int j = 0; j < sizeCol; j++)
		{
			array[i][j] = c;
		}
	}
}

// ініціаліз. статич. масивом
void matrixx::initStat(float* p, int size_r, int size_c)
```

```
{
	this->setSize(size_r, size_c);
	for (int i = 0; i < sizeRow; i++)
	{
		for (int j = 0; j < sizeCol; j++)
		{
			array[i][j] = p[i * size_c + j];
		}
	}
}

// ініціаліз. динаміч. масивом
void matrixx::initDynam(float** p, int size_r, int size_c)
{
	this->setSize(size_r, size_c);
	for (int i = 0; i < sizeRow; i++)
	{
		for (int j = 0; j < sizeCol; j++)
		{
			array[i][j] = p[i][j];
		}
	}
}

void matrixx::initmatrixx(matrixx& m, int size_r, int size_c)
{
	this->setSize(size_r, size_c);
	for (int i = 0; i < sizeRow; i++)
	{
		for (int j = 0; j < sizeCol; j++)
		{
			array[i][j] = m.array[i][j];
		}
	}
}

void matrixx::setElement(int row, int col, float k)
{
	array[row][col] = k;
}
```

```
float matrixx::getElement(int row, int col)
{
	return array[row][col];
}

// додавання матриць
matrixx& matrixx::operator + (matrixx& m)
{
	if (sizeRow == m.sizeRow && sizeCol == m.sizeCol)
		for (int i = 0; i < sizeRow; i++)
			for (int j = 0; j < sizeCol; j++)
				array[i][j] = array[i][j] + m.array[i][j];
	return *this;
}

// віднімання матриць
matrixx& matrixx::operator - (matrixx& m)
{
	if (sizeRow == m.sizeRow && sizeCol == m.sizeCol)
		for (int i = 0; i < sizeRow; i++)
			for (int j = 0; j < sizeCol; j++)
				array[i][j] = array[i][j] - m.array[i][j];
	return *this;
}

// множення матриць
matrixx& matrixx::operator * (matrixx& m)
{
	matrixx tmp(sizeRow, m.sizeCol);
	if (sizeCol == m.sizeRow)
		for (int i = 0; i < tmp.sizeRow; i++)
			for (int j = 0; j < tmp.sizeCol; j++)
			{
				float sum = 0;
				for (int k = 0; k < sizeCol; k++)
					sum += array[i][k] * m.array[k][j];
				tmp.array[i][j] = sum;
			}
	for (int ir = 0; ir < tmp.sizeRow; ir++)
		for (int jr = 0; jr < tmp.sizeCol; jr++)
			array[ir][jr] = tmp.array[ir][jr];
```

```
	return *this;
}

//операції зі скалярами
matrixx& matrixx::operator + (float c)
{
	if (sizeRow == sizeCol)
		for (int i = 0; i < sizeRow; i++)
			array[i][i] += c;
	return *this;
}

matrixx& matrixx::operator - (float c)
{
	if (sizeRow == sizeCol)
		for (int i = 0; i < sizeRow; i++)
			array[i][i] -= c;
	return *this;
}

matrixx& matrixx::operator * (float c)
{
	for (int i = 0; i < sizeRow; i++)
		for (int j = 0; j < sizeCol; j++)
			array[i][j] = array[i][j] * c;
	return *this;
}

matrixx& matrixx::operator / (float c)
{
	for (int i = 0; i < sizeRow; i++)
		for (int j = 0; j < sizeCol; j++)
			array[i][j] = array[i][j] / c;
	return *this;
}

// транспонування
matrixx matrixx::getTranspose()
{
	matrixx tmp(*this);
	for (int i = 0; i < sizeRow; i++)
```

```
            for (int j = 0; j < sizeCol; j++)
                array[j][i] = tmp.array[i][j];
    return *this;
}

// Отримати суму за рядком
float matrixx::getRowSum(int n)
{
    float sum = 0;
    for (int i = 0; i < sizeRow; i++)
    {
        sum += array[n][i];
    }
    return sum;
}

// зворотна матриця
matrixx matrixx::getInverse()
{
    matrixx tmp(*this);
    setSingle(sizeRow);

    int n;
    int j = 0;
    int m = 0;
    float prm = 0;
    int tmpSize = sizeRow * 4;
    float A[tmpSize][tmpSize], L[tmpSize][tmpSize];

    for (int it = 0; it < sizeRow; it++)
        for (int jt = 0; jt < sizeRow; jt++)
            A[it][jt] = tmp.array[it][jt];
    for (int ia = 0; ia < sizeRow; ia++)
        for (int ja = 0; ja < sizeRow; ja++)
            L[ia][ja] = array[ia][ja];
    do
    {
        m = j;
        for (int nj = j + 1; nj < sizeRow; nj++)
            if (fabs(A[m][j]) < fabs(A[nj][j])) m = nj;
```

```
                for (n = 0; n < sizeRow; n++)
                {
                        prm = A[m][n];
                        A[m][n] = A[j][n];
                        A[j][n] = prm;
                        prm = A[m][n];
                        A[m][n] = A[j][n];
                        A[j][n] = prm;
                }
                prm = A[j][j];
                for (int np = 0; np < sizeRow; np++)
                {
                        A[j][np] = float(A[j][np] / prm);
                        L[j][np] = float(L[j][np] / prm);
                }
                for (int in = 0; in <= sizeRow; in++)
                        if (in != j)
                        {
                                prm = A[in][j];
                                for (int n1 = j; n1 <= sizeRow; n1++) A[in][n1] = A[in][n1]
- float(prm * A[j][n1]);
                                for (int n2 = 0; n2 <= sizeRow; n2++) L[in][n2] = L[in][n2]
- float(prm * L[j][n2]);
                        }
                j++;
        } while (j < sizeRow);

        for (int ir = 0; ir < sizeRow; ir++)
                for (int jr = 0; jr < sizeRow; jr++)
                        array[ir][jr] = L[ir][jr];

        return *this;
}

// перевірка на невід'ємність
bool matrixx::isNotNegative()
{
        for (int i = 0; i < sizeRow; i++)
                for (int j = 0; j < sizeCol; j++)
                        if (array[i][j] < 0) return false;
        return true;
```

```
}

bool matrixx::isDiagonalLower()
{
	for (int i = 0; i < sizeRow; i++)
		if (array[i][i] > 1) return false;
	return true;
}

//перевірка на продуктивність
bool matrixx::isProductive()
{
	matrixx tmp(*this);
	matrixx E;
	E.setSingle(sizeRow);

	if (tmp.isNotNegative() == true)
	{
		if (tmp.isDiagonalLower() == true)
		{
			tmp = E - tmp;
			tmp.getInverse();
			for (int i = 0; i < tmp.sizeCol; i++)
				for (int j = 0; j < tmp.sizeRow; j++)
					if (tmp.array[i][j] < 0) return false;
			return true;
		}
		else return false;
	}
	else return false;
}

// Зведення в ступінь
matrixx matrixx::getPow(int p)
{
	matrixx tmp1(*this);
	matrixx tmp2(*this);
	matrixx tmp3(*this);

	if (p == -1)
		getInverse();
```

```
        if (p == 0)
                if (sizeRow = sizeCol) setSingle(sizeRow);
        if (p > 1)
        {
                matrixx mult(sizeRow, sizeCol);
                for (int i = 1; i < p; i++)
                {
                        if (i != 1) tmp1 = mult;
                        tmp1 = tmp1 * tmp2;
                        mult = tmp1;
                        tmp1 = tmp2;
                }
                for (int ir = 0; ir < sizeRow; ir++)
                        for (int jr = 0; jr < sizeRow; jr++)
                                array[ir][jr] = mult.array[ir][jr];
        }

        return *this;
}

matrixx matrixx::getExp(float dt)
{
        matrixx a(*this);
        matrixx ea(sizeRow, sizeCol);
        matrixx aa(sizeRow, sizeCol);
        matrixx wm(sizeRow, sizeCol);
        matrixx w(sizeRow, 1);
        double am, em, emi;
        int i, j, k, ii;
        em = 0;
        for (int ia = 0; ia < sizeRow; ia++)
        {
                for (int ja = 0; ja < sizeRow; ja++)
                {
                        ea.setElement(ia, ja, 0);
                        aa.setElement(ia, ja, 0);
                        wm.setElement(ia, ja, 0);
                        a.setElement(ia, ja, a.getElement(ia, ja) * dt);
                        am = Abs(a.getElement(ia, ja));
                        em = Max(am, em);
                }
```

```
            ea.setElement(ia, ia, 1);
            aa.setElement(ia, ia, 1);
            wm.setElement(ia, ia, 1);
        }
        emi = 1;
        ii = 0;
        while (emi > EPS)
        {
            ii++;
            if (ii >= 40) break;
            emi = 0;
            for (int jb = 0; jb < sizeRow; jb++)
            {
                for (int ib = 0; ib < sizeRow; ib++)
                    w.setElement(ib, 0, wm.getElement(ib, jb));
                for (int ic = 0; ic < sizeRow; ic++)
                {
                    wm.setElement(ic, jb, 0);
                    for (k = 0; k < sizeRow; k++)
                        wm.setElement(ic, jb, wm.getElement(ic, jb) +
(a.getElement(ic, k) * w.getElement(k, 0)));
                }
            }
            for (int id = 0; id < sizeRow; id++)
                for (int jd = 0; jd < sizeRow; jd++)
                {
                    wm.setElement(id, jd, wm.getElement(id, jd) / (double)ii);
                    ea.setElement(id, jd, ea.getElement(id, jd) +
wm.getElement(id, jd));
                    aa.setElement(id, jd, aa.getElement(id, jd) +
(wm.getElement(id, jd) / (double)(ii + 1)));
                    am = Abs(wm.getElement(id, jd));
                    emi = Max(am, emi);
                }

        }
        for (int it = 0; it < sizeRow; it++)
            for (int jt = 0; jt < sizeRow; jt++)
                array[it][jt] = ea.array[it][jt];
        return *this;
}
```

```
void matrixx::Print()
{
	for (int i = 0; i < sizeRow; i++)
	{
		for (int j = 0; j < sizeCol; j++)
			std::cout << array[i][j] << "\t";
		std::cout << "\n";
	}
}

void matrixx::Creat(const char* Object)
{
	for (int a = 0; a < sizeRow; a++)
	{
		for (int b = 0; b < sizeCol; b++)
		{
			std::cout << "Enter " << Object << "[" << a << "][" << b << "]:";
			std::cin >> array[a][b];
		}
	}
	std::cout << "\n";
}

matrixx matrixx::C_Linear(matrixx C0, matrixx C1, int n, int m)
{
	int j0 = 0;

	for (int i = 0; i < m; i++)
		for (int j = 0; j < n; j++)
			array[i][j] = C0.array[j][j0] + C1.array[j][j0] * i;

	return *this;
}

matrixx matrixx::C_Qua(matrixx C0, matrixx C1, int n, int m)
{
	int j0 = 0;

	for (int i = 0; i < m; i++)
		for (int j = 0; j < n; j++)
			array[i][j] = C0.array[j][j0] + C1.array[j][j0] * pow(i, 2);
```

```
	return *this;
}

matrixx matrixx::C_Gar(matrixx C0, matrixx C1, int n, int m, float qu)
{
	int j0 = 0;

	for (int i = 0; i < m; i++)
		for (int j = 0; j < n; j++)
			array[i][j] = C0.array[j][j0] + C1.array[j][j0] * sin(qu * i);

	return *this;
}

matrixx matrixx::Dyn_X(matrixx A, matrixx B, matrixx X0, matrixx C, matrixx C0,
int m, int n)
{
	matrixx E;
	matrixx X_1(n, 1);
	matrixx X_2(n, 1);
	E.setSingle(n);
	matrixx E_B = E - B;
	E_B.getInverse();
	matrixx E_A = E_B * A;
	matrixx X1(n, 1);
	X1.Mult(E_A, X0);
	matrixx X2(n, 1);
	X2.Mult(E_B, C0);
	X1 = X1 + X2;

	for (int i = 0; i < n; i++)
		array[0][i] = X1.array[i][0];

	for (int i = 1; i < m; i++)
	{
		X_1.init(0);
		X_2.init(0);

		for (int ix = 0; ix < n; ix++)
		{
```

```
                X_1.setElement(ix, 0, array[i - 1][ix]);
                X_2.setElement(ix, 0, C.array[i][ix]);

                X1.Mult(E_A, X_1);
                X2.Mult(E_B, X_2);
                X1 = X1 + X2;

                array[i][ix] = X1.getElement(ix, 0);

            }
        }
        return *this;
}
matrixx matrixx::Mult(matrixx A, matrixx B)
{


        for (int i = 0; i < A.sizeRow; i++)
            for (int j = 0; j < B.sizeCol; j++)
            {
                float sum = 0;
                for (int k = 0; k < A.sizeRow; k++)
                    sum += A.array[i][k] * B.array[k][j];
                array[i][j] = sum;
            }

        return *this;

}

matrixx matrixx::Cont_X(matrixx A, matrixx B, matrixx C, matrixx C0, matrixx X0,
int n, int m)
{
        matrixx E;
        E.setSingle(n);
        matrixx IB;
        IB = E - B;
        IB.getInverse();
        matrixx AI;
        AI = A - E;
        matrixx A1;
```

```
        A1 = IB * AI;
        matrixx Z1(n, 1);
        matrixx Z2(n, 1);

        Z1.Mult(AI, X0);
        Z2.Mult(IB, C0);

        matrixx Z3(n, 1);
        Z3 = Z1 + Z2;

        for (int i = 0; i < n; i++)
                array[0][i] = Z3.getElement(i, 0);

        matrixx Y1(n, 1);
        matrixx Y2(n, 1);

        for (int i = 1; i < m; i++)
        {
                Y1.init(0);
                Y2.init(0);

                for (int ix = 0; ix < n; ix++)
                {
                        Y1.setElement(ix, 0, array[i - 1][ix]);
                        Y2.setElement(ix, 0, C.array[i][ix]);

                        Z1.Mult(AI, Y1);
                        Z2.Mult(IB, Y2);

                        Z3 = Z1 + Z2;

                        array[i][ix] = Z3.getElement(ix, 0);
                }
        }
}

bool matrixx::Control(matrixx A)
{
   int m = sizeCol;
   int n = sizeRow;
   bool s;
```

```
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j++)
        if(array[i][j]<A.array[i][j] && 0<array[i][j])
          s = true;
        else s = false;

        return s;
}
```

Файл matrixx.h:

```
##ifndef MATRIXX_H
#define MATRIXX_H


class matrixx
{
private:
  float** array;
  int sizeRow;
  int sizeCol;
public:
  matrixx();
  matrixx(matrixx& m);
  matrixx(int row, int col);
  ~matrixx();
  void init(float c);
  void initStat(float* p, int size_r, int size_c);
  void initDynam(float** p, int size_r, int size_c);
  void initmatrixx(matrixx& m, int size_r, int size_c);
  void setElement(int row, int col, float k);
  float getElement(int row, int col);
  void setSize(int row, int col);
  int getSizeRow();
  int getSizeCol();
  float& operator () (int row, int col);
  float& operator [] (int k);
  matrixx& operator = (matrixx& m);
  matrixx& operator + (matrixx& m);
  matrixx& operator - (matrixx& m);
```

```
  matrixx& operator * (matrixx& m);
  matrixx& operator + (float c);
  matrixx& operator - (float c);
  matrixx& operator * (float c);
  matrixx& operator / (float c);
  float getRowSum(int n);
  matrixx  getExp(float dt);
  matrixx  getInverse();
  matrixx  getTranspose();
  matrixx  getPow(int p);
  matrixx& setSingle(int n);
  bool isNotNegative();
  bool isDiagonalLower();
  bool isProductive();
  void Print();
  void Creat(const char* Object);
  matrixx C_Linear(matrixx C0, matrixx C1, int n, int m);
  matrixx C_Qua(matrixx C0, matrixx C1, int n, int m);
  matrixx C_Gar(matrixx C0, matrixx C1, int n, int m, float qu);
  matrixx Dyn_X(matrixx A, matrixx B, matrixx X0, matrixx C, matrixx C0, int m,
int n);
  matrixx Mult(matrixx A, matrixx B);
  matrixx Cont_X(matrixx A, matrixx B, matrixx C, matrixx C0, matrixx X0, int n,
int m);
  bool Control(matrixx A);
};
#endif
```