НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Автоматики та управління в технічних системах

«На правах рукопису»
УДК ___519.685_____

«До захисту допущено»
Завідувач кафедри
_____ О.І., Ролік
«___»_____20_18_ р.

Магістерська дисертація

на здобуття ступеня бакалавра

зі спеціальності 151 автоматизація та комп'ютерно-інтегровані технології

на тему: «Метод динамічного програмування для задач цифрового оптимального керування»

Виконала:
студентка ІІ курсу, групи ІА-61м
Майер Ірина Володимирівна _____

Керівник:
доц., к.т.н., доц.
Писаренко А.В. _____

Рецензент:

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2018 року

РЕФЕРАТ

Дисертація освітньо-кваліфікаційного рівня “магістр” на тему «Метод динамічного програмування для задач цифрового оптимального керування»: 141 с., 22 рис., 24 табл., 9 додатки, 30 джерел.

Об'єкт дослідження - метод динамічного програмування в дискретній задачі оптимального керування.

Мета роботи – підвищення швидкодії методу динамічного програмування для задачі оптимального керування дискретними об’єктами.

Методами дослідження є метод рівномірного поділу, метод половинного поділу, метод золотого перетину.

Наукова новизна одержаних наукових результатів полягає у застосуванні математичних методів оптимізації в дискретній задачі динамічного програмування.

Виконано дослідження швидкості алгоритму розв’язання задачі оптимального керування дискретними об’єктами методом динамічного програмування шляхом застосування методів оптимізації: рівномірного поділу, половинного ділення, золотого перетину та покоординатного спуску для об’єктів першого та другого порядку.

Для дослідження роботи програмного засобу й обробки результатів дослідження використовувався пакет MATLAB і його бібліотека SIMULINK.

Результати даної роботи впроваджено в навчальний процес кафедри автоматики та управління в технічних процесах в дисципліну «Методи оптимізації в керуванні та управлінні».

Прогнозні припущення про розвиток дослідження – пошук оптимальних методів для задачі динамічного програмування, які знаходитимуть оптимальну траєкторію для будь-яких об’єктів за менший проміжок часу.

ДИНАМІЧНЕ ПРОГРАМУВАННЯ, ОПТИМАЛЬНЕ КЕРУВАННЯ, МЕТОД РІВНОМІРНОГО ПОДІЛУ, МЕТОД ПОЛОВИНОГО ПОДІЛУ, МЕТОД ЗОЛОТОГО ПЕРЕТИНУ, MATLAB/SIMULINK.

REPORT

Dissertation of the educational qualification level "Master" on the theme «Dynamic programming method for digital optimal control problems»: 141 p., 22 figures, 24 tables, 9 annexes, 30 sources.

The object of research is the method of dynamic programming in the discrete optimal control problem.

The purpose of the work is to increase the speed of the dynamic programming method for the optimal control of discrete objects.

Research methods are the method of uniform division, the method of half-division, the method of golden section.

Scientific novelty of the received scientific results consists in application of mathematical methods of optimization in a discrete problem of dynamic programming.

A study of the speed of the algorithm for solving the problem of optimal control of discrete objects by the method of dynamic programming is implemented by applying optimization methods: uniform division, half-division, golden section and coordinate descent for objects of the first and second order.

MATLAB and its library SIMULINK were used to study the work of the software and to process the results of the study.

The results of this work are implemented in the educational process of the Department of Automation and Control in technical processes in the discipline "Methods of optimization in management and management".

Foreseeable assumptions about the development of research - the search for optimal methods for the problem of dynamic programming, which will find the optimal trajectory for any objects over a shorter period of time.

DYNAMIC PROGRAMMING, OPTIMAL CONTROL, METHOD EQUILIBRIUM, METHOD OF A PARTIAL SUBSECTION, METHOD OF GOLD TRANSFER, MATLAB / SIMULINK.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ, ОДИНИЦЬ І ТЕРМІНІВ

МДП – метод динамічного програмування.

ДЗ – дискретна задача.

ЦО – цифровий об’єкт.

МЗП – метод золотого перетину.

МРП – метод рівномірного поділу.

МПП – метод половинного перетину.

МПС – метод покоординатного спуску.

ВСТУП

Актуальність теми.

Значне число задач, що виникають в суспільстві, пов’язані з процесами регулювання на основі прийняття рішень. Параметри цих процесів вибираються таким чином, щоб забезпечити екстремальне значення певного показника (вартість, маса, час роботи, тощо) при умові наявних обмежень на параметри процесу.

Сьогодні більшість систем керування проектують на основі різного роду цифрових обчислювальних пристроїв, що забезпечує високу точність, швидкодію, гнучкість при налаштуванні та збалансовану вартість. Розроблення та вдосконалення алгоритмів, що реалізують (зокрема) методи оптимального керування для побудови високошвидкісних цифрових систем автоматичного керування є актуальної задачею сьогодення.

Зв’язок роботи з науковими програмами, темами кафедри.

Результати даної роботи впроваджено в навчальний процес кафедри автоматики та управління в технічних процесах в дисципліну «Методи оптимізації в керуванні та управлінні»

Мета і задачі дослідження.

Мета – підвищення швидкодії методу динамічного програмування для задачі оптимального керування дискретними об’єктами. Для досягнення поставленої мети були сформульовані та розв’язані наступні задачі:

— огляд літератури та аналіз основних відомих результатів;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МРП для об’єкту першого порядку;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МПП для об’єкту першого порядку;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МЗП для об’єкту першого порядку;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МРП для об’єкту другого порядку;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МПП для об’єкту другого порядку;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МЗП для об’єкту другого порядку;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МПС для об’єкту другого порядку;

— розроблення моделі дискретної системи керування для перевірки отриманих результатів;

— дослідження швидкодії алгоритмів оптимізації в методі динамічного програмування дискретними об’єктами.

Об’єктами дослідження є метод динамічного програмування в дискретній задачі оптимального керування.

Предметом дослідження є алгоритми та методи оптимізації в дискретній задачі динамічного програмування.

Методами дослідження є метод рівномірного поділу, метод половинного поділу, метод золотого перетину.

Наукова новизна одержаних наукових результатів полягає у застосуванні математичних методів оптимізації в дискретній задачі динамічного програмування.

Практичне значення одержаних результатів полягає у створені програмних засобів на базі пакету MATLAB для реалізації методу динамічного програмування для дискретної задачі оптимального керування.

Апробація результатів роботи.

Основні результати роботи обговорювальсь на VI Міжнародній науково–практичній конференції з інформаційних систем та технологій — Summer InfoCom 2018, м. Київ, 20-21 травня 2018 р. (додаток А).

Публікації.

За результатами роботи роботи було опубліковано дві наукові праці:

— стаття у журналі «Інфокомунікації системи та технології » № 2(2), 2018 р. (додаток Б).

— тези доповіді на VI Міжнародній науково-практичній конференції з інформаційних систем та технологій – Summer InfoCom 2018, м. Київ, 20-21 травня 2018 р. (додаток А).

1 ЗАГАЛЬНІ ПОНЯТТЯ ПРО ОПТИМАЛЬНІ СИСТЕМИ

1.1 Загальні поняття про оптимальні системи

Оптимальна система — система автоматичного управління, що забезпечує найкраще (оптимальне) з деякою точки зору функціонування керованого об'єкта. Його характеристики і зовнішні впливи можуть змінюватися непередбачуваним чином, але, як правило, при певних обмеженнях. Оптимальне функціонування системи управління характеризується критерієм оптимального управління (критерієм оптимальності цільової функцією), який представляє собою величину, що визначає ефективність досягнення мети управління і залежить від зміни в часі або в просторі координат і параметрів системи. Критерієм оптимальності можуть бути різні технічні та економічні показники функціонування об'єкта: коефіцієн корисної дії, швидкодія, середнє або максимальне відхилення параметрів системи від заданих значень, собівартість продукції, окремі показники якості продукції або узагальнений показник якості і тощо.

Методи оптимізації діляться на прямі та ітераційні. Оптимізація полягає в знаходженні найкращого варіанту. Методи оптимізації застосовуються до пошуку розрахунку оптимальної технології, оптимальної геометричної конструкції, кращого часу для технологічних процесів і подібних завдань. Прикладом методу оптимізації є ітераційний метод Ньютона. Розрізняють: задачі безумовної оптимізації, задачі умовної оптимізації, задачі математичного програмування, задачі опуклого програмування, чисельні методи оптимізації [1].

Багатокритерійна оптимізація, або програмування — це процес одночасного оптимізації двох або більше конфліктуючих цільових функцій в заданій області визначення.

У задачах багатокритеріальної оптимізації абсолютно кращий варіант системи вибрати неможливо (за винятком окремих випадків), так як при переході від одного варіанта до іншого, як правило, поліпшуються значення одних критеріїв, але

погіршуються значення інших. Склад таких критеріїв називається суперечливим, і остаточно вибране рішення завжди буде компромісним.

Задачі багатокритеріальної оптимізації зустрічаються в багатьох областях науки, техніки та економіки.

Критерій оптимальності може ставитися як до перехідного, так і до сталого процесу, або і до того і до іного. Розрізняють регулярний і статистичний критерії оптимальності. Перший залежить від регулярних параметрів і від координат керованої і керуючої систем. Другий застосовується тоді, коли вхідні сигнали – випадкові функції або потрібно врахувати випадкові обурення, породжені окремими елементами системи. За математичного опису критерій оптимальності може бути або функцією кінцевого числа параметрів і координат керованого процесу, яка приймає екстремальне значення при оптимальному функціонуванні системи, або функціоналом від функції, яка описує закон управління; при цьому визначається такий вид цієї функції, при якому функціонал приймає екстремальне значення. Для розрахунку оптимальної системи користуються принципом максимуму Понтрягіна або теорією динамічного програмування.

Правильний вибір критеріїв грає істотну роль у виборі оптимального рішення. В теорії прийняття рішень не знайдено загального методу вибору критеріїв оптимальності. В основному керуються досвідом або рекомендаціями [2].

Оптимальне функціонування складних об'єктів досягається при використанні адаптивних систем управління, які мають здатність автоматично змінювати в процесі функціонування алгоритмів управління, свої характеристики або структуру для збереження незмінним критерію оптимальності при довільно змінюютьчих параметрах системи та умови її роботи. Тому в загальному випадку оптимальна система. складається з двох частин: постійної (незмінної), що включає об'єкт управління і деякі елементи керуючої системи, і змінної (змінної), що об'єднує інші елементи.

1.2 Задача оптимльного керування

Завдання оптимального керування відносяться до найбільш складним в теорії оптимізації. Найпростіша в цій теорії — задача знаходження екстремуму (мінімуму або максимуму) функції однієї змінної. її природним узагальненням є завдання знаходження екстремуму функції багатьох змінних. Вирішення цього завдання є кінцевий вектор, елемент векторного простору, тому завдання такого характеру називаються задачами конечномерной оптимізації.

У загальному випадку задачу керування не можна обмежувати тільки досягненням деякого значення вектора стану. Може виявитися, що в такому строгому досягненні цього стану і немає необхідності: важливо, щоб стан динамічної системи не вийшло з деякою області, що визначає різноманіття допустимих значень вектора стану. Природно, кожному заданому закону управління відповідає закон зміни координат вектора стану, тобто траєкторія "руху" керованого об'єкта в фазовому просторі. Найчастіше процес управління здійснюється з "обмеженими ресурсами", тобто закон управління не може бути довільним, а повинен вибиратися з деякої множини $\Omega(u)$.

Процес постановки конкретного завдання оптимального управління включає в себе ряд етапів:

— побудова систем математичних співвідношень, що описують керований об'єкт;

— визначення керуючого впливу;

— складання цільової функції і вказівка напряму її оптимізації;

— накладення обмежень на траєкторію зміни системи і керуючий вплив;

— визначення періоду оптимізації.

Залежно від виду даного явища і бажаного ступеня деталізації для побудови математичної моделі можуть використовуватися різні типи рівнянь: звичайні, диференціальні, рівняння з післядією, стохастичні рівняння, рівняння в приватних похідних тощо.

Узагальнений критерій якості представляє собою функціонал в інтегральному вигляді (1.2.1)

$$I = \int_{t_0}^{T} G(\boldsymbol{u}(t), x(t), \xi(t), t)dt \qquad (1.2.1)$$

де G() — визначає фізичний сенс критерія, що може залежати від станів об’єкту керування *x(t)*, керуючих впливів ***u(t)***, задавальних впливів системи керування $\xi(t)$.

Задача оптимального керування полягає у тому що : потрібно знайти таке керування ***u(t)*** в області допустимих керувань **Ω(u)**, при якому критерій якості (1.2.1) досягає екстремального значення, а об’єкт керування переводить з початкового стану $\boldsymbol{x(t_0)}$ в кінцевий ***x(T)***.

1.3 Основні методи розв’язання оптимальних задач

Розвиток теорії оптимального керування змусив багатьох учених займатися знаходженям методів розв’язання даної задачі. Можна виділити дві основні групи методів розв’язку: чисельні та матиматичні методи. Розв’язок за допомогою матиматичного методу можливий у малій кількості задач. Це пов’язано зі складністю об’єктів керування, невизначеністю їх структури, параметрів, зовнішніх впливів. В таких випадках основними методами виступають методи чисельної оптимізації. До математичних методів теорії оптимального керування належать варіаційне числення (розділ 1.3.1), динамічне програмування розділ 2 та принцип максимума Понтрягіна (розділ 1.3.2). Чисельні методи оптимізації спираються на метод математичного програмування, метод нечіткого програмування, робастне програмування тощо.

1.3.1 Варіаціне числення

Варіаційне обчислення – це галузь математики, пов'язана з проблемою пошуку функції для якого значення певного інтеграла є або найбільшим, або найменшим можливим. Це інтеграл технічно відомий як функціональний. Багато проблем такого

роду легко визначити, але їх рішення часто тягнуть за собою складні процедури диференціального обчислення. У свою чергу ці звичайно включають звичайні диференціальні рівняння (ДР), а також рівняння в приватних похідних (ПП).

Ізопериметрична проблема — це пошук серед усіх плоских фігур даного периметра одинщо охоплює найбільшу площу – був відомий грецьким математикам 2-го століття до нашої ери [3]. Термін було поширено в сучасній епосі, щоб означати будь-яку проблему в варіаційному обчисленні, в якій функція повинен бути максимально або мінімальним, за умови додаткового стану, що називається ізопериметричним обмеження, хоча це може не мати нічого спільного з периметрами. Наприклад, проблема пошуку твердого тіла заданого об'єму, що має найменшу поверхню, є ізопериметричною задачею, заданим об'ємом бути ізоперіметричним або допоміжним станом.

Методи варіаційного числення широко застосовуються в різних областях математики. Наприклад, в диференціальної геометрії з їх допомогою шукають геодезичні лінії і мінімальні поверхні. У фізиці варіаційний метод — один з найпотужніших інструментів отримання рівнянь руху (див. Наприклад Принцип найменшої дії), як для дискретних, так і для розподілених систем, в тому числі і для фізичних полів. Методи варіаційного обчислення застосовні і в статиці.

Найважливішими поняттями варіаційного обчислення є наступні:

— варіація (перша варіація),

— варіаційна похідна (перша варіаційна похідна),

— крім першої варіації і першої варіаційної похідної, розглядаються і варіації і варіаційні похідні другого і вищих порядків.

Ніяк не пов'язана з варіаційним обчисленням збігається за назвою варіація функції в аналізі.

Термін варіювання (варіювати) — застосовується у варіаційному численні для позначення знаходження варіації або варіаційної похідної (це аналог терміна диференціювання для випадку нескінченновимірного аргументу, що є предметом варіаційного обчислення) [4]. Також нерідко для стислості (особливо в додатках)

термін варіювання застосовується для позначення рішення варіаційної задачі, зводиться до знаходження варіаційної похідної та прирівнювання її нулю.

Варіаційна задача означає, як правило, знаходження функції (в рамках варіаційного обчислення — рівняння на функцію), що задовольняє умові стаціонарності деякого заданого функціонала, тобто такої функції, (нескінченно малі) обурення якої не викликають зміни функціоналу принаймні в першому порядку малості. Також варіаційної завданням називають тісно пов'язану з цим задачу знаходження функції (рівняння на функцію), на якій даний функціонал досягає локального екстремуму (багато в чому це завдання зводиться до першої, іноді практично повністю). Зазвичай при такому вживанні термінів мається на увазі, що завдання вирішується методами варіаційного обчислення.

Типовими прикладами варіаційної задачі є ізопериметричні завдання в геометрії та механіки; у фізиці — завдання знаходження рівнянь поля із заданого виду дії для цього поля.

Функціонал — математичне поняття, що виникло в варіаційному численні для позначення змінної величини, заданої на множині функцій. Залежить від вибору однієї або декількох функцій. Напр., Довжина дуги кривої, що з'єднує дві фіксовані точки, буде функціоналом, так як величина довжини дуги залежить від вибору функції, графік якої з'єднує ці точки.

1.3.2 Принцип максимума Потрягіна

Принцип максимума, сформульований в кінці 50-х, на початку 60-х років XX століття, і дотепер є одним з основних методів теорії оптимального керування. Принцип максимума Понтрягіна можна пояснити на прикладі задачі про максимальну швидкодію [5]. Нехай потрібна за мінімальний проміжок часу перевести точку з початкового положення H фазового простору окола точки K існує оптимальна фазова траєкторія і відповідне мінімальний час переходу в точку K. Навколо точки можна побудувати ізохрони — поверхню, яка являється геометричним місцем точок з однаковими мінімальним часом t_i переходу в цю точку. Оптимальна по швидкодії

траєкторія з точки Н в точку К в ідеальному випадку повинне спів пасти з нормалями до ізохронам. На практиці обмеження, накладені на координати об’єкта, не завжди дозволяють реалізувати ідеальну, оптимальну по швидкодії траєкторію. Тому оптимальна траєкторія буде та, яка максимально, наскільки дозволяють обмеження, близька до нормалей до ізохронам. Ця умова математично означає, що протягом усієї траєкторії скалярний добуток Н вектор швидкості V=$\frac{dx}{dt}$ руху зображення точки на вектор φ, зворотній (по напрямку0 градієнта часу переходу в кінцеву точку повинне бути максимальне(1.3.2.1):

$$H = \varphi V = \sum_{i=1}^{n} \varphi_i V_i = max \qquad (1.3.2.1)$$

де φ = - grad t_n – вектор, зворотній градієнт часу переходу; $\varphi_i V_i$ — координати векторів φ($\varphi_i, \dots, \varphi_n$) та V($V_i, \dots, V_n$). Так як скалярний добуток двох векторів рівний добутку їх моделей на косинус між ними, та умовою оптимальності являються максимум проекції вектору швидкості V на направлення φ. Дана умова оптимальності і є принцип максимума Понтрягіна. В даному прикладі умовою оптимальності є час. Та умова оптимальності (1.3.2.1) справедлива і для загального випадку, для будь-якого критерію.

Принцип максимуму особливо важливий в системах управління з максимальною швидкодією і мінімальною витратою енергії, де застосовуються управління релейного типу, які беруть крайні, а не проміжні значення на допустимому інтервалі управління.

2 МЕТОД ДИНАМІЧНОГО ПРОГРАМУВАННЯ

Динамічне програмування — один з найбільш потужних методів оптимізації. З завданнями прийняття раціональних рішень, вибору найкращих варіантів, оптимального управління мають справу фахівці різного профілю [6]. Серед методів оптимізації динамічне програмування займає особливе положення. Цей метод виключно привабливий завдяки простоті і ясності свого основного принципу — принципу оптимальності. Сфера застосування принципу оптимальності надзвичайно широка, коло завдань, до яких він може бути застосований, до теперішнього часу ще повністю не окреслено. Динамічне програмування з самого початку виступає як засіб практичного вирішення завдань оптимізації.

Виникнення цього методу пов'язують з ім'ям американського вченого Р. Беллмана, який на початку 50-х років XX століття застосував до ряду конкретних завдань прийом, названий згодом принципом оптимальності. Основною областю застосування останнього є багатокрокові процеси, так звані процеси, що розвиваються у часі, що дало підставу назвати новий метод оптимізації динамічним. Зазначенням на динамічність цей метод відрізнявся від лінійного та математичного програмування, вихідна постановка основних задач яких мала статичний характер.

Важко дати чітке визначення динамічному програмуванню. Зазначимо лише на три характерні його особливості. Крім принципу оптимальності, основного прийому дослідження, більшу роль в апаратах динамічного програмування грає ідея занурення конкретної задачі оптимізації в сімейство аналогічних завдань. Третьою його особливістю, що виділяє його серед інших методів оптимізації, є форма кінцевого результату. Застосування принципу оптимальності і принципу занурення в багатокрокових, дискретних процесах призводять до рекурентно-функціональним рівнянням щодо оптимального значення критерію якості. Отримані рівняння дозволяють послідовно виписати оптимальні управління для вихідного завдання. Виграш тут полягає в тому, що завдання обчислення управління для всього процесу

розбивається на ряд більш простих завдань обчислення управління для окремих етапів процесу.

Головним недоліком методу є, його складність катастрофічно зростає зі збільшенням розмірності задачі [7].

Останнім часом метод динамічного програмування використовується досить широко при синтезі технологічних схем поділу. Метод полягає в тому, що оптимальні схеми синтезують крок за кроком, починаючи з кінця. В даному випадку технологічна схема розглядається як багатостадійний процес поділу без зворотних масових і енергетичних потоків. На початковому етапі розглядаються колони, в яких діляться бінарні суміші, а далі трьох-, чотирьох компонентні і так далі, з урахуванням оптимального варіанту на попередньому етапі.

У кожному разі відшукується оптимальна по відношенню до прийнятого критерію технологічна схема поділу (будь-яка частина оптимального шляху є оптимальною). Це дозволяє відшукати оптимальний шлях поетапно, використовуючи на кожному етапі частини цього шляху, знайдені на попередніх етапах.

В кінцевому рахунку, можна обчислити значення критерію оптимальності для всіх схем і вибрати оптимальний варіант. Перевагою даного методу синтезу оптимального варіанту технологічної схеми поділу багатокомпонентних сумішей є строгий математичний підхід і зниження розмірності задачі, тобто скорочення розрахунків всіх можливих колон при поділі багатокомпонентної суміші.

Ідеєю динамічного програмування є:

— розбиття задачі на під задачі меншого розміру;

— знаходження оптимального рішення під задач рекурсивна, проробляючи такий же три кроковий алгоритм;

— використання отриманого рішення під задач для конструювання рішення вихідної задачі.

Підводячи підсумки вищесказаного можна сказати, що динамічне програмування користується такими властивостями:

— оптимальна підструктура;

— перекриваючі під задачі;

— можливість запам'ятовування рішення часто зустрічаючих під задач.

2.1 Безперервна задача

Безперевна задача звучить так — задано багатовимірний об’єкт керування рівняннями стану (2.1.1). Початковий стан об’єкту $\boldsymbol{x}(t_0)$, критерій оптимальності (2.1.2)

$$\dot{\boldsymbol{x}}(t) = \psi(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{2.1.1}$$

$$I = \int_{t_0}^{T} G\ (\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \tag{2.1.2}$$

Необхідно знайти такий вектор керуючих впливів $\mathbf{u}(t)$, при якому критерій (2.1.2) досягає мінімального значення.

Для розв’язання даної задачі, будемо вважати, що оптимальне керування $\mathbf{u}(t)$ відоме. Позначимо мінімальне значення критерію (2.1.2), що відповідає знайденому керуванню як $S(\boldsymbol{x}(t_0), t_0)$. Задамо моменти часу t та $t+\Delta t$, такі що t, $t + \Delta t \in [t_0; T]$, а Δt — мала величина. У відповідності до принципу оптимальності, частини траєкторій від точок $\boldsymbol{x}(t)$ та $\boldsymbol{x}(t+\Delta t)$ до кінцевої точки $\boldsymbol{x}(T)$ самі по собі є оптимальними. Нехай $S(\boldsymbol{x}(t),t)$ та $S(\boldsymbol{x}(t+\Delta t), t+\Delta t)$ — мінімальні значення критерію (2.1.2) на вказаних ділянках траєкторії (2.1.3), (2.1.4).

$$S(\boldsymbol{x}(t), t) = \min_{\boldsymbol{u}(t)} \int_{t}^{T} G\ (\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt, \tag{2.1.3}$$

$$S(\boldsymbol{x}(t + \Delta t), t + \Delta t) = \min_{\boldsymbol{u}(t)} \int_{t+\Delta t}^{T} G\ (\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt, \tag{2.1.4}$$

Зрозуміло, що мінімальне значення $S(\boldsymbol{x}(t), t)$ складається з мінімального значення критерію на ділянці $[t; t + \Delta t]$ і $[t + \Delta t; T]$. Тоді з урахуванням (2.1.3) та (2.1.4) можна записати (2.1.5)

$$S(\boldsymbol{x}(t),t)=\min_{\boldsymbol{u}(t)}\left\{\int_t^{t+\Delta T} G\ (\boldsymbol{x}(t),\boldsymbol{u}(t),t)dt+S(\boldsymbol{x}(t+\Delta t),t+\Delta t)\right\} \qquad (2.1.5)$$

Оскільки значення Δt мале, то функціонал можна представити у вигляді (2.1.6), де $o_1(t)$ — мала величина.

$$\int_t^{t+\Delta T} G\ (\boldsymbol{x}(t),\boldsymbol{u}(t),t)dt = G\ (\boldsymbol{x}(t),\boldsymbol{u}(t),t)dt+o_1(t) \qquad (2.1.6)$$

Розкладемо функцію $\boldsymbol{x}(t+\Delta t)$ у ряд Тейлора в околі точки t (2.1.7)

$$\boldsymbol{x}(t+\Delta t)=\boldsymbol{x}(t)+\frac{d\boldsymbol{x}}{dt}\Delta t+\boldsymbol{o}_2(t) \qquad (2.1.7)$$

де $\boldsymbol{o}_2(t)$ – наступні члени ряду.

Будемо вважати, що функція $S(\boldsymbol{x}(t),\ t)$ може бути диференційована за своїми аргументами. Тож якщо, ця функція невідома, таке припущення вносить деяку невизначеність у подальші міркування, але для безперервно диференційованих функцій $S(\boldsymbol{x}(t),\ t)$ наступні кроки повністю справедливі. Розкладемо функцію $S(\mathbf{x}(t+\Delta t),\ t+\Delta t)$ в ряд Тейлора в околі точки $(\mathbf{x}(t),t)$ (2.1.8). де $\boldsymbol{o}_3(t)$ — наступні члени ряду.

$$S(\boldsymbol{x}(t+\Delta t),t+\Delta t)=S(\boldsymbol{x}(t),t)+\frac{\partial S}{\partial \boldsymbol{x}}\Delta\boldsymbol{x}+\frac{\partial S}{\partial t}\Delta t+\boldsymbol{o}_3(t) \qquad (2.1.8)$$

Оскільки $\Delta\boldsymbol{x}=\boldsymbol{x}(t+\Delta t)-\boldsymbol{x}(t)$, тоді з (2.1.7) маємо

$$\Delta\boldsymbol{x}=\boldsymbol{x}(t+\Delta t)-\boldsymbol{x}(t)=\frac{d\boldsymbol{x}}{dt}\Delta t+\boldsymbol{o}_2(t) \qquad (2.1.9)$$

Підставимо отриманий вираз (2.1.9) у (2.1.8)

$$S(\boldsymbol{x}(t+\Delta t),t+\Delta t) = S(\boldsymbol{x}(t),t) + \frac{\partial S}{\partial \boldsymbol{x}}\left(\frac{d\boldsymbol{x}}{dt}\Delta t + \boldsymbol{o}_2(t)\right) + \frac{\partial S}{\partial t}\Delta t + \boldsymbol{o}_3(t) = S(\boldsymbol{x}(t),t) + \frac{\partial S}{\partial \boldsymbol{x}}\frac{d\boldsymbol{x}}{dt}\Delta t + \frac{\partial S}{\partial t}\Delta t + \boldsymbol{o}_4(t) \quad (2.1.10)$$

де $\boldsymbol{o}_4(t) = \frac{\partial S}{\partial \boldsymbol{x}}\boldsymbol{o}_2(t) + \boldsymbol{o}_3(t)$ — мала величина. Тепер отримане співвідношення та вираз (2.1.6) підставимо в (Р3.2.5)

$$S(\boldsymbol{x}(t),t) = \min_{\boldsymbol{u}(t)}\left\{G\ (\boldsymbol{x}(t),\boldsymbol{u}(t),t)\Delta t + S(\boldsymbol{x}(t),t) + \frac{\partial S}{\partial \boldsymbol{x}}\frac{d\boldsymbol{x}}{dt}\Delta t + \frac{\partial S}{\partial t}\Delta t + \boldsymbol{o}_5(t)\right\} \quad (2.1.11)$$

де $\boldsymbol{o}_5(t) = \boldsymbol{o}_1(t) + \boldsymbol{o}_4(t)$**.**

Оскільки *S*(x(*t*), *t*) — мінімальне значення функціоналу (при оптимальному керуванні $\overline{\boldsymbol{u}}(t)$), то воно не залежить від, керування та може бути винесено з-під знаку мінімум в (Р3.2.11) та скорочено з такою самою величиною у лівій частині.

$$O = \min_{\boldsymbol{u}(t)}\left\{G\ (\boldsymbol{x}(t),\boldsymbol{u}(t),t)\Delta t + \frac{\partial S}{\partial \boldsymbol{x}}\frac{d\boldsymbol{x}}{dt}\Delta t + \frac{\partial S}{\partial t}\Delta t + \boldsymbol{o}_5(t)\right\} \quad (2.1.12)$$

Поділимо останній вираз на Δt та спрямуємо його до нуля. Оскільки порядок малості величини $\boldsymbol{o}_5$ (t) більший за порядок малості величини Δt , то границя $\lim_{\Delta t\to 0}\frac{\mathbf{0}_5}{\Delta t} = 0$. Маємо (2.1.13)

$$O = \min_{\boldsymbol{u}(t)}\left\{G\ (\boldsymbol{x}(t),\boldsymbol{u}(t),t) + \frac{\partial S}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}}(t) + \frac{\partial S}{\partial t}\right\} \quad (2.1.13)$$

Часткова похідна $\frac{\partial S}{\partial t}$ також не залежить від керування $\mathbf{u}(t)$ та може бути винесена з правої частини рівняння (2.1.13) до лівої. Величину $\boldsymbol{x}(t)$ з урахуванням рівнянь стану (2.1.1) замінимо на їх праву частину. Тоді остаточно (2.1.14):

$$-\frac{\partial S(\boldsymbol{x}(t),t)}{\partial t} = \min_{\boldsymbol{u}(t)}\left\{G\ (\boldsymbol{x}(t),\boldsymbol{u}(t),t) + \frac{\partial S(\boldsymbol{x}(t),t)}{\partial \boldsymbol{x}}\psi(\boldsymbol{x}(t),\boldsymbol{u}(t),t)\right\} \quad (2.1.14)$$

Останнє рівняння має назву функціонального рівняння Беллмана, або просто рівняння Беллмана у векторній формі, а функція S(x(t), t) — функції Беллмана.

Скалярний запис рівняння Беллмана (2.1.15):

$$-\frac{\partial S(\boldsymbol{x}(t),t)}{\partial t} = \min_{\boldsymbol{u}(t)} \left\{ G\ (\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \sum_{i=1}^{n} \frac{\partial S(\boldsymbol{x}(t),t)}{\partial x_i} \psi_i(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \right\} \qquad (2.1.15)$$

Рівняння Беллмана представляє собою необхідну умову мінімум. Якщо функція $S(x(t), t)$ диференційована за своїми змінними $x(t)$, t , то вона задовольняє рівнянню Беллмана.

2.2 Дискретна задача

Дискретна задача динамічне програмування визначає оптимальне рішення в багатомірній задачі шляхом її декомпозиції на етапи, кожний з яких представляє підзадачу щодо однієї змінної [8]. Перевага такого підходу полягає в тому, що замість цього багатовимірна задача перетворюється в багато одновимірних, в результаті на кожному етапі вирішуються одномірні оптимальні задачі.

При його реалізації одночасний вибір значень великого числа змінних вирішуваної екстремальної задачі заміняється почерговим визначенням кожної з них залежно від можливих обставин. Процедуру вибору значень змінних будем трактовати як багатоетапний процес управління деякою системою. Далі объект називаємо дискретною керованою системою, якщо множина його можливих станів і управління в ньому здійснюються в дискретному часі, покроково; кожне керування укладаються в застосуванні одного з кінцевого числа можливих впливів; результат будь якої дії є зміна стану системи [9]. Кожна послідовність управління, переводящая система від початкового в один з кінцевих станів, фактично визначають деяку повну траекторію руху системи. Потрібно знайти повну траєкторію, оптимальну по значенню.

В задачі дискретного програмування задано об’єкт керування, представлений дискретною моделлю простору стану (2.2.1). Задано початковий стан $\boldsymbol{x}[0]$ та критерій оптимальності (2.2.2). Важливо, щоб керування належати області допустимих керувань Ω(u).

$$\boldsymbol{x}[\mathrm{k+1}] = \boldsymbol{\varphi}(\boldsymbol{x}[k], u[k]) \tag{2.2.1}$$

$$I = \sum_{k=0}^{N} G(\boldsymbol{x}[k], \boldsymbol{u}[\boldsymbol{k}]) \tag{2.2.2}$$

Потрібно знайти послідовність керувань $u[k]$, $k = 0,1,\ldots, N$ таку, щоб критерій (2.2.2) досягав мінімального значення.

Почнемо розв’язувати задачу з кінця. Будемо вважати, що усі керування $u[0], u[1],\ldots,u[N-1]$ відомі. Тоді за рівнянням (2.2.1) та початковим станом об’єкта $\boldsymbol{x}[0]$ знайдемо усі інші значення стану об’єкту $\boldsymbol{x}[1],\boldsymbol{x}[2],\ldots, \boldsymbol{x}[N]$. Від єдиного невідомого керування $u[N]$ залежить останній доданок у критерії (2.2.2).

$$I[N] = G(\boldsymbol{x}[N], u[N]) \tag{2.2.3}$$

Мінімум даного критерію за $u[N]$ можна знайти за будь- яким методом мінімізації одновимірної функції з урахуванням обмежень Ω(u) . Існує безміч методів мінімізації серед таких методів є методи: рівномірного поділу, половинного ділення, золотого перетину (їх розглянемо в розділі 3).

В попередньої умови зрозуміло, що значення керування буде залежати від стану об’єкта $\boldsymbol{x}[N]$.

$$\bar{u}[N] = \bar{u}[N](\boldsymbol{x}[N]) \tag{2.2.4}$$

Далі потрібно обчислити значення $I[N]$ з урахуванням (2.2.4). Воно є мінімальним, оскільки керування $u[N]$ знайдене з умови мінімуму критерію.

Мінімальне значення критерію при мінімальному керуванні $u\ [N]$ будемо позначати через $S[N]$.

$$S[N] = S[N](\boldsymbol{x}[N]) = \min_{u[N]\in\Omega(u)} I[\boldsymbol{N}] \qquad (2.2.5)$$

Для пошуку мінімуму (2.2.3) потрібно розбити область допустимих керувань Ω(u) на M значень, так щоб було достаньо для збереження точності обчислень, керування матимуть вигляд – $u[N]_1$, $u[N]_2$,..., $u[N]_M$, данний спосіб носить назву метод рівномірного поділу тобто. Область допустимих станів $Q(\mathbf{x})$, до якої належить $\boldsymbol{x}[N]$ також розбивається на Z значень, тим же методом мінімізації, тоді отримаємо послідовність $\boldsymbol{x}[\mathrm{N}]_1$, $\boldsymbol{x}[\mathrm{N}]_2$,…, $\boldsymbol{x}[\mathrm{N}]_\mathrm{Z}$.

Для пошуку значень керувань у уформі (2.2.4) підставимо $\boldsymbol{x}[\mathrm{N}]_1$ у (2.2.3) та перебираємо усі значення $u[\mathrm{N}]_1$, $u[\mathrm{N}]_2$,…, $u[\mathrm{N}]_\mathrm{M}$. Отримаємо набір з $I(u[N]_1, \boldsymbol{x}[N]_1)$, $I(u[N]_2, \boldsymbol{x}[N]_1)$,…, $I(u[N]_\mathrm{M}, \boldsymbol{x}[N]_1)$ та вибираємо з нього мінімальне значення, отримане значення запишемо в $S[N](\boldsymbol{x}[N]_1)$. Керування, щому позначається $\bar{u}[N](\boldsymbol{x}[\mathrm{N}]_1)$. Повторуємо попередні кроки для значень $\boldsymbol{x}[\mathrm{N}]_2$, $\boldsymbol{x}[\mathrm{N}]_3$,…, $\boldsymbol{x}[\mathrm{N}]_\mathrm{Z}$. В результаті будемо мати набір можливих значень станів $\boldsymbol{x}[N]_1$, $\boldsymbol{x}[N]_2$,…, $\boldsymbol{x}[N]_Z$, та отримаємо керування $\bar{u}[N]](\boldsymbol{x}[N]_1)$, $\bar{u}[N](\mathbf{x}[N]_2)$,…, $\bar{u}[N](\boldsymbol{x}[N]_\mathrm{Z})$ відповідно до отриманих станів та мінімальних значень $I[N]$ записаних у змінних $S[N](\boldsymbol{x}[N]_1)$, $S[N](\boldsymbol{x}[N]_2)$,…, $S[N](\boldsymbol{x}[N]_\mathrm{Z})$.

На наступному етапі, аналогічно припускаємо, що усі керування $u[0]$,$u[1]$,…, $u[\mathrm{N}-2]$ знайдені. Обчислюємо відповідні їм значення $\boldsymbol{x}[1]$, $\boldsymbol{x}[2]$,…,$\boldsymbol{x}[\mathrm{N}-1]$. Необхідно знайти оптимальні керування $u[\mathrm{N}-1]$ та $u[\mathrm{N}]$. Принцип оптимальності говорить, що вони залежатимуть від поточного стану об’єкту x[N−1] та мінімуму двох останніх доданків функціоналу (2.2.2):

$$I[N-1] = G(\boldsymbol{x}[N-1], u[N-1]) + G(\boldsymbol{x}[N], u[N]) \to \min_{u[N-1],u[N]\in\Omega(u)} 1 \qquad (2.2.6)$$

Та від $u[\mathrm{N}]$ залежить тільки другий доданок в (2.2.6), мінімізація його уже зроблена на попередньому кроці. Коли дістанемо з пам’яті оптимальне керування

u[N] та відповідне йому мінімальне значення критерію S[N]. Тому вираз (2.2.6) матиме вигляд (2.2.7):

$$I[N-1] = G(\boldsymbol{x}[N-1], u[N-1]) + S[N](\boldsymbol{x}[N]) \rightarrow \min_{u[N-1]\in\Omega(u)} \quad (2.2.7)$$

Врахуємо вираз (2.2.1) та перепишемо в (2.2.8):

$$\boldsymbol{x}[N] = \boldsymbol{\varphi}(x[N-1], u[N-1]) \quad (2.2.8)$$

Тоді критерій (2.2.7) матиме вигляд (2.2.9)

$$I[N-1] = G(\boldsymbol{x}[N-1], u[N-1]) + S[N](\boldsymbol{\varphi}(\boldsymbol{x}[N-1], u[N-1])) \rightarrow \min_{u[N-1]\in\Omega(u)} \quad (2.2.9)$$

Далі задача мінімізації знову залежить від однієї змінної $u[N-1]$, результатом розв'язання якої є знаходження $u[N-1] = u[N](\boldsymbol{x}[N-1])$ та відповідної величини $S[N-1](\boldsymbol{x}[N-1])$.

Знову область допустимих станів в $Q(\mathbf{x})$ розбиваємо на Z значень, які можуть приймати керування $\boldsymbol{x}[N-1]$. Аналогічно розбиваємо область значень $u[N-1]$ на M точок. Обчислюємо можливі значення $\boldsymbol{x}[N]$ по (2.2.8) при $\boldsymbol{x}[N-1] = \boldsymbol{x}[N-1]_1$ та усіх $u[N-1]_1$, $u[N-1]_2$,…, $u[N-1]_\text{M}$. Тобто маємо набір $\boldsymbol{x}[N]_1$, $\boldsymbol{x}[N]_2$,…, $\boldsymbol{x}[N]_\text{M}$. Отримані значення округляємо до найближчих значень $\boldsymbol{x}[N]$, що обчислені на попередньому етапі та дістаємо з пам'яті відповідну їм величину $S[N](\boldsymbol{x}[N])$. Формуємо на основі цих даних набір

$$G\big(\boldsymbol{x}[N-1]_1, u[N-1]_j\big) + S[N]\big(\boldsymbol{x}[N]_j\big), j = \overline{1,\text{M}} \quad (2.2.10)$$

Знаходимо мінімальне значення в (2.2.10) та записуємо в $S[N-1](\boldsymbol{x}[N-1]_1)$, а відповідне йому керування в $\bar{u}[N-1](\mathbf{x}[N-1]_1)$. Далі перебираємо усі решту значень $\boldsymbol{x}[N-1]_2$, $\boldsymbol{x}[N-1]_3$, …, $\boldsymbol{x}[N-1]_\text{Z}$ і аналогічно знаходимо $\bar{u}[N-1](\boldsymbol{x}[N-1]_2)$,

$\bar{u}[N-1](\boldsymbol{x}[N-1]_3)$,…, $\bar{u}[N-1](\boldsymbol{x}[N-1]_Z)$ та мінімальні значення $S[N-1](\boldsymbol{x}[N-1]_2)$, $S[N-1](\boldsymbol{x}[N-1]_3)$,…, $S[N-1](\boldsymbol{x}[N-1]_z)$, що відповідають набору $\boldsymbol{x}[N-1]_2$, $\boldsymbol{x}[N-1]_3$,…, $\boldsymbol{x}[N-1]_Z$. Далі пропускаємо три кроки та повторюємо описану послідовність дій, знаходячи $\bar{u}[N-2]$ та $S[N-2]$. Решта кроків аналогічна попереднім. На останньому етапі задачі з умови мінімуму

$$I[0] = G(\boldsymbol{x}[0], u[0]) + S[1](\boldsymbol{\varphi}(\boldsymbol{x}[0], u[0])) \rightarrow \min_{u[0]\in\Omega(u)} \qquad (2.2.11)$$

Визначаємо оптимальне керування $\bar{u}[0](\boldsymbol{x}[0])$. Оскільки $\boldsymbol{x}[0]$ задано, то непотрібно його розбитті на Z значень. Квантуємо лише $u[0]$: $u[0]_1$, $u[0]_2$,…, $u[0]_M$. Підставляємо $\boldsymbol{x}[0]$ та набір $u[0]_1$, $u[0]_2$,…, $u[0]_M$ в (2.2.11) та вибираємо мінімальне значення, записуємо його в $S[0]$. Фіксуємо керування, що йому відповідає, яке позначаємо $\bar{u}[0]$.

Далі застосуємо рух у зворотньому для визначення конкретних значень $\bar{u}[1]$, $\bar{u}[2]$,…, $\bar{u}[N]$. Отже, знаючи $\boldsymbol{x}[0]$ та $\bar{u}[0]$ з рівнянь стану об’єкту (2.2.1) визначаємо $\boldsymbol{x}[1]$ та округляємо до найближчого значення з набору $\boldsymbol{x}[1]_1$, $\mathbf{x}[1]_2$,…, $\mathbf{x}[1]_Z$, якому відповідає конкретне керування, яке фіксуємо як $\bar{u}[1]$. Далі на основі $\boldsymbol{x}[1]$ та $\bar{u}[1]$ знаходимо з (2.2.1) $\boldsymbol{x}[2]$, вибираємо з розрахованого набору $\boldsymbol{x}[2]_1$, $\boldsymbol{x}[2]_2$,…, $\boldsymbol{x}[2]_Z$ найближче значення та фіксуємо відповідне йому $\bar{u}[2]$. Проходимо весь шлях до $\boldsymbol{x}[N]$ послідовно відновлюючи всі значення оптимального керування.

Тож за $N+1$ кроків знаходимо послідовність оптимальних керувань, причому як видно, задача мінімізації критерію (2.2.2) за $N+1$ змінною зведена до $N+1$ задачі мінімізації критеріїв вигляду (2.2.7) за однією змінною.

3 МАТЕМАТИЧНІ МЕТОДИ ОПТИМІЗАЦІЇ

Оптимізація — в математиці, інформатиці та дослідженні операцій завдання знаходження екстремуму (мінімуму або максимуму) цільової функції в деякій області конечномерного векторного простору, обмеженою набором лінійних, нелінійних рівності, нерівностей [10].

Математичне програмування — це область математики, що розробляє теорію, чисельні методи рішення багатовимірних задач з обмеженнями. На відміну від класичної математики, математичне програмування займається математичними методами вирішення завдань знаходження найкращих варіантів з усіх можливих [11].

3.1 Класифікація методів оптимізації

Загальний запис завдань оптимізації задає велика різноманітність їх класів. Від класу завдання залежить підбір методу (ефективність її рішення). Класифікацію задач визначають: цільова функція і допустима область (задається системою нерівностей і рівностей або більш складним алгоритмом) [12].

Методи оптимізації класифікують відповідно до завдань оптимізації:

— Локальні методи: сходяться до якого-небудь локального екстремуму цільової функції. У разі унімодальної цільової функції, цей екстремум єдиний, і буде глобальним максимумом чи мінімумом.

— Глобальні методи: мають справу з багатоекстремальними цільовими функціями. При глобальному пошуку основним завданням є виявлення тенденцій глобальної поведінки цільової функції.

Існуючі в даний час методи пошуку можна розбити на три великі групи:

— детерміновані;

— випадкові (стохастичні);

— комбіновані.

За критерієм розмірності, методи оптимізації ділять на методи одновимірної оптимізації та методи багатовимірної оптимізації.

По виду цільової функції і допустимої множини, завдання оптимізації та методи їх вирішення можна розділити на наступні класи:

— Завдання оптимізації, в яких цільова функція $f(x)$ та обмеження $g_i(x)$, $i=\overline{1,m}$ є лінійними функціями, вирішуються так званими методами лінійного програмування.

— В іншому випадку мають справу з завданням нелінійного програмування і застосовують відповідні методи.

За вимогами до гладкості і наявності у цільової функції приватних похідних, їх також можна розділити на:

— прямі методи, можуть бути використані лише для обчислень цільової функції в точках наближень;

— методи першого порядку: вимагають обчислення перших приватних похідних функції;

— методи другого порядку: вимагають обчислення друге приватних похідних, тобто гессіан цільової функції.

Крім того, оптимізаційні методи діляться на наступні групи:

— аналітичні методи (наприклад, метод множників Лагранжа і умови Каруша-Куна-Таккера);

— чисельні методи;

— графічні методи.

Залежно від природи множини задача математичного програмування класифікуються як:

— завдання дискретного програмування (або комбінаторної оптимізації) – якщо задача звичайна або лічильна;

— задачі цілочислового програмування – якщо задача є підмножиною множини цілих чисел;

— задачі нелінійного програмування, якщо обмеження або цільова функція містять нелінійні функції і задача є підмножиною кінцевого векторного простору.

— Якщо ж все обмеження і цільова функція містять лише лінійні функції, то це – завдання лінійного програмування.

Крім того, розділами математичного програмування є параметричне програмування, динамічне програмування і стохастичне програмування.

Для використання методів оптимізації при розробленні технічних систем необхідно зрозуміти місце застосування математичних методів оптимізації у загальній схемі вирішення інженерної задачі.

Загалом прийнято вважати, що процес вирішення задачі за допомогою комп'ютера включає такі кроки:

— вербальна постановка задачі;

— формалізація задачі та розроблення математичної моделі;

— дослідження математичної моделі і розроблення методу вирішення задачі або вибір його із існуючих;

— розроблення алгоритму вирішення задачі;

— реалізація алгоритму (написання програми);

— відлагодження і тестування програми;

— документування програми;

— перевірка результатів на практиці.

За результатами останнього кроку можливе повернення на будь-який з попередніх кроків і продовження процесу, який, по суті, є послідовно-ітераційним.

Для нас із цієї схеми найважливішими є перші етапи, коли необхідно поставити задачу, розробити модель і відповідний метод. Більш того, оскільки метод визначається результатами формалізації, доцільно розглянути перші два етапи більш детально. Традиційно сутність цих етапів зводиться до:

— визначення меж системи, яка підлягає оптимізації;

— визначення кількісного критерія, на основі якого можна проводити аналіз варіантів з метою вибору найкращого з них;

— підібору внутрішньосистемних змінних, які використовують для визначення характеристик і ідентифікації варіантів;

—побудови моделі, яка відображає взаємозв'язки між змінними.

На основі виконання цих кроків, які часто називають процесом постановки задачі інженерної оптимізації, і виконується подальше поглиблення дослідження моделі, розроблення і обґрунтування методів, встановлення необхідних результатів.

Коректна постановка задачі вважається ключем до успіху оптимізаційного дослідження. Звичайно цю частину загальної схеми вирішення задач пов'язують більш з мистецтвом, ніж точною наукою [13].

Це мистецтво можна опанувати лише у процесі практичної діяльності на прикладах успішно виконаних розробок.

Для цього треба чітко уявляти переваги, недоліки і специфічні особливості різних методів оптимізації.

3.2 Метод рівномірного поділу

Метод відноситься до пасивних стратегій пошуку точки екстремуму. Здається початковий інтервал невизначеності [a,b] і кількість обчислень N. Обчислення проводяться в N рівновіддалених один від одного точках. При цьому інтервал ділиться на N+1 рівні інтервали. Шляхом порівняння величин $f(x_i)$, $i=\overline{1,N}$ знаходиться точка x_k, в якій значення функції найменше. Шукана точка x^* вважається укладеною в інтервалі $[x_{k-1}, x_{k+1}]$ (рисунок 3.2.1).

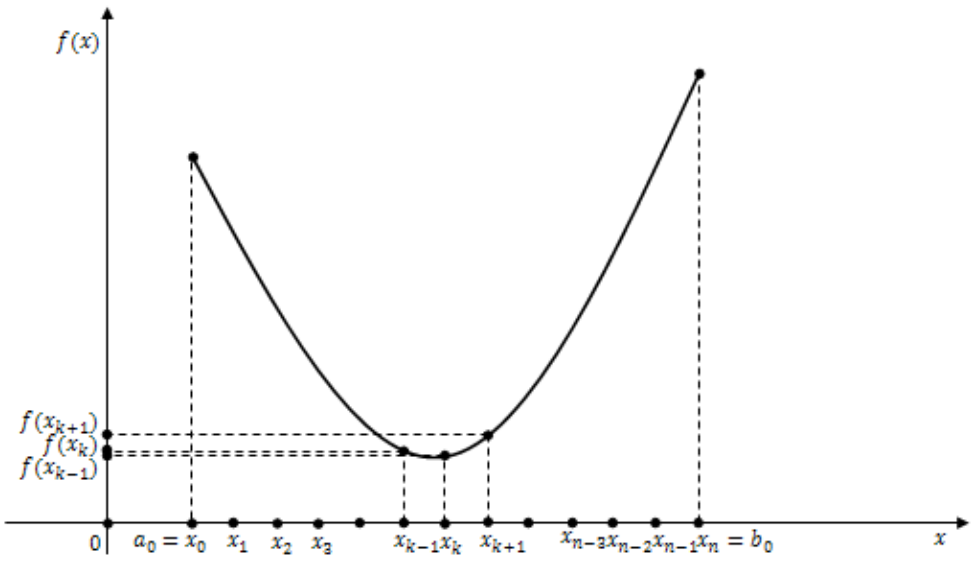


Рисунок 3.2.1 – Графічне представлення методу рівномірного пошуку[29]

Алгоритм заключається в наступному:

1) етап. Задається початковий інтервал невизначеності [a,b] і N — кількість обчислень функції.

2) етап. Обчислити точки $x_i = a_0 + i\frac{L_0}{N+1}, i = \overline{1,N}$, рівновіддалені один від одного.

3) етап. Обчислити значення функції N в знайдених точках $f(x_i), i = \overline{1,N}$

4) етап. Серед точок $x_i, i = \overline{1,N}$, знайти таку, в якій функція приймає найменше значення $f(x_k) = \min\limits_{1 \le i \le N} f(x_i)$.

5) етап. Точка мінімуму x^* належить інтервалу $x^* \in [x_{k-1}, x_{k+1}]$, на якому в якості наближеного рішення може бути обрана точка $x^* \cong x_k$.

Для оцінки збіжності використовується характеристика відносного зменшення початкового інтервалу невизначеності.

Примітка. Якщо розбиття інтервалу $[a,b]$ проводиться на $N+1$ рівні частини (метод перебору), то етапи 2-4 виглядають наступним чином [14]:

2) етап. Обчислити точки $x_i = a + i\frac{[a,b]}{N+1}, i = \overline{0,N+1}$, рівновіддалені один від одного.

3) етап. Обчислити значення функції в знайдених точках $f(x_i), i = \overline{0,N+1}$.

4) етап. Серед точок $x_i, i = \overline{0,N+1}$, знайти таку, в якій функція приймає найменше значення $f(x_k) = \min\limits_{0 \le i \le N+1} f(x_i)$.

Похибка знаходження точки мінімуму методом перебору не перевищує $\frac{[a,b]}{N+1}$.

3.3 Метод половиного поділу

Метод відноситься до послідовних стратегій і дозволяє виключати з подальшого розгляду на кожній ітерації в точності половину поточного інтервалу невизначеності. Алгоритм зменшення інтервалу заснований на аналізі величин функції в трьох точках, рівномірно розподілених на поточному інтервалі (ділять його на чотири рівні

частини) [20]. Пошук закінчується, якщо довжина поточного інтервалу невизначеності менше заданої величини.

Детально розглянмо МПД — задано функцію *f(x)*, вона неперервна на інтервалі [a,b]. Відомо що на відрізку [a,b] наявний хоча б один корінь.

Поділимо відрізок [a,b] навпіл точкою c, координати якої $c = \frac{a+b}{2}$ та обчислемо значення функції *f(c).* Можливо два випадки:

1) *f(a)f(c)>0* – тобто значення функції на кінцях відрізка [a,c] однакові по знаку, тоді корінь рівняння знаходиться на відрізку [c,b] і відрізок [a,c] можна виключити з подальшого розгляду, пересело точку a в точку c: a=c; *f(a)=f(c) (*рисунок 3.2.1).

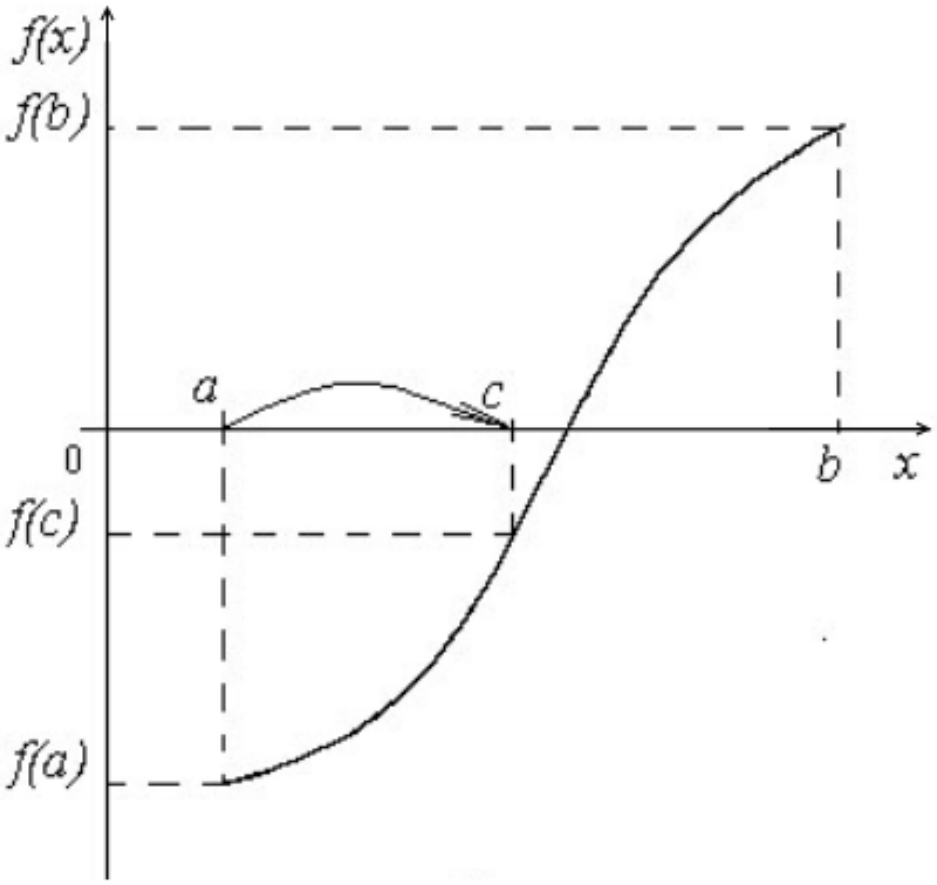


Рисунок 3.2.1 – Перший випадок МПД

2) *f(a)f(c)<0* — тобто значення функції на кінцях відрізка [a,c] протилежні по знаку, тоді корінь знаходиться на відрізку [a,c] і відрізок [c,b] можна виключити з подальшого розгляду, пересело точку b в точку c: b=c; *f(b)=f(c) (*рисунок 3.2.2).

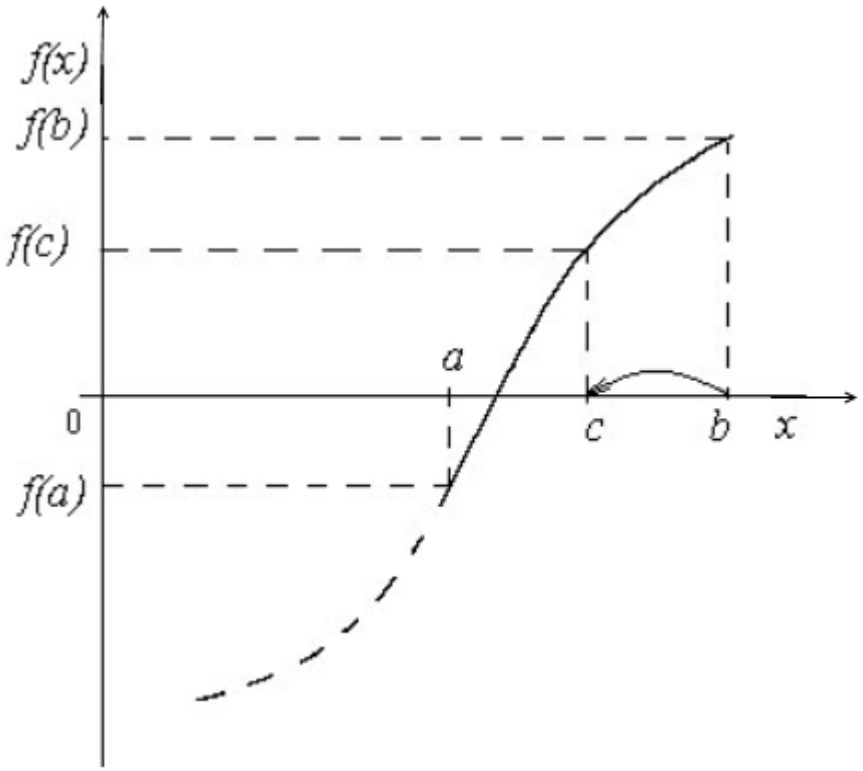


Рисунок 3.2.2 – Другий випадок МПД

Після виключення правої або лівої половини відрізка продовжується поділнавпіл до тих пір, поки довжина інтервалу [a, b] не стане менше деякої заданої малої величини ε, тобто $|b - a| < \varepsilon$ і тоді будь-яке значення аргументу з відрізка [a, b] можна вважати коренем з похибкою ε. Зазвичай приймають в якості кореня середину відрізка [23]. Відзначимо, що ε тут має сенс допустимої абсолютної похибки обчислення кореня.

Перевагою методу є його безумовна збіжність, якщо на інтервалі [a, b] є хоча б один корінь. Крім того, метод не використовує похідних. До недоліків відносять повільну збіжність, тобто досить велика кількість обчислень функції *f(x)* в порівнянні з іншими методами. Рекомендується до використання в тих випадках, якщо немає жорстких вимог до часу рахунки [24].

3.4 Метод золотого перетину

У методі золотого перетину в якості двох внутрішніх точок вибираються точки золотого перетину.

Точка виробляє золотий перетин, якщо відношення довжини всього відрізка до більшої частини дорівнює відношенню більшої частини до меншої частини. На відрізку $[a,b]$ є дві симетричні щодо його кінців точки c і d:

$$\tau = \frac{b-a}{b-c} = \frac{b-c}{d-a} = \frac{b-a}{d-a} = \frac{d-a}{b-d} = \frac{1+\sqrt{5}}{2} \cong 1{,}618 \qquad (3.4.1)$$

Точка c виробляє золотий перетин відрізка [a,d], а точка d – відрізка [c,b] (рисунок 3.4.1).

Метод відноситься до послідовних стратегіям. Здається початковий інтервал невизначеності і необхідної точності. Алгоритм спирається на аналіз значень функції в двох точках. Якості точок обчислення функції вибираються точки золотого перетину. На кожній ітерації, крім першого, потрібно тільки одне нове обчислення функції. Пошук закінчується, якщо довжина поточного інтервалу невизначеності менше заданої величини.

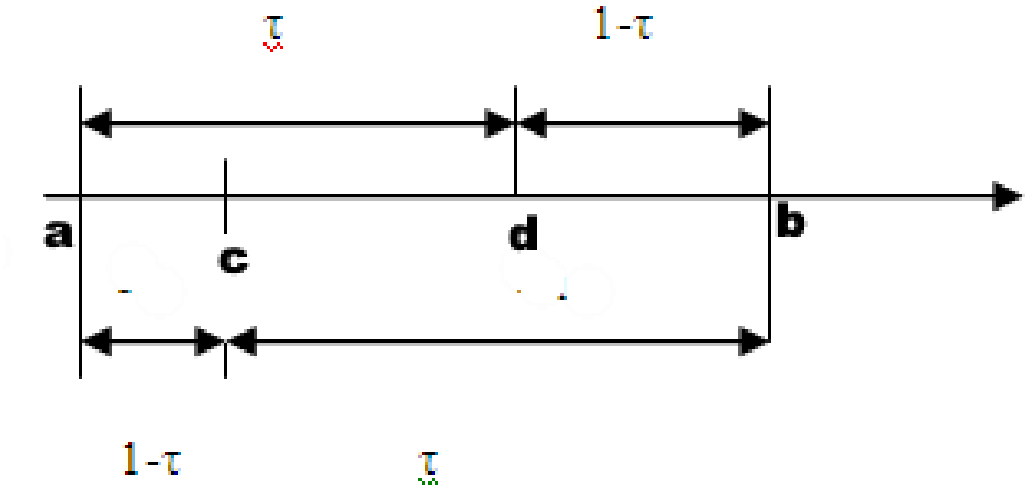


Рисунок 3.4.1 – Зображення МЗП

Алгоритм пошуку мінімуму функції зводиться до виконання наступних етапів.

1) етап. Здається початковий інтервал невизначеності $[a,b]$ і $\alpha > 0$ — необхідна точність.

2) етап. Задати $k = 0$.

3) етап. Обчислити $c_k = a + \frac{3-\sqrt{5}}{2}(b-a),\ d_k = a + b - c_0$.

4) етап. Обчислити $f(c_k),\ f(z_k)$.

5) етап. Якщо $f(c_k) \le f(d_k)$, то прийняти $a_{k+1} = a_k, b_{k+1} = d_k$ і $c_{k+1} = a_{k+1} + b_{k+1} - c_k, d_{k+1} = c_k$. Перейти до етапу 6. Якщо $f(c_k) > f(d_k)$, прийняти $a_{k+1} = c_k, b_{k+1} = b_k$ і $c_{k+1} = d_k, d_{k+1} = a_{k+1} + b_{k+1} - d_k$.

6) етап. Обчислити $\Delta = |a_{k+1} - b_{k+1}|$ і перевірити умова закінчення пошуку. Якщо $\Delta \le \alpha$, процес пошуку припиняється і $x^* \in [a_{k+1} + b_{k+1}]$. Якості наближеного рішення приймають середину останнього інтервалу $x^* \cong \frac{a_{k+1} + b_{k+1}}{2}$.

Якщо $\Delta > \alpha$, прийняти $k = k + 1$ і перейти до етапу 4.

3.5 Метод покординатного спуску

Тепер розглянемо багатовимірну оптимізацію, а точніше МПС. З курсу математики відомо, що напрямок найбільшого зростання будь-якої функції, в нашому випадку $F = f(x_1, x_2, \dots, x_n)$характеризується її градієнтом (3.5.1):

$$gradf = \frac{\partial f}{\partial x_1} e_1 + \frac{\partial f}{\partial x_2} e_2 + \cdots + \frac{\partial f}{\partial x_n} e_n \qquad (3.5.1)$$

де $e_1, e_2, \dots, e_n$ — одиничні вектори у напрямку координатних осей. Отже, напрям, протилежний градієнтному, вкаже напрямок найбільшого спадання функції а методи, засновані на виборі шляху оптимізації за допомогою градієнта, називаються градієнтними.

Процес відшукання точки мінімуму функції F за методом градієнтного спуску полягає в наступному: на початку вибираємо деяку початкову точку $M_0(x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})$ і обчислюємо в ній градієнт функції F. Далі, робимо крок у антиградієнтному напрямку $x^1 = x^0 - a^1 gradf(M_0)$ (де $a^1 > 0$). У результаті отримуємо нову точку $M_1(x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$, значення функції в якій зазвичай менше за значення функції в точці M_0. Якщо ця умова не виконується, тобто значення функції не змінилося або навіть зросла, то потрібно зменшити крок $a^1(a^2 = \frac{a^1}{2})$, після чого, у новій точці обчислюємо градієнт і знову робимо крок у зворотному до нього напрямку $x^2 = x^1 - a^2 gradf(M_1)$.

Процес продовжується до отримання найменшого значення цільової функції. Строго кажучи, момент закінчення пошуку настане тоді, коли рух з отриманої точки, при виборі будь-якого кроку, призводить до зростання значення цільової функції. Якщо мінімум функції досягається всередині розглядуваної області, то в цій точці градієнт дорівнює нулю, що також може служити сигналом про закінчення процесу оптимізації [16].

Формули для частинних похідних можна отримати в явному вигляді лише в тому випадку, коли цільова функція задана аналітично. В іншому випадку ці похідні обчислюються за допомогою чисельного диференціювання (3.5.2):

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x_1,\dots,x_i+\Delta x_i,\dots,x_n)-f(x_1,\dots,x_i,\dots,x_n)}{\Delta x_i}, i = 1,2,\dots,n \qquad (3.5.2)$$

Зауважимо, що знайти точку мінімуму функції F можна також шляхом зведення багатовимірної задачі оптимізації до послідовності одновимірних задач на кожному кроці оптимізації. Такий спосіб називається МПС. Різниця полягає в тому, що у методі градієнтного спуску напрямок оптимізації визначається градієнтом цільової функції, тоді як у методі покоординатного спуску проводиться спуск на кожному кроці вздовж одного з координатних напрямків. МПС зображений на рисунку 3.5.1.

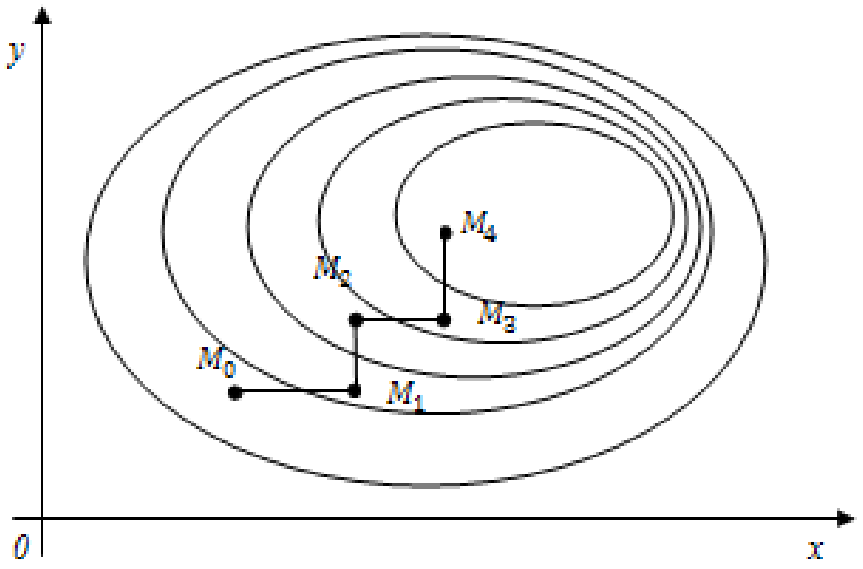


Рисунок 3.5.1 – Робота МПС [30]

4 РОЗРОБЛЕННЯ АЛГОРИТМІВ ДЛЯ МЕТОДУ ДИНАМІЧНОГО ПРОГРАМУВАННЯ ДЛЯ ДИСКРЕТНОЇ ЗАДАЧІ ОПТИМАЛЬНОГО КЕРУВАННЯ

Ідея, новизна даної десертації полягає у об’єднанні методу динамічного програмування для дискретної задачі з методами оптимізації, зокрема: рівномірного поділу, половинного поділу, золотого перетину та покоординатного спуску, та досліджено впливу цих методів оптимізації на швидкодію.

На першому кроці дослідження було вибрано об’єкт керування першого порядку, який представлений рівняннями станів. Задано початковий та кінцевий стан. Досліджено область допустимих станів за допомогою моделі в MATLAB/Simulink (детально в розділі 4.2).

На другому етапі було розроблено програмний засіб на базі пакету MATLAB в середовищі MATLAB для класичної задачі МДП в якій використовується алгоритм рівномірного поділу стану об’єкта (розділ 4.2).

Далі аналогічно до попереднього етапу МРП був спочатку замінений на МПП (розділ 4.3), а далі на МЗП результати в розділі 4.4.

Наступний крок був моделювання отимальної системи керування з використанням отриманої керуючої послідовності.

Далі було проведено дослідження впливу методів оптимізації на МДП з використанням об’єкту другого порядку. В процесі дослідження було виявлено, що використання методів одновимірної оптимізації для багатовимірних об’єктів не є ефективним, тому було обрано алгоритм покоординатного спуску та розроблено програмний засіб на базі пакету MATLAB детально в розділі 4.8.

4.1 Розроблення алгоритму для методу динамічного програмувння

Алгоритм методу динамічного програмування. Загальний алгоритм скаладається з алгоритму прямого проходу та алгоритм зворотнього проходу.

Загальний алгоритм:

Крок 1. Задаємо почакові умови: початковий стан об'єкту *x[0]*, рівняння зв'язку *x[n+1]=x[n]+φ(x[n],u[n])*, критерій оптимальності $I = \sum_{n=0}^{N} G(u[n], x[n])$, область допустимих значень Ω(u).

Крок 2. Зворотній прохід у якому знаходяться значення пов'язаних величин $x[n]^{(r)}, \bar{u}[n](x[n]^r, S[n](x[n]^{(r)})$, для кожного можливого значення стану $x^{(r)}$, *r=1,2,…,R*, для кожного моменту часу t_n, *n=0,1,…,N*.

Крок 3. Прямий прохід у якому потрібно знайти значення зміної стану *x[n] з* рівняння зв'язку *x[n+1]=x[n]+φ(x[n],u[n])* та відповідні їм значення оптимальних управлінь $,\bar{u}[n]$ шляхом вибору зі знайдених наборів $x[n]^{(r)}, \bar{u}[n](x[n]^r)$ для кожного моменту часу t_n, *n=0,1,…,N*.

Зворотній прохід:

Крок 1. Розбиваємо діапазон значень керувань *u* на *(M-1)* ділянок – $u^{(1)}$, $u^{(2)}$,…, $u^{(M)}$.

Крок 2. Розбиваємоо діапазон значень станів на *(R-1)* ділянок – $x^{(1)}$, $x^{(2)}$,…,$x^{(r)}$.

Крок 3. Вважаємо що керування *u[0],u[1],…,u[N-1]* знайдено і розраховано усі відповідні значення станів. Знаходимо *u[N]* для цього переберемо усі значення стану об'єкта в момент часу *N*.

Крок 3.1. Вважаємо, що $x[N]=x[N]^{(r)}$. Знаходимо усі значення функції $G(u[N]^{(m)}, x[N]^{(r)})$, перебравши з значенням стану усі можливі значення керування.

Крок 3.2. Знаходимо мінімальне значення функції *G* та записуємо в $S[N](x[N]^{(r)})$.

Крок 3.3. Зберігаємо комбінацію $x[N]^{(r)}$, $u[N](x[N]^{(r)})$, $S[N](x[N]^{(r)})$

Крок 3.4. Повертаємося до кроку 3.1 та пророблюємо всі наступні кроки з кожним значенням стану в даний момент часу. Якщо усі можливі керування знайдені переходимо до кроку 4.

Крок 4. Вважаємо що керування *u[0],u[1],…,u[N-n-1]* знайдено і розраховано усі відповідні значення станів *x[1],x[2],…,x[N-n]*. Знаходимо *u[N-n]* для цього виконаємо виконаємо кроки 4.1 – 4.7 у моменти часу *n=1,n=n+1,N-1*.

Крок 4.1. Вважаємо що $x[N-n]=x[N-n]^{(r)}$

Крок 4.2. Знаходимо значення $x[N-n+1]^{(m)}=x[N-n]^{(r)}+\varphi(x[N-n]^{(r)},u[N-n]^{(m)})$.

Крок 4.3. Округляємо $x[N-n+1]^{(m)}$ до найблищього квантового значення і дістаємо з пам'яті відповідну йому величину $S[N-n+1](x[N-n+1]^{(m)})$.

Крок 4.4. Обчислюємо значення $G(u[N-n]^{(m)},x[N-n]^{(r)})+S[N-n+1](x[N-n+1]^{(m)})$.

Крок 4.5. Пророблюємо кроки 4.2 – 4.4 для усіх керувань.

Крок 4.5. Далі знаходимо мінімальне значення $G(u[N-n]^{(m)},x[N-n]^{(r)})+S[N-n+1](x[N-n+1]^{(m)})$ для відповідного стану об'єкту та записуємо його в $S[N-n](x[N-n]^{(r)})$.

Крок 4.6. Зберігаємо комбінацію $x[N-n]^{(r)},\bar{u}[N-n](x[N-n]^{(r)}$, $S[N-n](x[N-n]^{(r)})$.

Крок 4.7. Пророблюємо кроки 4.1 – 4.6 поки не переберемо усі стани об'єкту.

Крок 5. Знаходимо значення $u[0]$. Обчислюємо значення $x[1]=x[0]+\varphi(x[0],u[0]^{(m)})$.

Крок 5.1. Значення $x[1]^{(m)}$ округляємо до найближчого квантового значення і дістаємо з пам'яті відповідну йому велечину $S[1](x[1]^{(m)})$.

Крок 5.2. Обчислюємо значення $G(u[0]^{(m)},x[0])+S[1](x[1]^{(m)})$.

Крок 5.3. Виконуємо кроки 5 –5.3 для усіх значень керувань.

Крок 5.4. Знаходимо мінімальне значення $G(u[0]^{(m)},x[0])+S[1](x[1]^{(m)})$ та записуємо його в $S[0](x[0])$.

Крок 5.5. Зберігаємо комбінацію $x[0]$, $\bar{u}[0](x[0])$, $S[0](x[0])$.

Прямий прохід:

Крок 1. Обчислюємо $x[n+1]=x[n]+\varphi(x[n],u[n])$.

Крок 2. Знаходимо значення $x[n+1]$ округляємо до найближчого квантового значення і дістаємо з пам'яті відповідну йому величину $u[n+1]$.

Крок 3. Виконуємо крок 1 – 2 для кожного моменту часу n=0, n=n+1, N-1.

4.2 Розроблення програмного засобу на базі пакету MATLAB з використанням методу рівномірного поділу для об'єкту першого порядку

Алгоритм рівномірного поділу:

Крок 1. Задається початковий інтервал невизначеності [a,b] і N – кількість обчислень функції.

Крок 2. Обчислити точки $x_i = a_0 + i\frac{L_0}{N+1}, i = \overline{1,N}$, рівновіддалені один від одного.

Крок 3. Обчислити значення функції N в знайдених точках $f(x_i), i = \overline{1,N}$.

Крок 4. Серед точок $x_i, i = \overline{1,N}$, знайти таку, в якій функція приймає найменше значення $f(x_k) = \min\limits_{1 \le i \le N} f(x_i)$.

В даному розділі на основі алгоритму з розділу 4.1 з використанням алгоритму рівномірного поділу було написано програмний засіб на базі пакету MATLAB для об’єкту. Розглянемо детально написаний програмний засіб на базі пакету MATLAB.

Опис першого фрагменту програмного засобу на базі пакету MATLAB: вхідні умови: кількість ітерацій N=10, з початкової точки 5 прийти в 0. Рівняння стану (4.1.1), а критерій (4.1.2):

$$x[k+1] = x[k] + u[k] \tag{4.1.1}$$

$$\mathrm{I} = \int x^2 + u^2 \tag{4.1.2}$$

```
N=10;
Xstart=5;
Xend=0;
G= @(u,x) (x^2+u^2);
xnm=xr(r)+u(m);
```

Область допустимих керувань [1;-1] та область допустимих станів [-19;10]. Для знаходження усіх можливих рішень станів та керувань використовується МРП, в алгоритмі поділ керувань здійснюється на N=4, а станів на N=100:

```
M=4;
Umin=-1;
```

```
Umax=1;
Ustep=((abs(Umin)+abs(Umax))/(M-1));
u=(Umin:Ustep:Umax);
 R=100;
xMin=-19;
xMax=10;
xStep=((abs(xMin)+abs(xMax))/(R-1));
xr=(xMin:xStep:xMax);
```

Зворотній прохід ділиться на 3 цикли: останій момент часу, середина та початковий. Різниця заключається в знаходженні S, на першому кроці він відсутній, та останьому єдиний. Розглянемо середину, а інші 2 цикли можна спостерігати в додатку В.

```
for n=1:N-1   // перебираємо момент часу з першого по N-1;
for r=1:R  // перебираємо усі знайдені раніше стани;
g=zeros(M); // ініціалізуємо порожній вектор G;
for m=1:M // перебираємо усі керування;
xnm=xr(r)+u(m); // знаходимо поточний стан;
index=findclosestkvantum(xnm,result(N-n+1,:)); // знаходимо найближче значення з попередніх результатів;
g(m,1)=G(u(m),xr(r))+result(N-n+1,index).S; // знаходимо S;
g(m,2)=u(m); // запам’ятовуємо керування;
end
index=minimum(g(:,1)); // Знаходимо мінімальне S;
result(N-n,r)=ResultPoint(xr(r),g(index,2),g(index,1)); // Записуємо результат в пам’ять;
end
end
```

Та прямий прохід складається з знаходження керування в перший момент часу, та циклу для знаходження подальших керувань.

```
un=result0.U;// дістаємо поточне u;
```

```
xn=result0.X+un;// знаходження значення стану в перший момент часу;
index=findclosestkvantum(xn,result(1,:)); // знаходження найближчого значення стан;
un=result(1,index).U; // дістаємо відповідне керування стану;
 for n=2:N
    xn=xn+un;
    index=findclosestkvantum(xn,result(n,:));
    un=result(n,index).U;
end
```

Функція знаходження найближчого значення та мінімального буде представлена в додатку В.

Виконавши програмний засіб на базі пакету MATLAB можна спостерігати результати моделювання такі:

```
Direct walk
n=0; x=5; U=-1
n=1; x=4.14141; U=-1.000
n=2; x=2.96970; U=-1.000
n=3; x=2.09091; U=-1.000
n=4; x=0.91919; U=-0.333
n=5; x=0.62626; U=-0.333
n=6; x=0.33333; U=-0.333
n=7; x=0.04040; U=-0.333
n=8; x=-0.25253; U=0.333
n=9; x=0.04040; U=-0.333
n=10; x=-0.25253; U=-0.333
******* End *******
```

Результати знайденого керування потрібно записати в матрицю та подати на об’єкт – спостерігати результати. (розділ 4.9).

4.3 Розроблення програмного засобу на базі пакету MATLAB з використанням методу половинного поділу для об’єкту першого порядку

Алгоритм половиного ділення скаладається з наступних кроків:

Крок 1. Задамо відрізок [a;b] (a – ліва границя обмеження, b — права границя обмеження стану об’єкта).

Крок 2. Розділимо відрізок [a;b] навпіл точкою $c = \frac{a+b}{2}$, якщо функція в точні c відмінна від нуля то можливі два варіанти розв’язку.

Крок 3. Перший якщо *f(a)*f(c)>0,* ліва границя точка *a* переміщається в точку поділу *c*.

Крок 4. В іншому випадку *f(a)*f(c)<0,* права границя *b* переміщається в точку поділу *c.*

Крок 5. Далі все повторюється поки, а інтервал звужується, поки |a-b|<ε

Далі в програмному засобі замінемо МРП на МПП. Об’єкт, кількість ітерацій, обмеження керування та станів, критерій оптимальності, початковий та кінцевий стан об’єкта не зміний. Змінюється тільки метод оптимізації для станів відповідно змінюється і цикли в зворотньому проході.

```
epsilon=0.07; // ε
for n=1:N-1 // перебираємо момент часу з першого по N-1;
   a=xMin; // початок відрізку x
   b=xMax; // кінець відрізку x
   i=1;
   while 1
      x1=(a+b-beta)/2; // знаходження x1(c)
      x2=(a+b+beta)/2;// знаходження x2 (d)
         f1 = findMin(x1,u,M,N-n,result,G);// знаходження мінімум по x1
      f2 = findMin(x2,u,M,N-n,result,G); знаходження мінімум по x2
      if f1.S > f2.S // порівняння значень S функцій
         a=x1;// присвоєння початку відрізку значення x1
         result(N-n,i)=f2;
```

```
        end
        if f2.S > f1.S //порівняння значень S функцій
           b=x2;// присвоєння кінцю відрізка значення x2
           result(N-n,i)=f1;
                end
        if (b-a)/2 < epsilon // якщо ε більше за половину різницю кінці відрізка, то закінчуємо обчислення
           break
        end
       i=i+1;
     end
  end
```

Прямий прохід не змінюється, а нульовий та N момент часу аналогічно до попереднього алгоритму тільки з використанням МПД. Виконавши програмний засіб на базі пакету MATLAB – результати, а моделювання в розділі 4.9: А весь програмний засіб на базі пакету MATLAB представлений у додатку Г.

```
Direct walk
n=0; x=5; U=-0.33333
n=1; x=4.6750000000; U=-1.0000000000
n=2; x=3.6750000000; U=-1.0000000000
n=3; x=2.6750000000; U=-1.0000000000
n=4; x=1.8687500000; U=-1.0000000000
n=5; x=1.3687500000; U=-1.0000000000
n=6; x=0.8593750000; U=0.3333333333
n=7; x=0.5091796875; U=-0.3333333333
n=8; x=0.2091796875; U=-0.3333333333
n=9; x=-0.0091796875; U=-0.3333333333
n=10; x=0.0091796875; U=-0.3333333333
******* End *******
```

4.4 Розроблення програмного засобу на базі пакету MATLAB з використанням методу золотого перетину для об’єкту першого порядку

Алгоритм золотого перетину заключається в:

Крок 1. Здається початковий інтервал невизначеності $[a,b]$ і $\alpha > 0$ — необхідна точність.

Крок 2. Задати $k = 0$.

Крок 3. На першій ітерації заданий відрізок [a;b] ділиться двома симетричними відносно його центру точками $c_k = a + \frac{3-\sqrt{5}}{2}(b-a),\ d_k = a + b - c_0$.

Крок 4. Обчислити $f(c_k),\ f(d_k)$.

Крок 5. Якщо $f(c_k) \le f(d_k)$, то прийняти $a_{k+1} = a_k, b_{k+1} = d_k$ і $c_{k+1} = a_{k+1} + b_{k+1} - c_k, d_{k+1} = c_k$. Перейти до кроку 6. Якщо $f(c_k) > f(d_k)$, прийняти $a_{k+1} = c_k, b_{k+1} = b_k$ і $c_{k+1} = d_k, d_{k+1} = a_{k+1} + b_{k+1} - d_k$.

Крок 6. Обчислити $\Delta = \left|a_{k+1} - b_{k+1}\right|$ і перевірити умова закінчення пошуку. Якщо $\Delta \le \alpha$ процес пошуку припиняється і $x^* \in [a_{k+1} + b_{k+1}]$. Якості наближеного рішення приймають середину останнього інтервалу $x^* \cong \frac{a_{k+1} + b_{k+1}}{2}$. Якщо $\Delta > \alpha$, прийняти $k = k + 1$ і перейти до етапу 4.

Далі використаємо алгоритм МЗП. Об’єкт, кількість ітерацій, обмеження керування та станів, критерій оптимальності, початковий та кінцевий стан об’єкта не зміний. Змінюється тільки метод оптимізації для станів відповідно змінюється і цикли в зворотньому проході. Детально програмний продукт представлений в додатку Д.

Прямий прохід не змінюється змінюється тільки зворотній, Розглянемо зворотній прохід, в момент часу від 1 до N-1, два інші моменти часу аналогічно.

```
for n=1:N-1 // перебір моментів часу від одного до N-1
   a=xMin; // присвоїти a початкове мінімальне значення x
   b=xMax;// присвоїти b початкове максимальне значення x
   i=1;
```

```
        while 1
            x1=b-((b-a)/tau); // знаходження значення x1 за МЗП
            x2=a+((b-a)/tau); // знаходження значення x2 за МЗП
             f1 = findMin(x1,u,M,N-n,result,G);// знаходження мінімум по x1
            f2 = findMin(x2,u,M,N-n,result,G);// знаходження мінімум по x2
                 if f1.S >= f2.S// порівняння значень S в функціях
                a=x1;// якщо умова виконується а присвоюємо значення x1
                result(N-n,i)=f2;
            else
                b=x2; // якщо ні, присвоюємо b значення x2
                result(N-n,i)=f1;
            end
            if abs(b-a)/2 < epsilon// якщо ε більше за половину різницю кінці відрізка, то
закінчуємо обчислення
                break
            end
          i=i+1;
        end
    end
```

Результати виконання програмного засобу:

```
Direct walk
n=0; x=5; U=-0.33333
n=1; x=4.5391002838; U=-0.3333333333
n=2; x=3.5391002838; U=-0.3333333333
n=3; x=3.05391002838; U=-0.3333333333
n=4; x=2.5391002838; U=-0.3333333333
n=5; x=1.5391002838; U=-0.3333333333
n=6; x=1.05391002838; U=-0.3333333333
n=7; x=0.85391002838; U=-0.3333333333
n=8; x=0.5391002838; U=-0.3333333333
```

n=9; x=-0.5391002838; U=-0.3333333333

n=10; x=0.05391002838; U=-0.3333333333

******* End *******

4.5 Розроблення програмного засобу на базі пакету MATLAB з використанням методу рівномірного поділу для об’єкту другого порядку

Алгоритм рівномірного поділу:

Крок 1. Задається початковий інтервал невизначеності [a,b] і N - кількість обчислень функції.

Крок 2. Обчислити точки $x_i = a_0 + i\frac{L_0}{N+1}, i = \overline{1,N}$, рівновіддалені один від одного.

Крок 3. Обчислити значення функції N в знайдених точках $f(x_i), i = \overline{1,N}$.

Крок 4. Серед точок $x_i, i = \overline{1,N}$, знайти таку, в якій функція приймає найменше значення $f(x_k) = \min\limits_{1 \le i \le N} f(x_i)$.

В даному розділі алгоритму МДП було написано програмний засіб на базі пакету MATLAB для об’єкту другого порядку з методом оптимізація МРП. Розглянемо детально написаний програмний засіб на базі пакету MATLAB.

Опис першого фрагменту програмного засібу на базі пакету MATLAB: вхідні умови: кількість ітерацій N=10, з початкової точки [5;-5] прийти в [0;0]. Рівняння стану (4.5.1), а критерій (4.5.2):

$$\begin{cases} x_1[k+1] = 0{,}1764x_1[k] - 0.07686x_2[k] + 0{,}25u[k]; \\ x_2[k+1] = 0{,}5x_1[k+1] + 0{,}5x_2[k+1]. \end{cases} \quad (4.5.1)$$

$$\mathrm{I} = \int {x_1}^2 + {x_2}^2 + u^2 \quad (4.5.2)$$

X1start=5;

X1end=0;

X2start=-5;

```
X2end=0;
Fx1= @(x,u) (0.1764*x(1)-0.07686*x(2)+0.25*u);
Fx2= @(x,u) (0.5*x(1)+0.5*x(2));
G= @(u,x1,x2) (x1^2+x2^2+u^2);
```

Область допустимих керувань [1;-1], область допустимих станів x_1 [-20;30] та область допустимих станів x_2 [-20;30]. Для знаходження усіх можливих рішень станів та керувань використовується МРП, в алгоритмі поділ керувань здійснюється на N=10, а станів на N=100:

```
M=10;
Umin=-1;
Umax=1;
Ustep=((abs(Umin)+abs(Umax))/(M-1));// поділ усіх значень u на рівні частини
u=(Umin:Ustep:Umax);
 R=100;
x1Min=-20;
x1Max=30;
x2Min=-20;
x2Max=30;
x1Step=((abs(x1Min)+abs(x1Max))/(R-1));// поділ на рівні частини x1
x2Step=((abs(x2Min)+abs(x2Max))/(R-1)); поділ на рівні частини x2
x1=(x1Min:x1Step:x1Max);
x2=(x2Min:x2Step:x2Max);
```

Зворотній прохід ділиться на 3 цикли: останій момент часу, середина та початковий. Різниця заключається в знаходженні S, на першому кроці він відсутній, та останьому єдиний. Розглянемо момент часу від 1 до N-1, а інші 2 цикли можна спостерігати у додатку Д.

```
 for n=1:N-1// перебір моментів часу від одного до N-1
    for r1=1:R   // перебираємо усі значення x1
       for r2=1:R   // перебираємо усі значення x2
```

```
            g=zeros(M); // заповнюємо матрицю g нулями
            for m=1:M // перебираємо усі значення керування
               xnm1=Fi1(x1(r1),x2(r2),u(m)); // знаходимо значення x1 з
«наступного» кроку
               xnm2=Fi2(x1(r1),x2(r2),u(m)); // знаходимо значення x2 з
«наступного» кроку
         [index1,index2]=findclosestkvantum2d_2(xnm1,xnm2,result(N-n+1,:,:));//
знаходимо найближче значення та витягуємо S
               g(m,1)=G(u(m),x1(r1),x2(r2))+result(N-n+1,index1,index2).S; //
Записуємо в пам’ять нове значення g
               g(m,2)=u(m);// та записуємо відповідне йому u
            end
            index=minimum(g(:,1)); // знаходимо мінімальне g
            result(N-n,r1,r2)=ResultPoint2D(x1(r1),x2(r2),g(index,2),g(index,1)); //
записуємо в пам’ять результат керування, станів та g
          end
        end
      end
```

Прямий прохід складається з знаходження керування в перший момент часу, та циклу для знаходження подальших керувань.

```
un=result0.U;
xn1=Fi1(result0.X1,result0.X2,un); // знаходження значення для моменту часу 1
x1
xn2=Fi2(result0.X1,result0.X2,un);// знаходження значення для моменту часу 1 x2
[index1,index2]=findclosestkvantum2d_2(xn1,xn2,result(1,:,:)); знаходження
найближчого з «наступного» моменту часу
un=result(1,index1,index2).U; // витягуємо з пам’яті відповідне значення u
for n=2:N // знаходження значень u в моменти часу від 2 до N
   xn1=Fi1(xn1,xn2,un);// знаходження значення x1
   xn2=Fi2(xn1,xn2,un);// знаходження значення x1
```

[index1,index2]=findclosestkvantum2d_2(xn1,xn2,result(n,:,:)); // знаходження найближчого значення з «наступного кроку»

un=result(n,index1,index2).U;// знаходження відповідного u

end

Функція знаходження найближчого значення та мінімального буде представлена в додатку Е. Виконавши програмний засіб на базі пакету MATLAB можна спостерігати результати моделювання такі:

Direct walk

n=0; x1=5; x2=-5; U=-0.052632

n=1; x1=0,00000000000; x2=-1.551851852; U=-0.0526315789

n=2; x1=-0.5629629630; x2=-0.239629629630; U=-0.0526315789

n=3; x1=-0.4814814815; x2=0.000000000005; U=-0.0526315789

n=4; x1=-0.2907407407; x2=-0.00007407407; U=-0.3684210526

n=5; x1=0.07407407407; x2=-0.07407407407; U=0.3684210526

n=6; x1=0.1286450975; x2=0.0900000000; U=-0.0526315789

n=7; x1=0.0000000000; x2=0.0000000000; U=-0.0526315789

n=8; x1=0.0000000000; x2=0.0000000000; U=-0.0526315789

n=9; x1=0.0000000000; x2=0.0000000000; U=-0.0526315789

n=10; x1=0.0000000000; x2=0.0000000000; U=-0.0526315789

******* End *******

4.6 Розроблення програмного засобу на базі пакету MATLAB з використанням методу половинного поділу для об’єкту другого порядку

Алгоритм половиного ділення скаладається з наступних кроків:

Крок 1. Задамо відрізок [a;b] (a – ліва границя обмеження, b — права границя обмеження стану об’єкта).

Крок 2. Розділимо відрізок [a;b] навпіл точкою $c = \frac{a+b}{2}$, якщо функція в точні c відмінна від нуля то можливі два варіанти розв’язку.

Крок 3. Перший якщо *f(a)*f(c)>0,* ліва границя точка *a* переміщається в точку поділу *c*.

Крок 4. В іншому випадку *f(a)*f(c)<0,* права границя *b* переміщається в точку поділу *c*.

Крок 5. Далі все повторюється поки, а інтервал звужується, поки |a-b|<ε

Далі було розроблено МПП. Було вдосконалено програмний засіб на базі пакету MATLAB на написано основий файл з загальними кроками для різних алгоритмів — Generic.

```
N=10; // кількість проміжків часу
Xstart=[-5 5]; // початкова координата точки X
Xend=[0 0]; // бажана кінцева координата точки X
 Xmin=[-8 -9]; // задаємо мінімальні межі для X1 та X2
Xmax=[12 12]; // задаємо максимальні межі X1 та X2
 Fx1= @(x,u) (0.1764*x(1)-0.07686*x(2)+0.25*u); // перше рівняння об’єкту
Fx2= @(x,u) (0.5*x(1)+0.5*x(2));// друге рівняння об’єкту
NextX=@(x,u) ([Fx1(x,u) Fx2(x,u)]); // задаємо єдину функцію для обох рівнянь
 f=@(priv,xmin,xmax,fx) (polovinoe2D(priv,xmin,xmax,fx)); // виклик функції половинного ділення
 t=f([],Xmin,Xmax,NextX);
result(N,:)=t;
sz=size(t);
for n=1:N-1
   t=f(result(N-n+1,:),Xmin,Xmax,NextX); // обраховуємо результат для моменту часу N-n
   sz2=size(t); // записуємо розмір результату
   if sz2(2) > sz(2) // перевіряємо розмірність результату, якщо
      result(N-n,:)=t(1:sz(2)); // записуємо результат до загального результату
   else
      t(sz(2))=ResultPoint2D;
      result(N-n,:)=t; // записуємо результат до загального результату
```

```
    end
end
result0=func2D(Xstart,result(1,:),NextX); // кінцевий результат
```

Реалізація прямого проходу:

```
un=result0.U; // початкове значення u
xn=NextX([result0.X1 result0.X2],un); // знаходимо стан об’єкта в перший момент часу
index=findclosestkvantum2d(xn,result(1,:)); шукаємо найближче наступне значення в перший момент часу
un=result(1,index).U;// знайдене керування
for n=2:N // цикл перебору з 2 моменту часу N
    xn=NextX(xn,un);// знаходимо стан об’єкту
    index=findclosestkvantum2d(xn,result(n,:)); // знаходимо найближче значення з «наступного» моменту часу
    resultN=result(n,:); // записуємо результат
    un=resultN(index).U;// беремо відповідне потрібне керування
end
```

Та окремо розроблений функцію для половиного ділення:

```
function result = polovinoe2D(priv,xmin,xmax,fx)
    result=[];
    beta=0.01;
    epsilon=0.02; // значення ε
     if beta >= (xmax(1)-xmin(1))/2 || beta >= (xmax(2)-xmin(2))/2
        return
    end
     a1=xmin(1);// присвоєння a1 початок відрізка першого стану об’єкта
    b1=xmax(1); присвоєння b1 кінець відрізка першого стану об’єкта
    i1=1;
    while 1
        x11=(a1+b1-beta)/2; // знаходження x1 для першого стану об’єкта
```

```
x12=(a1+b1+beta)/2; // знаходження x2 для першого стану об’єкта
 a2=xmin(2); // присвоєння a2 початок відрізка другого стану об’єкта
b2=xmax(2);// присвоєння b2 кінця відрізка другого стану об’єкта
while 1
   x21=(a2+b2-beta)/2; // знаходження x1 для другого стану об’єкта
   x22=(a2+b2+beta)/2; // знаходження x2 для другого стану об’єкта
    f11=func2D([x11 x21],priv,fx); // знаходження значення функції для x11
x21
   f12=func2D([x11 x22],priv,fx); // знаходження значення функції для x11
x22
    if f11.S >= f12.S // умова порівняння функцій f11 >= f12 по S (для першого
стану об’єкту)
      a2=f11.X2;// якщо умова виконується a2 присвоюємо значення
      f1=f12;
   else
      b2=f12.X2; //якщо не виконується присвоюємо значення
      f1=f11;
   end
   if abs(b2-a2) < epsilon // умова закінчення роботи алгоритму
      break
   end
end
 a2=xmin(2);// присвоєння a2 початок відрізка другого стану об’єкта
b2=xmax(2);// присвоєння и2 кінця відрізка другого стану об’єкта
while 1
   x21=(a2+b2-beta)/2; // знаходимо значення x21
   x22=(a2+b2+beta)/2; // знаходимо значення x22
    f11=func2D([x12 x21],priv,fx); // знаходження значення функції для x12,
x21
```

```
            f12=func2D([x12 x22],priv,fx);// знаходження значення функції для x12, x22
             if f11.S >= f12.S // умова порівняння функцій f11 та f12 по S
               a2=f11.X2;//якщо вимова виконується a2 присвоюємо значення стану з функції f11
               f2=f12;// a f2 значення функції f12
            else
               b2=f12.X2; // якщо умова не виконується присвоюємо b2 значення стану функції f12
               f2=f11;//a f2 значення f11
            end
            if abs(b2-a2) < epsilon// умова закінчення роботи циклу
               break
            end
         end
         if f1.S >= f2.S // умова порівняння функцій f1 та f2 по S
            a1=f1.X1;// якщо умова виконується то a1 присвоюємо значення першого стану функції f1
            result=[result,f2]; // записуємо результат
         else
            b1=f2.X1;// якщо умова не виконується b1 присвоюємо значення другого стану функції f2
            result=[result,f1];// записуємо результат
         end
         if abs(b1-a1) < epsilon // умова закінчення роботи циклу
            break
         end
      end
   end
```

Детально програмний засіб на базі пакету MATLAB представлений в додатку Ж. Виконавши програмний засіб на базі пакету MATLAB можна спостерігати результати моделювання :

Direct walk

n=0; x1=5.000000000000000;x2=-5.000000000000000;U=-0.034482758620690

n=1; x1=-0.000065000000000; x2=-1.643518066406250;U=-0.034482758620690

n=2; x1=-0.81406250000000;x2=0.213518066406250;U=-0.034482758620690

n=3; x1=0.5423808593750000;x2=0.0003518066406250;U=0.034482758620690

n=4; x1=0.2365191562500000;x2=0.0003518066406250;U=0.034482758620690

n=5; x1=0.0042851562500000;x2=0.0003518066406250;U=0.034482758620690

n=6; x1=0.000000000000000;x2=0.0000000000000000;U=0.034482758620690

n=7; x1=0.000000000000000;x2=0.0000000000000000;U=0.034482758620690

n=8; x1=0.000000000000000;x2=0.0000000000000000;U=0.034482758620690

n=9; x1=0.000000000000000;x2=0.0000000000000000;U=0.034482758620690

n=10; x1=0.000000000000000;x2=0.0000000000000000;U=0.034482758620690

******* End *******

4.7 Розроблення програмного засобу на базі пакету MATLAB з використанням методу золотого перетину для об’єкту другого порядку

Алгоритм золотого перетину заключається в:

Крок 1. Здається початковий інтервал невизначеності $[a,b]$ і $\alpha > 0$ — необхідна точність.

Крок 2. Задати $k = 0$.

Крок 3. На першій ітерації заданий відрізок [a;b] ділиться двома симетричними відносно його центру точками $c_k = a + \frac{3-\sqrt{5}}{2}(b-a),\ d_k = a + b - c_0$.

Крок 4. Обчислити $f(c_k),\ f(d_k)$.

Крок 5. Якщо $f(c_k) \le f(d_k)$, то прийняти $a_{k+1} = a_k, b_{k+1} = d_k$ і $c_{k+1} = a_{k+1} + b_{k+1} - c_k, d_{k+1} = c_k$. Перейти до кроку 6. Якщо $f(c_k) > f(d_k)$, прийняти $a_{k+1} = c_k, b_{k+1} = b_k$ і $c_{k+1} = d_k, d_{k+1} = a_{k+1} + b_{k+1} - d_k$.

Крок 6. Обчислити $\Delta = |a_{k+1} - b_{k+1}|$ і перевірити умова закінчення пошуку. Якщо $\Delta \le \alpha$, процес пошуку припиняється і $x^* \in [a_{k+1} + b_{k+1}]$. Якості наближеного рішення приймають середину останнього інтервалу $x^* \cong \frac{a_{k+1} + b_{k+1}}{2}$. Якщо $\Delta > \alpha$, прийняти $k = k + 1$ і перейти до етапу 4.

В даному програмному засобі на базі пакету MATLAB вхідні параметри залишаються не змінні, видозмінюється сам метод оптимізації розглянемо його детально:

```
% end
a1=x1Min; // присвоюємо a1 значення початку відрізка x1
b1=x1Max; // присвоюємо b1 значення кінця відрізка x1
i1=1;
while 1
   x11=b1-(b1-a1)/tau;// обчислюємо значення x1 для першого стану
   x12=a1+(b1-a1)/tau; // обчислюємо значення x2 для першого стану
    a2=x2Min; // присвоюємо a2 значення початку відрізка x2
   b2=x2Max; // присвоюємо b2 значення початку відрізка x2
   i2=1;
   while 1
      x21=b2-(b2-a2)/tau; // обчислюємо значення x1 для другого стану
      x22=a2+(b2-a2)/tau; // обчислюємо значення x2 для другого стану
       g=zeros(M); // заповнюємо матрицю g нулями
      for m=1:M // цикл переробу усіх значень u для першого стану
         g(m,1)=G(x11,x21,u(m)); // заповнюємо перший стопець матрицю значеннями G
         g(m,2)=u(m); // заповнюємо другий стовпець значеннями u
```

```
        end
        index=minimum(g(:,1)); // знаходимо мінімальне матриці g
        f11 = ResultPoint2D(x11,x21,g(index,2),g(index,1)); // записуємо результат для першого стану
         g=zeros(M); // заповнюємо матрицю g нулями
        for m=1:M// цикл переробу усіх значень u для другого стану
           g(m,1)=G(x11,x22,u(m)); // заповнюємо перший стопець матрицю значеннями  G
           g(m,2)=u(m); // заповнюємо другий стовпець значеннями u
        end
        index=minimum(g(:,1)); // знаходимо мінімальне матриці g
        f12 = ResultPoint2D(x11,x22,g(index,2),g(index,1)); // записуємо результат для першого стану в f12
        if f11.S >= f12.S // умова порівняння по S функцій f11 та f12
           a2=x21;// якщо умова виконується a2 присвоюємо значення x21
           f1=f12; // a f1 присвоюємо f12
        else
           b2=x22;// не виконується умова b2 присвоюємо x22
           f1=f11;// та присвоюємо f1 значення f11
        end
        if abs(b2-a2)/2 < epsilon // умова закінчення обчислень
           break
        end
       i2=i2+1;
      end
       a2=x2Min; // присвоюємо a2 значення x2Min
      b2=x2Max; // присвоюємо b2 значення x2Max
        x21=b2-(b2-a2)/tau; // знаходження значення x21
         x22=a2+(b2-a2)/tau;// знаходження значення  x22
          g=zeros(M);// заповнення матриці g нулями
```

```
for m=1:M// цикл перебору усіх u та знаходження відповідних їм G
  g(m,1)=G(x12,x21,u(m)); // заповнюємо перший стопець матрицю значеннями G
  g(m,2)=u(m); // заповнюємо другий стовпь значеннями u
end
index=minimum(g(:,1)); // знаходимо мінімальне матриці g
f11 = ResultPoint2D(x12,x21,g(index,2),g(index,1)); // записуємо результат для в f11
g=zeros(M); // заповнюємо матрицю g нулями
for m=1:M// цикл переробу усіх значень u для другого стану
  g(m,1)=G(x12,x22,u(m)); //заповнюємо перший стопець матрицю значеннями G
  g(m,2)=u(m); // заповнюємо другий стовпець значеннями u
end
index=minimum(g(:,1)); // знаходимо мінімальне матриці g
f12 = ResultPoint2D(x12,x22,g(index,2),g(index,1)); // записуємо результат для в f12
if f11.S >= f12.S // // умова порівняння по S функцій f11 та f12
  a2=x21; // якщо умова виконується a2 присвоюємо значення x21
  f2=f12; // a f22 присвоюємо f12
else
  b2=x22; // не виконується умова b2 присвоюємо x22
  f2=f11; // a f2 присвоюємо f11
end
if abs(b2-a2)/2 < epsilon // умова закінчення обчислень
  break
end
 end
if f1.S >= f2.S// умова порівняння по S функцій f11 та f2
  a1=f1.X1;// присвоюємо a1 значення x1 функціїf1
```

```
        result(N,i1)=f2; // результат запишемо в f2
      else
        b1=f2.X1; // якщо умова не виконується b1 запишемо x1 функції f2
        result(N,i1)=f1;// результат запишемо в f1
      end
      if abs(b1-a1)/2 < epsilon epsilon // умова закінчення обчислень
        break
      end
    end
```

Розглянемо в даному програмному засобі на базі пакету MATLAB прямий прохід

```
% priamoy
un=result0.U; // початкове значення u
xn1=Fi1(result0.X1,result0.X2,un); // знаходимо значення x1
xn2=Fi2(result0.X1,result0.X2,un);// знаходимо значення x2
[index1]=findclosestkvantum2d([xn1 xn2],result(1,:)); // знаходимо найближче значення об’єкта дістаємо його з пам’яті та записуємо в un
un=result(1,index1).U; /
for n=2:N // цикл знаходження керування в момент часу від 2 до N
  xn1=Fi1(xn1,xn2,un);// знаходимо значення x1
  xn2=Fi2(xn1,xn2,un); знаходимо значення x1
  [index1]=findclosestkvantum2d([xn1 xn2],result(n,:)); // знаходження найближчого значення
  un=result(n,index1).U;// запис його в un
end
```

Детальні в можна розглянути програмний засіб у додатку К. Виконання:

```
Direct walk
n=0; x1=5; x2=-5; U=-0.010101
n=1; x1=0.00003932499; x2=-1.41626123751; U=-0.0101010101
n=2; x1=-0.79016994375; x2=-0.21626123751; U=0.0101010101
n=3; x1=-0.6203932499; x2=0.1626123751; U=-0.0101010101
```

n=4; x1=-0.19016994375; x2=0.11626123751; U=-0.0101010101

n=5; x1=-0.01626123751; x2=0.11626123751; U=0.0101010101

n=6; x1=0.01626123751; x2=0.00000000000; U=0.0101010101

n=7; x1=0.00000000000; x2=0.00000000000; U=0.0101010101

n=8; x1=0.00000000000; x2=0.00000000000; U=-0.0101010101

n=9; x1=0.00000000000; x2=0.00000000000; U=-0.0101010101

n=10; x1=0.1000000000; x2=0.00000000000; U=-0.0101010101

******* End *******

4.8 Розроблення програмного засобу на базі пакету MATLAB з використанням методу покордінатного спуску для об’єкту другого порядку

Алгоритм методу покоординатного спуску :

Крок 1. Спуск відбувається по координаті x_1. Фіксуємо значення по координат $x_2=x_2^k,\ldots,x_n=x_n^k$ та вирішуємо задачу мінімізації функції одної зміної(можна використовувати метод золотого перетину).

Крок 2. Проводимо спуск по координаті x_2. Фіксуємо значення координат $x_1=x_1^{k+1}$, $x_3=x_3^k,\ldots,x_n = x_n^k$ та вирішуємо задачу одновимірної мінімізації.

Крок 3. Аналогічно відбувається спуск по решті координат (якщо порядок об’єкту більший ніж два).

Крок 4. На останьому n- кроці координату x_n^{k+1} процес продовжується до отримання найменшого значення цільової функції.

Крок 5. Перевіряємо умову закінцення виконання алгоритму. Строго кажучи, момент закінчення пошуку настане тоді, коли рух з отриманої точки, при виборі будь-якого кроку, призводить до зростання значення цільової функції. Якщо мінімум функції досягається всередині розглядуваної області, то в цій точці градієнт дорівнює нулю, що також може служити сигналом про закінчення процесу оптимізації.

У даному програмному засобі на базі пакету MATLAB теж застосовується Generic-файл. А метод покординатного спуску для двовимірного об’єкту розглянемо детально:

```
function result = coordinate_descent(priv,xmin,xmax,fx) // функція покоординатного спуску
    eps=0.008; // задаємо ε
    result=[]; // ініціалізуємо змінну під результат
    x1p=xmin(1); // присвоєння x1p мінімальне значення першого стану
    x2p=xmin(2);// присвоєння x2p мінімальне значення другого стану
    k=true; // визначаємо змінну «стоп-індикатор»
    fmin=0;
    while true
      if k == false // якщо стоп індикатор, тоді
        break; // припиняємо обробку
      end
      i=1;
      [x1p,r]=coordinate_descent_min(x2p,x2p,xmax(2),i,priv,fx,result); // знаходимо мінімум по x2p
      f=func2D([x1p x2p],priv,fx); //
      fmin1=f.S; // записуємо розраховану S
      if abs(fmin1-fmin)<eps // якщо різниця мінімумів досягла значення eps, тоді
        break; // припиняємо обробку
      end
      result=[result,r]; //
      fmin=fmin1;//
      [x2p,r]=coordinate_descent_min(x1p,x1p,xmax(1),i,priv,fx,result); // знаходимо мінімум по x1p
      f=func2D([x1p x2p],priv,fx);
      fmin1=f.S;
      if abs(fmin1-fmin)<eps
```

```
            break;
        end
        result=[result,r];
        fmin=fmin1;
    end
end
```

Та функція знаходження по координатного мінімум:

```
function [Xmin,result] = coordinate_descent_min(x,a,b,i,priv,fx,result)
    eps=0.00008; // задаємо епсілон
    t=(3-sqrt(5))/2;
    c=a+t*(b-a);
    d=a+b-c;
    while true
        if i==1 // якщо не парний крок
            f1=func2D([c x],priv,fx); // розрахунок функції для змінних c та x
            k1=f1.S;
            f2=func2D([d x],priv,fx); // розрахунок функції для змінних d та x
            k2=f2.S;
        else
            if i==2 // якщо парний крок
                f1=func2D([x c],priv,fx); // розрахунок функції для змінних x та c
                k1=f1.S;
                f2=func2D([x d],priv,fx);
                k2=f2.S;
            end
        end
        if k1 >= k2
            a=c;
            c=d;
            d=d-t*(b-a);
```

```
        else
            b=d;
            d=c;
            c=a+b-d;
        end
         if abs(b-a)<eps // якщо досягнуто межі точності, тоді
            break; // кінець
        end

        if i==1
            f=func2D([(a+b)/2 x],priv,fx);
        else
            if i==2
                f=func2D([x (a+b)/2],priv,fx);
            end
        end
        result=[result,f]; // додаємо розрахунки до результату
    end
        Xmin = (a+b)/2; // записуємо знайдений мінімум
end
```

Виконавши програмний засіб на базі пакету MATLAB оптимаємо :

Direct walk

n=0; x1=-5.000000000000000;x2=5.000000000000000;U=-0.034482758620690

n=1; x1=-1.7436872690296679;x2=0.037115936507640;U=-0.034482758620690

n=2; x1=-0.231929354378314;x2=-0.978713763747792;U=-0.034482758620690

n=3; x1=0.003564180986508;x2=-0.466484782133440;U=-0.034482758620690

n=4; x1=0.011818663791810;x2=-0.137115936507640;U=-0.034482758620690

n=5; x1=0.015733046195595;x2=-0.094207022084515;U=-0.034482758620690

n=6; x1=0.017845354839514;x2=0.000000000000000;U=-0.034482758620690

n=7; x1=0.000000000000000;x2=0.000000000000000;U=-0.034482758620690

n=8; x1=0.000000000000000;x2=0.00000000000000;U=0.034482758620690

n=9; x1=0.00000000000000; x2=-0.00000000000000;U=-0.034482758620690

n=10; x1=0.000000000000000;x2=-0.0000000000000;U=-0.034482758620690

******* End *******

4.9 Розроблення моделі оптимальної системи та дослідження роботи алгоритмів в пакеті Matlab/Simulink

В даному розділі розробимо модель системи, перевірено результати виконання програмних засобів на базі пакету MATLAB, досліджено яке обмеження потрібно наложити на стани.

Спочатку проведемо усі дослідження з об’єктом першого порядку (4.2.1). Від рівнянь станів перейдемо до матричної моделі (4.9.1), час дискретизації T=0,1с.

$$\mathrm{A} = [1], B = [1], \mathrm{C} = [1], D = [0] \qquad (4.9.1)$$

Побудуємо модель системи першого порядку (рисунок 4.8.1). В блок From Workspace подається керування, яке отримане в процесі виконання програмного засобу на базі пакету MATLAB, у вигляді матриці залежності часу дискретизації від керування. В блоці Discrete State-Space зображений об’єкт (4.9.1), в Scope – зображений рух стану об’єкта та керування.

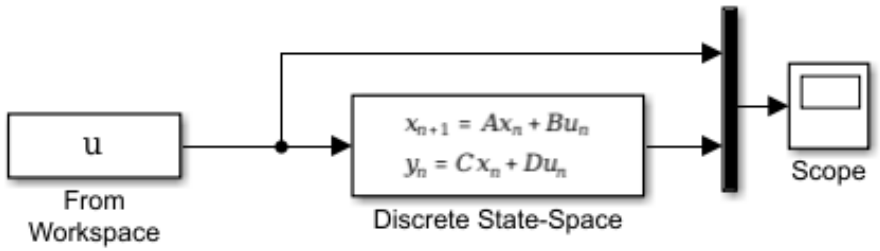


Рисунок 4.9.1 – Модель оптимальної системи першого порядку

Далі керування оптимане в методі рівномірного поділу записуємо в матрицю, подаємо на об’єкт на спостерігаємо результат на рисунку 4.9.2. Об’єкт зображений пунктирною лінією, а керування суцільною.

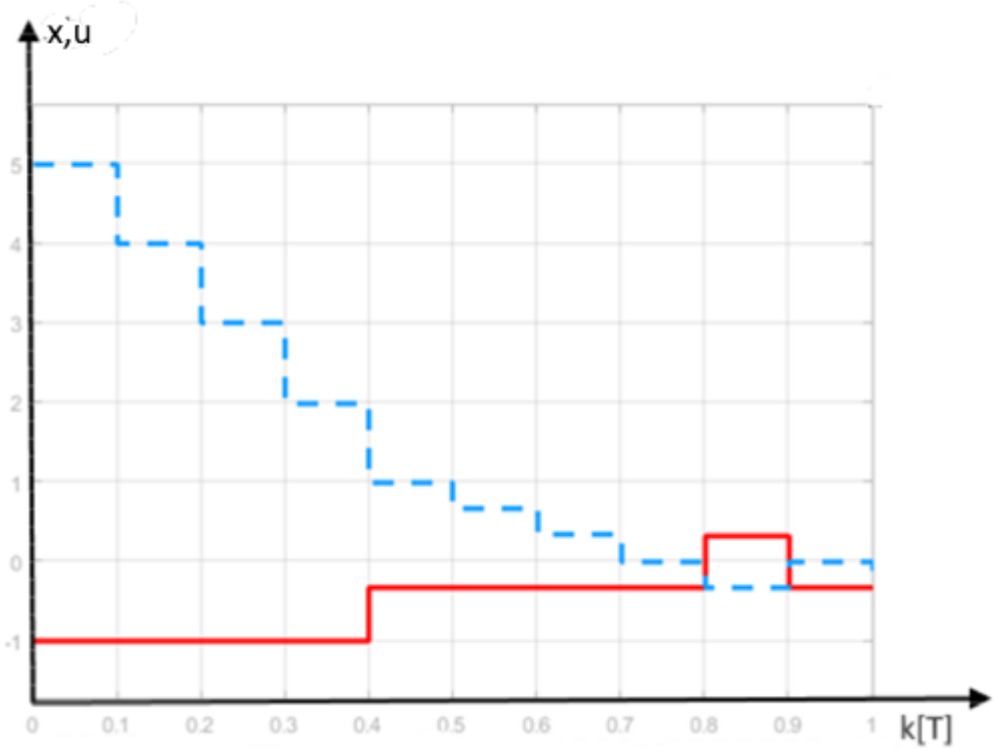


Рисунок 4.9.2 – Результати моделювання МРП

З рисунку 4.9.2 можна спостерігати, що керування подане на об’єкт здійснює рух об’єкту відповідний до оптиманих результатів у виконанні програмного засобу на базі пакету MATLAB.

Далі використаємо керування отримане від МПП та будемо спостерігати результати на рисунку 4.9.3. Об’єкт зображений пунктирною лінією, а керування суцільною.

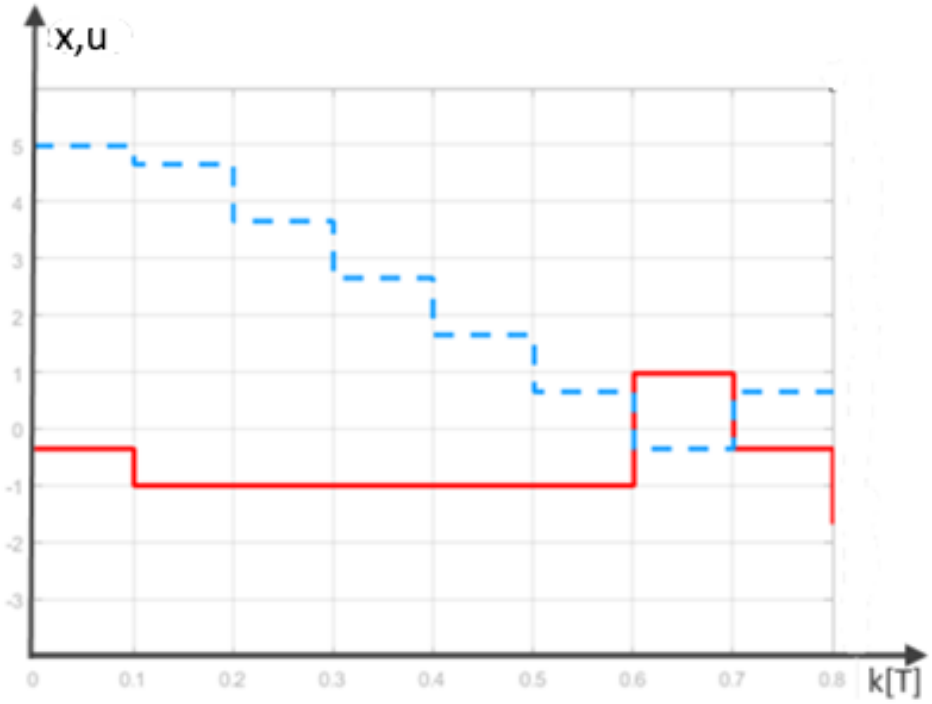


Рисунок 4.9.3 – Результати моделювання МПП

З рисунку 4.9.3 можна спостерігати, результати відповідають очікуваним.

Далі використаємо керування отримане від МЗП та будемо спостерігати результати на рисунку 4.9.4. Об’єкт зображений пунктирною лінією, а керування суцільною.

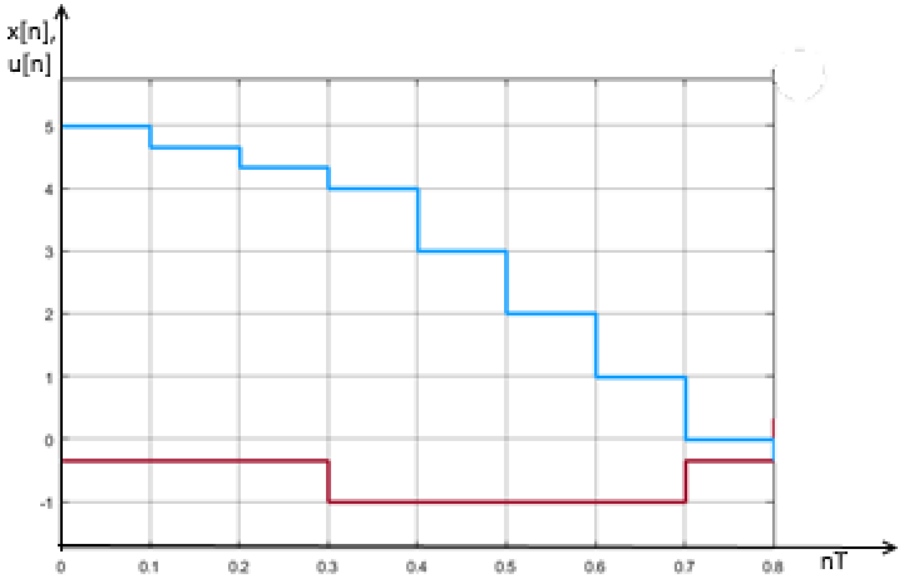


Рисунок 4.9.4 – Результати моделювання МЗП

Отож з результатів моделювання видно, що метод динамічного програмування з методами оптимізації працюють вірно. Ефективність роботи розглянемо в наступному розділі.

Далі змодулюємо усі керування різних методів оптимізації. Визначимо, яке обмеження потрібно накласти на стани об’єкту (4.5.1). Від рівнянь станів перейдемо до матричної моделі (4.9.2), час дискретизації T=0,1с.

$$A = \begin{bmatrix} 0{,}1764 & -0{,}07686 \\ 0{,}5 & 0{,}5 \end{bmatrix}; B = \begin{bmatrix} 0{,}25 \\ 0 \end{bmatrix}; C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; D = [0; 0] \qquad (4.8.2)$$

Побудуємо модель системи другого порядку (рисунок 4.9.5). В блок From Workspace подається керування, яке отримане в процесі виконання програмного засобу на базі пакету MATLAB, у вигляді матриці залежності часу дискретизації від керування. В блоці Discrete State-Space зображений об’єкт (4.9.2), в Scope — зображений рух стану об’єкта та керування, та в моделі можна спостерігати траєкторію об’єкту.

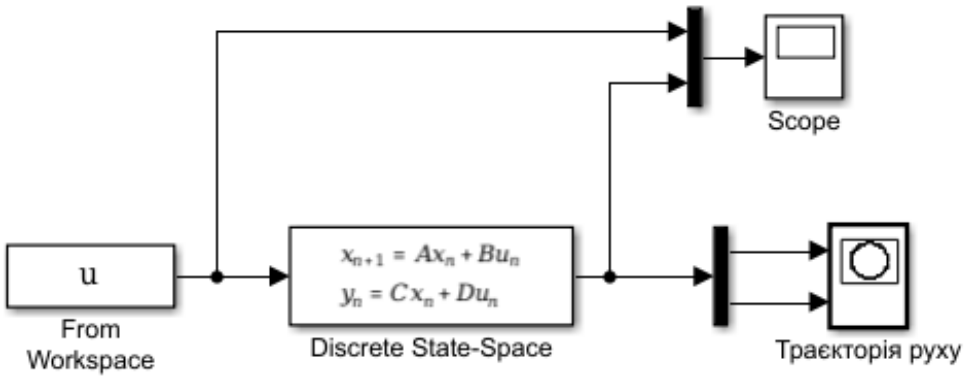


Рисунок 4.8.5 – Модель об’єкту другого порядку

Далі керування оптимане в методі рівномірного поділу записуємо в матрицю, та подамо на об’єкт, спостерігаємо результат на рисунку 4.9.6. Суцільною лінією зображений x_2, шрих лінією — x_1. Керування представлено на рисунку 4.9.7. А траєкторію руху будемо спостерігати на рисунку 4.9.8.

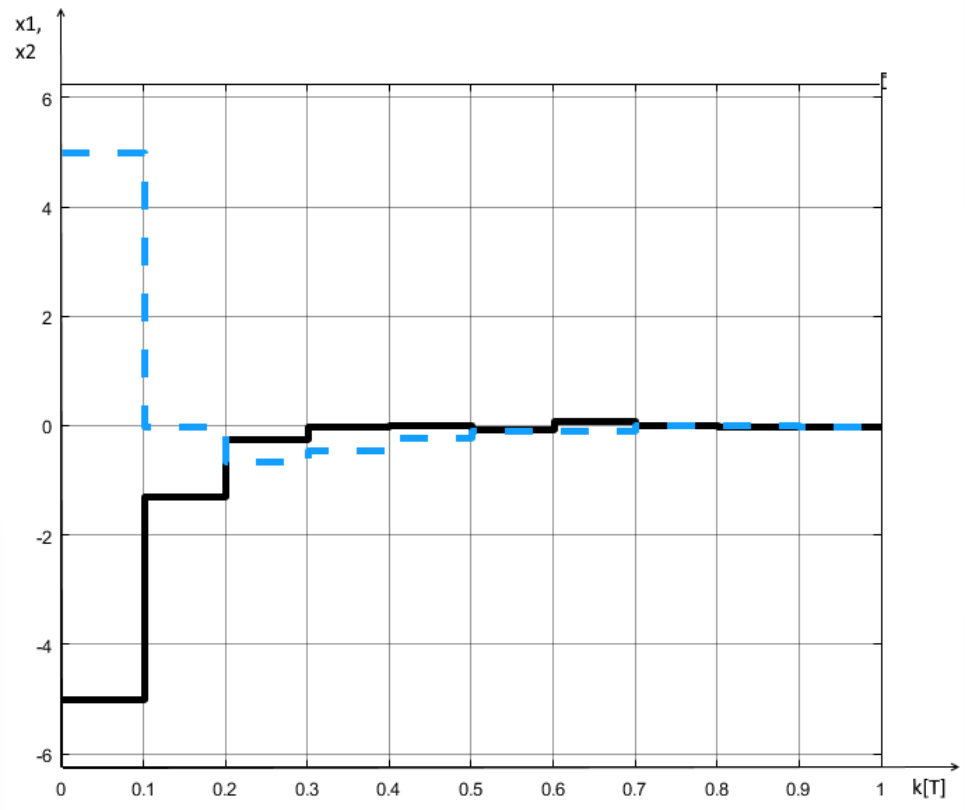


Рисунок 4.9.6 – Результати моделювання для МРП

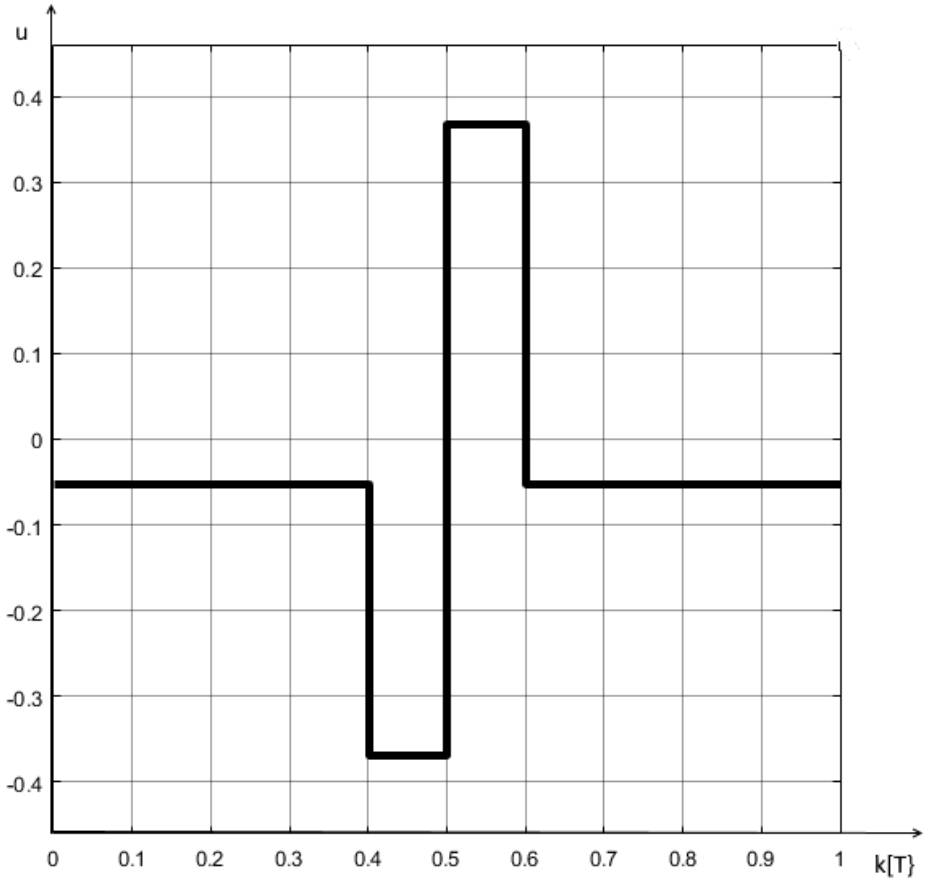


Рисунок 4.9.7 – Керування для МРП

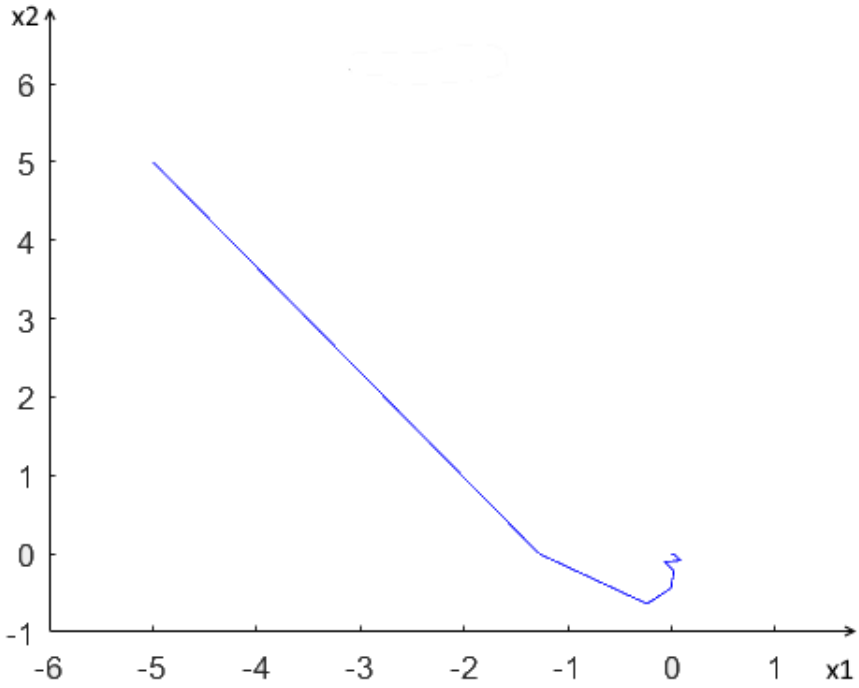


Рисунок 4.9.8 – Траєкторія руху МДП з МРП

Далі змоделюємо систему з МПП та будемо спостерігати результати на рисунку 4.9.9. Суцільною лінією зображений x_2, шрих лінією — x_1. Керування представлено на рисунку 4.9.10. А траєкторію руху будемо спостерігати на рисунку 4.9.11.

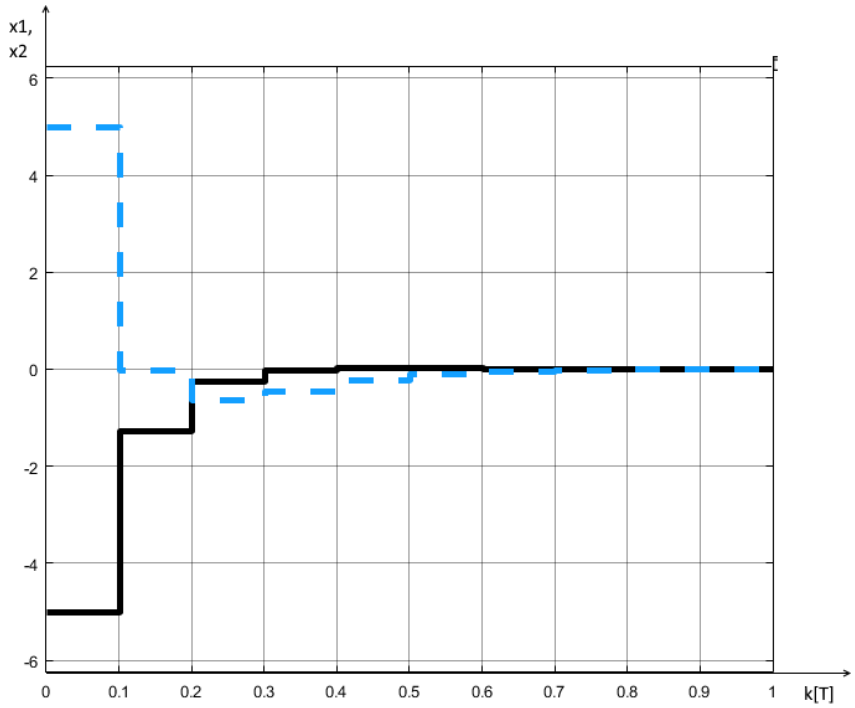


Рисунок 4.9.9 – Результати моделювання для МПП

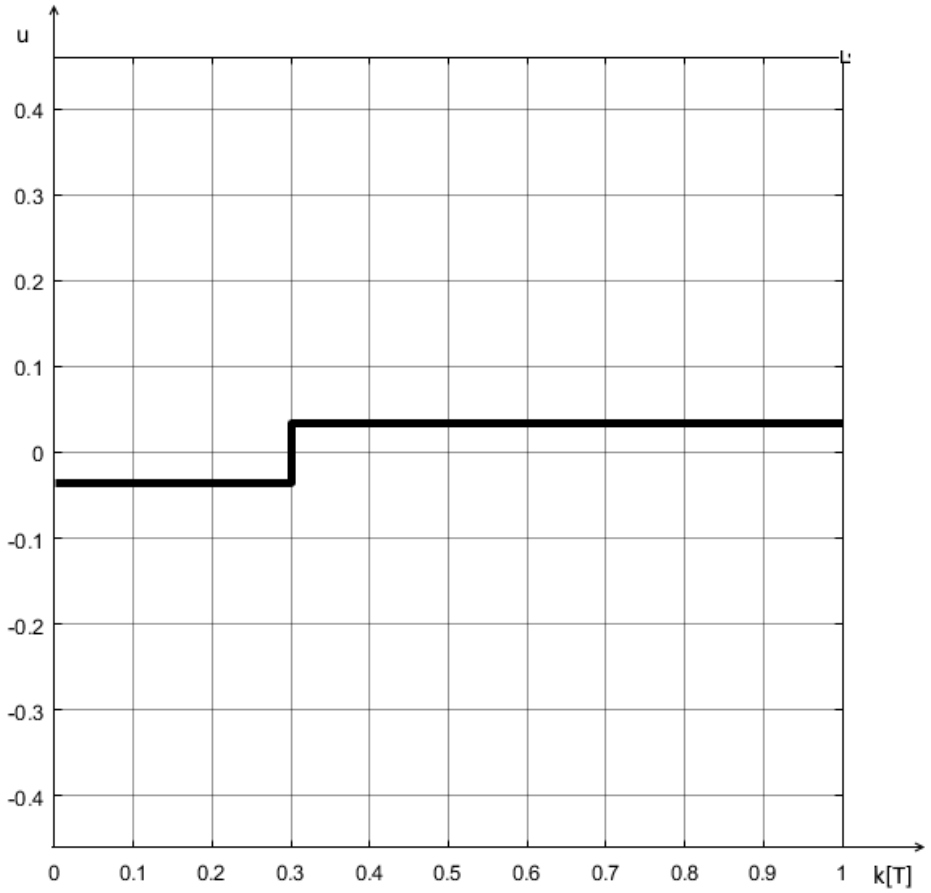


Рисунок 4.9.10 – Керування МПП

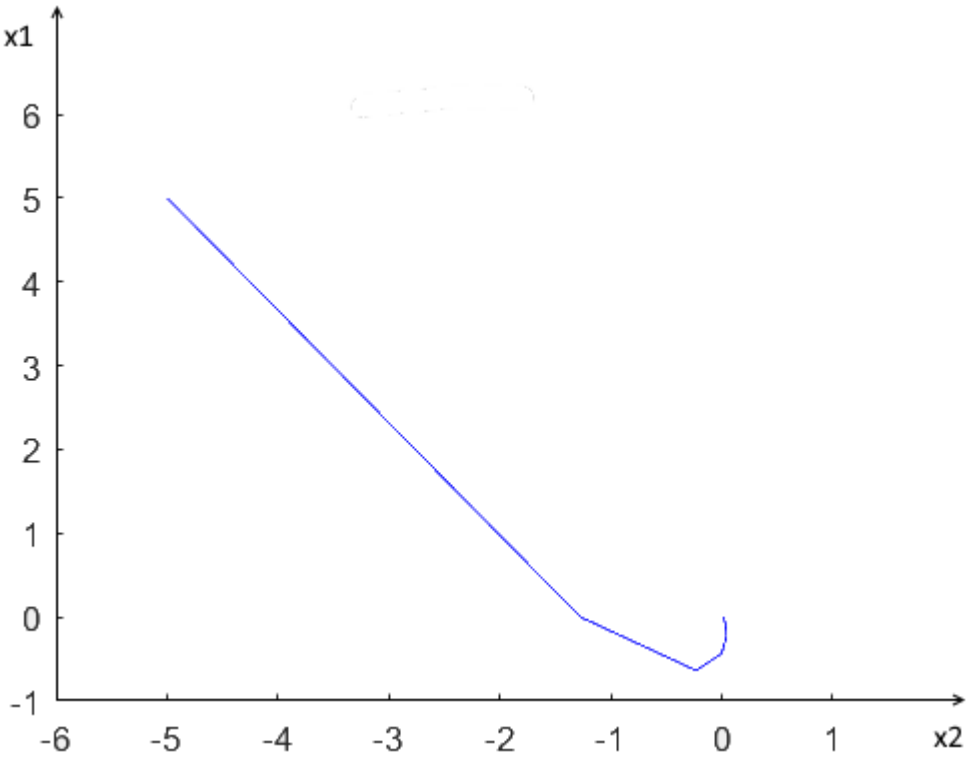


Рисунок 4.9.11 – Траєкторія руху МДП з МПП

Далі змоделюємо систему з МЗП та будемо спостерігати результати на рисунку 4.9.12. Суцільною лінією зображений x_2, шрих лінією — x_1. Керування представлено на рисунку 4.9.13. А траєкторію руху будемо спостерігати на рисунку 4.9.14.

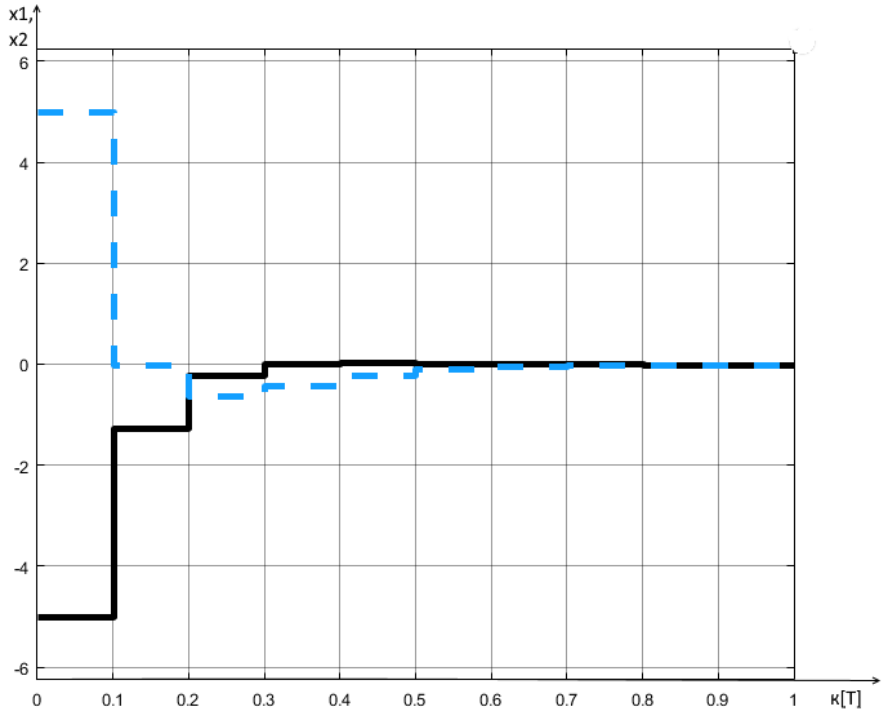


Рисунок 4.9.12 – Результати моделювання для МЗП

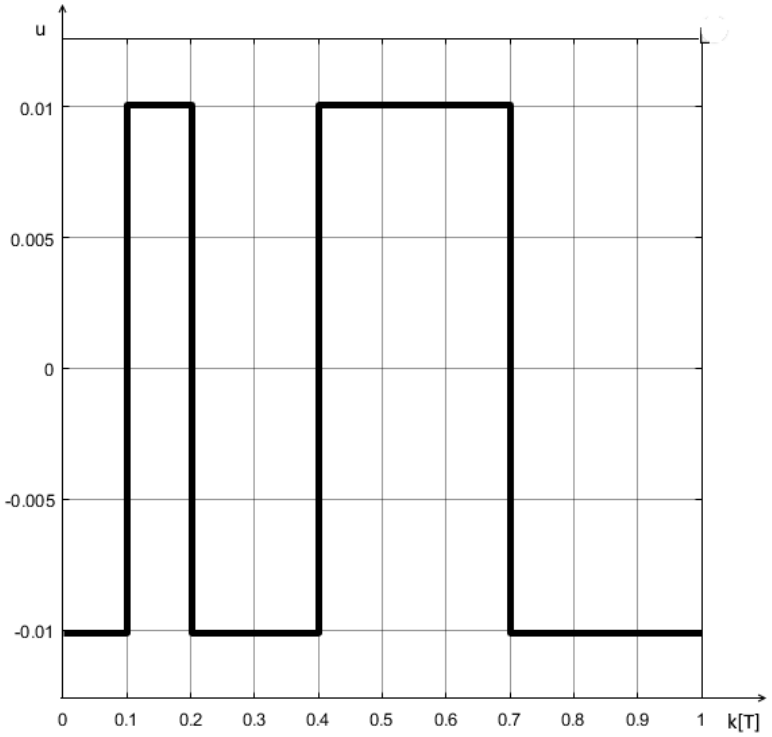


Рисунок 4.9.13 – Керування МЗП

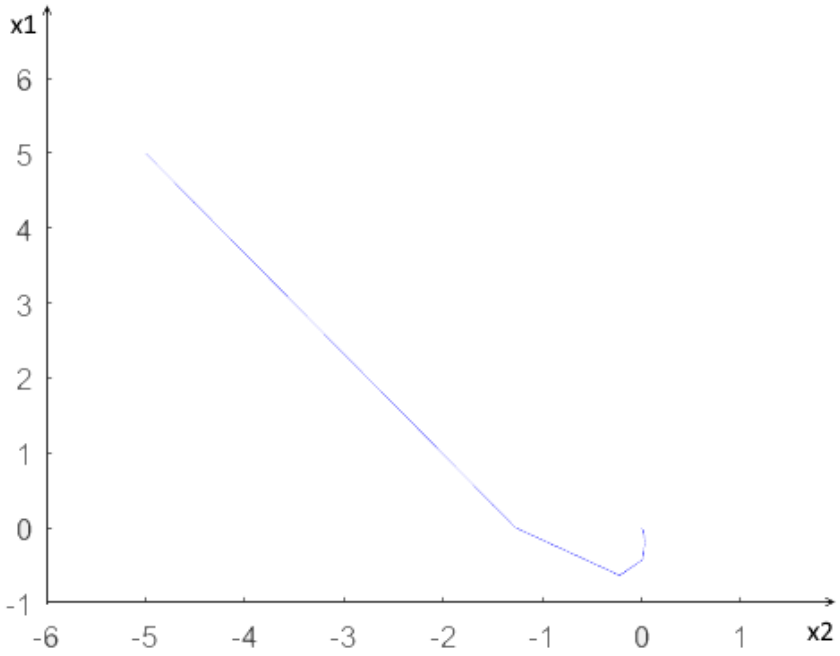


Рисунок 4.9.14 – Траєкторія руху МДП з МПП

Далі змоделюємо систему з МПС та будемо спостерігати результати на рисунку 4.9.15. Суцільною лінією зображений x_2, шрих лінією – x_1. Керування представлено на рисунку 4.9.16. А траєкторію руху будемо спостерігати на рисунку 4.9.17.

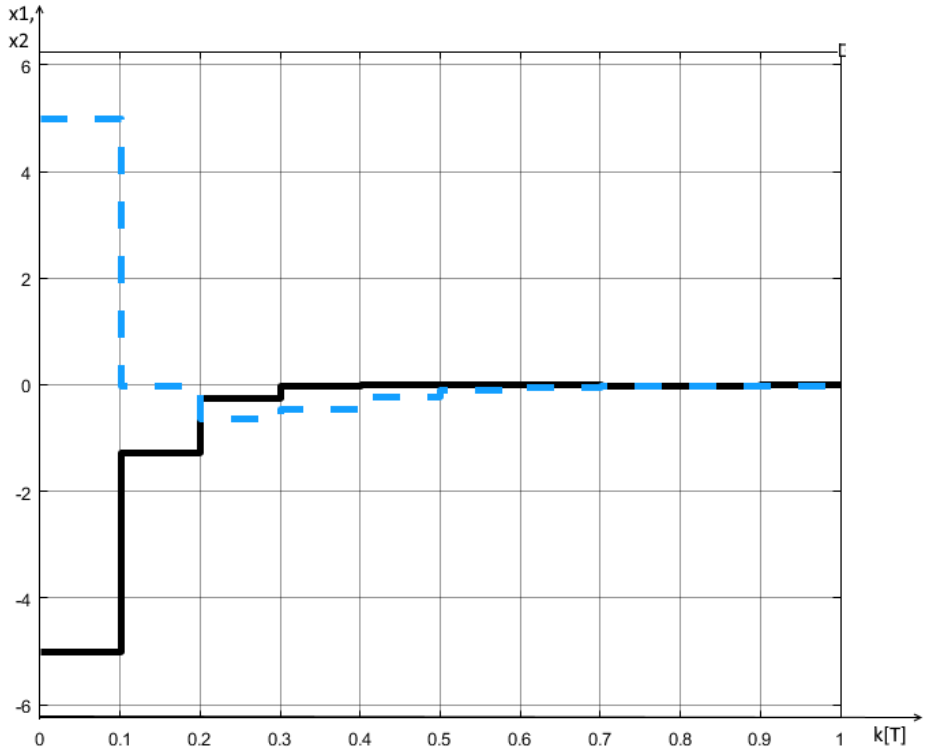


Рисунок 4.9.15 – Результати моделювання для МПС

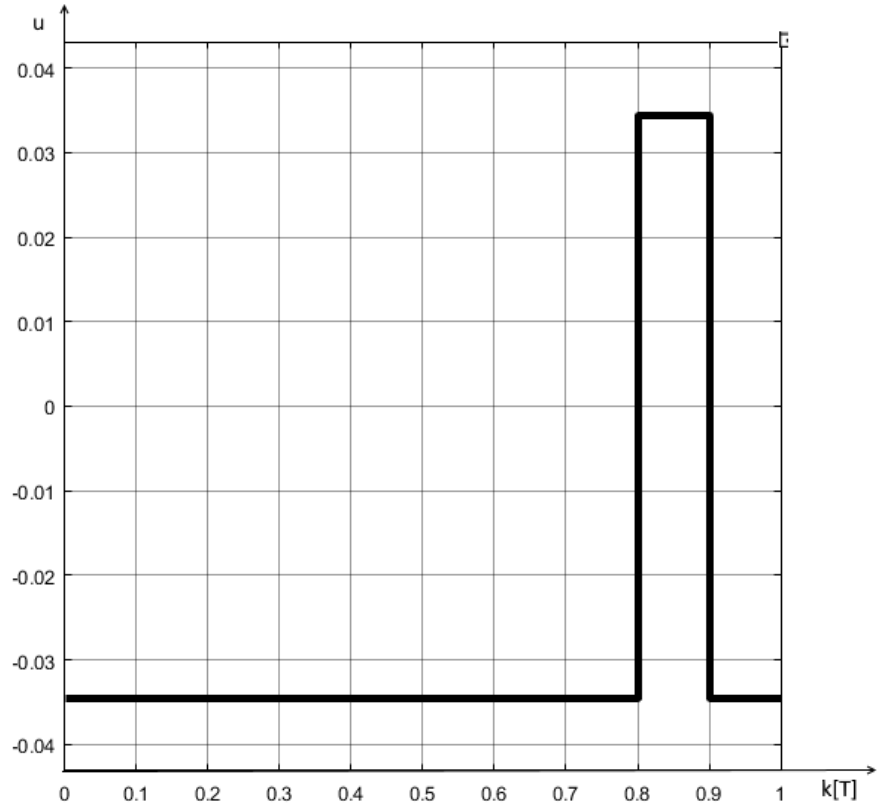


Рисунок 4.9.16 – Керування МПС

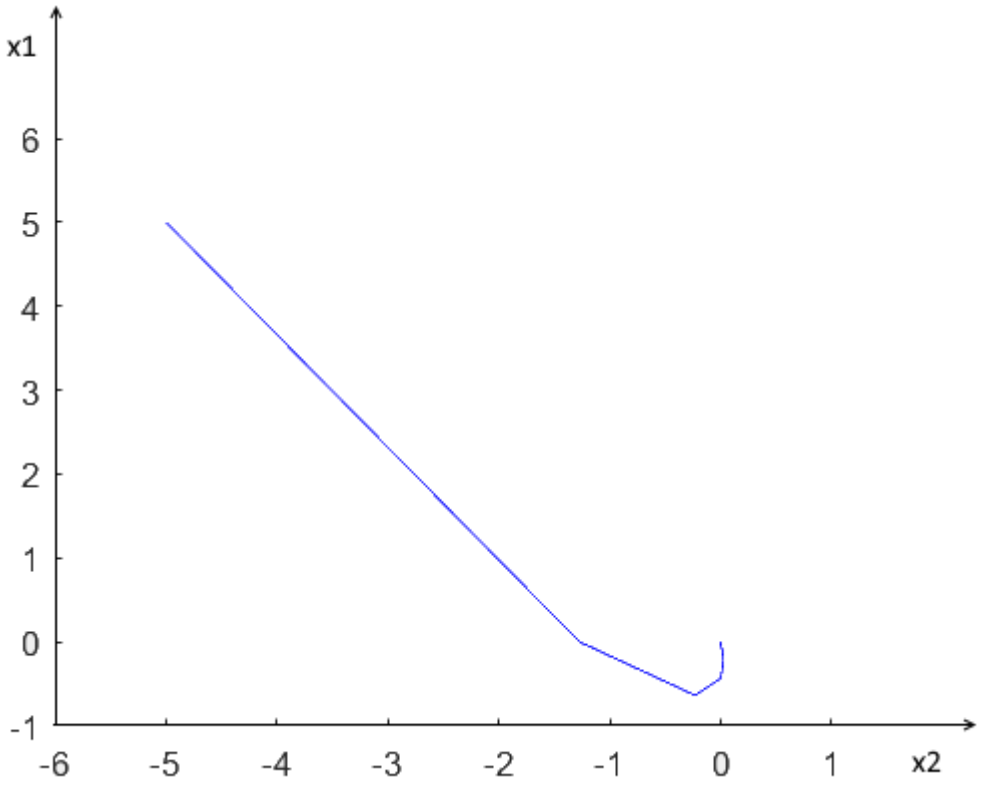


Рисунок 4.9.17 – Траєкторія руху МДП з МПС

Отже з моделювання видно, що алгоритми працюють вірно. А який з них найкращий дізнаємось в наступному розділі.

4.10 Порівняння методів оптимізації для методу динамічного програва в задачі дискретного програмування

В даному розділі зробимо порівняння алгоритмів оптимізації в методі динамічного програмування для першого та другого порядку за такими показниками: кількість ітерацій, час виконання, швидкодія (таблиця 4.10.1). Швидкодія – це відношення кількості ітерацій до часу виконання програмного засобу на базі пакету MATLAB.

Таблиця 4.10.1 – Якісні показники алгоритмів оптимізації для об’єкту першого порядку

	Алгоритм рівномірного ділення	Алгоритм половинного ділення	Алгоритм золотого перетину
Час виконання, с	5,14	0,14	0,14
Кількість ітерацій	4909	78	71
Швидкодія ітр/с	955	557	597

Далі складемо порівняльну таблицю 4.10.2 для об’єкту друго порядку за тими ж характеристиками.

Таблиця 4.10.2 – Якісні показники алгоритмів оптимізації для об’єкту другого порядку

	Алгоритм рівномірного ділення	Алгоритм половинного ділення	Алгоритм золотого перетину	Алгоритм покоординатного спуску
Час виконання,с	600,4	0,41	0,39	0,36
Кількість ітерацій	19909	152	149	130
Швидкодія, ітр/с	33	370	382	361

Отож з таблиці 4.10.1 видно, що найшвидший алгоритм це метод золотого перетину для задачі динамічного програмування об’єкту першого порядку. Найефективніший алгоритм половинного ділення. А у всіх позиціях програє алгоритм рівномірного поділу. Але він є найпростіший у реалізації.

З таблиці 4.10.2 можна зробити висновок, що найшвидший алгоритм покоординатного спсуку, він виграє у всіх позиціях. Неефективним є алгоритм рівномірного поділу. Алгоритми золотого і половинного перетину показують непогані результати. Але вони краще працюють для одновимірних об’єктів.

5 РОЗРОБЛЕННЯ СТАРАП ПРОЕКТУ

5.1 Опис ідеї проекту

Метою розділу є розроблення програмного засобу на базі пакету MATLAB для багатовимірного об’єкту методу динамічного програмування з оптимізацією станів об’єкту алгоритмом покоординатного спуску. Розглянемо зміст ідеї, можливі напрямки застосування, осносні переваги, які зможе отримати користувач предсталено у таблиці 5.1.1

Таблиця 5.1.1. Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	1. В виробництві	Зменшення часу виконання, роботу з n-об’єктами, швидкодія обчислень.
	2.В транспортній сфері	Зменшення часу на розроблення маршрутів, роботу з n-об’єктами, швидкодія обчислень.
	3. В економічні сфері	Роботу з n-об’єктами, оптимізація бізнес процесів.
	4.В телекомунікаційній сфері	Зменшення часу виконання, роботу з n-об’єктами, швидкодію
	5.В інформаційних технологіях	Зменшення часу виконання алгоритмів, роботу з n-об’єктами, швидкодію

Даний алгоритм відрізняється від класичної задачі, тим що використовується багатовимірний метод оптимізації МПС. В результаті робота з багатовимірними об’єктами стає простішою, час виконання зменшується, зменшуються обчислювальні потужності, швидкодія та час виконання.

На ринку потенційні та конкретні конкуренти відсутні, присутні тільки товари замінники – перший конкурент -класний алгоритм, тобто МДП з використанням МПС. Другий алгоритм – МДП з використанням МЗП. Терій конкурент – МДП з МЗП. Мій проект МДП з оптимізацією МПС.

Таблиця 5.1.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
	Мій проект	Конкурент1	Конкурент2	Конкурент3			

Продовження таблиці 5.1.2

Технічна	- робота з багатовимірними об’єктами - швидкодія - досконалий пошук оптимального рішення	Великі витрати ресурсів на обробку потрібної задачі	Для роботи з багатовимірними об’єктами потрібні великі обчислювальні потужності	Для роботи з багатовимірними об’єктами потрібні великі обчислювальні потужності		- досконалий пошук оптимального рішення -затрати часу на реалізацію	-робота з багатовимірним об’єктами - швидкодія
Економічна	27 тис. – 40 тис. грн	25тис. грн	35 тис. грн	45 тис.грн		Середня ціна на ринку	
Надійність	Не обмежений термін діїї	Не обмежений термін діїї	Не обмежений термін діїї	Не обмежений термін діїї			Довговічність не менша ніж у конкурентів

5.2 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (табл. 5.2.1):

Таблиця 5.2.1. Технологічна здійсненність ідеї проекту

Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
		Середовище MATLAB, високорівневою мовою	Технологія доступна
		Середовище розробки Eclipse, мовою програмування Java	Технологія доступна
		Середовище розробки MS Visual studio, мовою програмування C#	Технологія доступна
Обрана технологія реалізації ідеї проекту: Технологією розроблення було вибрано середовище MATLAB. В даному середовищі можливе не тільки реалізація продукту, а й моделювання потрібних умов для перевірки правильності роботи алгоритму			

За результатами аналізу таблиці робиться висновок щодо можливості технологічної реалізації проекту: так чи ні, а також технологічного шляху, яким це доцільно зробити (з поміж названих технологій обираються такі, що доступні авторам проекту та є наявними на ринку).

5.3 Аналіз ринкових можливостей запуску стартап-проекту

1) Спочатку проводиться аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 5.3.1).

Таблиця 5.3.1. Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	10
2	Загальний обсяг продаж, грн/ум.од	10000 за рік
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Сертифікація
6	Середня норма рентабельності в галузі (або по ринку), %	60

Ринок є привабливий для входу та потребує новітніх алгоритмів пошуку найкращих, оптимальних методів вирішення задач.

3) Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (таблиці 5.3.2).

Таблиця 3.3.2 Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Пошук оптимального руху об’єкту за малий проміжок часу	Потенційна група клієнтів є підприємці, бізнесмети	-потреба в продуті оптимізації -можливе застосування у різних сферах -допустима цціна	-швидко отриматии результат -дешево отримати результат -якісне надання, ефективне надання послуги

4) Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (таблиці 5.3.3 та 5.3.4). Фактори в таблиці подавати в порядку зменшення значущості.

Таблиця 5.3.3 Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Збільшення ціни на розроблення	Збільшення ціни заробітньої праці розробників	Зміна цінової політики
2	З’являються конкуренти	На ринку утворюються конкуренти з якіснішими продуктами	-Реклама -Удосконалення продукту

Таблиця 5.3.4. Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Попит на товар	Попит на оптимальні системи пошуку.	-Пришвидшення розробки -збільшення обсягів виробнитва
2	Зменшення ціни на розроблення	Зменшення ціни заробітньої прати працівників	-Пришвидшення розробки -Вихід на нові ринки збуту

5) Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку (таблиці 5.3.5).

Таблиця 5.3.5. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1.Вказати тип конкуренції - олігополія	Домінує мала кількість фірм	-Пришвидшення розроблення унікального продукту -Впровадження реклами
2. За рівнем конкурентної боротьби національний	Боротьба ведеться на національному ринку	Вихід на всесвітній ринок
3. За галузевою ознакою - міжгалузева	Боротьба між товаритсвами різних галузей	Розроблення універсального додатку, розроблення додаткових доповнень для конкреної галузі

Продовження таблиці 5.3.5.

4. Конкуренція за видами товарів: - товарно-видова	Різновиди однієї категорії товару, які здатні задовольнити конкретне бажання покупця	Відсутня
5. За характером конкурентних переваг - не цінова	Ціна не є основним фактором конкуренції	Поштовх до зменшення ціни продукту
6. За інтенсивністю - не марочна	Не прив'язаний до певної марки, може використовувати будь-де	Співпраця з різими марками

6) Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (таблиця 5.3.6).

Таблиця 5.3.6 Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Відсутні	Присутні	Відсутні	Споживачі диктують умови. Все сьогодні потребує оптимізації	Присутні, зокрема Конкурент 1,2,3.

Продовження таблиці 5.3.6

Висновки:	Прямі конкуренти на ринку відсутні.	Можливість виходу на ринок є, проте це буде дещо складно, потенційні конкуренти – Конкурент 1,2,3. На ринок можна вийти до 1 року за рахунок гарної команди спеціалістів	Постачальники відсутні, так що для розробки продукту потрібні прогармісти.	Клієнти диктують умови. Якісний, дешевий, швидкодіючий продукт.	Процесі дослідження ринку, можливий аналіз та розробка нових методів оптимізаії.

Робота на ринку можлива, тому що відсутні прямі конкуренти, а ринок потребує оптимізації у всьому. Сильними сторононами алгоритму є його унікальність та швидкість роботи для будь-якого об’єкту.

7) На основі аналізу конкуренції, проведеного в таблиця 5.3.6, а також із урахуванням характеристик ідеї проекту (табл. 5.1.2), вимог споживачів до товару (таблиці 5.3.2) та факторів маркетингового середовища (таблиці №№ 5.3.3-5.3.4) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за таблицею. 5.3.7

Таблиця 4. Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Швидкість	Алгоритм за короткий час дає потрібний результат, на відміну від аналогів

Продовження таблиці 5.3.7

2	Робота з багатовимірними об’єктами	Можливість швидкої роботи з багатовимірними об’єктами, не витрачаючи бгато часу на розробку(тому що МПс – цу дозволяє) та виконання обчислень
3	Простота розробки	Швидка розробка проекту

8) За визначеними факторами конкурентоспроможності (таблиця 5.3.7) проводиться аналіз сильних та слабких сторін стартап-проекту (таблиця 5.3.8). (С.П. – стартап проект, К.1 – Конкурент 1, К.2 – Конкурент 2, К.3 – Конкурент 3)

Таблиця 5.3.8. Порівняльний аналіз сильних та слабких сторін «назва проекту»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з … (Конкурент 1,2,3)						
			–3	-2	-1	0	1	2	3
1	Швидкість	18	К.1				К.2	К.3	С.П.
2	Робота з багатовимірними об’єктами	20				К.1К.2К.3			С.П.
3	Пошук оптимального руху об’єкту	20						К.1К2К3	С.П.

9) Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (таблиця.= 5.3.9) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 5.3.8).

Таблиця 5.3.95. SWOT- аналіз стартап-проекту

Сильні сторони: Зменшення часу на обробку данних, зменшення обчислювальних потужностей, довговічність.	Слабкі сторони:Складність алгоритму розробки, ціна.
Можливості: Попит на товар, зменьшення коштів на розроблення	Загрози:Збільшення ціни на розроблення, з’являються конкуренти

8) На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (див. таблиця 5.3.6, аналіз потенційних конкурентів).

Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (таблиця 5.3.10).

Таблиця 6.10. Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Здешевлення матеріалів(резурсів)	низька	–
2	Розроблення унікального алгоритму	висока	до півроку
3	Потреба в оптимізації	висока	до двох років
4	Запуск реклами	середня	до одного місяця

Після аналізу зазначити обрану альтернативу.

З означених альтернатив обирається та, для якої: а) отримання ресурсів є більш простим та ймовірним; б) строки реалізації – більш стислими.

5.4 Розроблення ринкової стратегії проекту

1) Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (таблиця 5.4.1).

Таблиця 7.1. Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Виробництво, яке потребує оптимізації, інформаційні технології, транспортна система	Споживачі готові сприйняти продукт. Сьогодні оптимізація потрібна усім.	Попит є в цільовому сегменті	Конкуренція не пряма. Даний метод оптимізації є унікальний і у багатьох показниках він є виграшним.	Простота входу в сегмент не складатиме значних зусиль. Продукт простий у впровадження його на ринок
Які цільові групи обрано: Виробництво та транспортна схема				

За результатами аналізу потенційних груп споживачів (сегментів) автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар, та визначають стратегію охоплення ринку:

—якщо компанія зосереджується на одному сегменті – вона обирає стратегію концентрованого маркетингу;

—якщо працює із кількома сегментами, розробляючи для них окремо програми ринкового впливу – вона використовує стратегію диференційованого маркетингу;

—якщо компанія працює із всім ринком, пропонуючи стандартизовану програму (включно із характеристиками товару/послуги) – вона використовує масовий маркетинг.

2) Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (таблиця 5.4.2).

Таблиця 5.4.28. Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
Стратегія виклику лідера	Стратегія диференційного маркетингу	Індивідуальний підхід до клієнта; краща якість, ніж у конкурентів	Стратегія диференціації передбачає надання товару важливих з точки зору споживача відмітних властивостей, які роблять товар відмін-ним від товарів можливих конкурентів.

3) Наступним кроком є вибір стратегії конкурентної поведінки (таблиця 5.4.3).

Таблиця 5.4.39. Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Так	Шукати нових та забирати споживачів у конкурентів	Копіювати характеристики не буде. За основу буде узято задачу динамічного програмуання	Стратегія лідера

4) На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (див. таблиця 5.3.2), а також в залежності від обраної базової стратегії розвитку (таблиця 5.4.2) та стратегії конкурентної поведінки (таблиця 5.4.3) розробляється стратегія позиціонування (таблиця 5.4.4). що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 5.4.4. Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Інтегрованість, висока якість, іноваціна розробка, надійність	Стратегія диференціювання	-Швидкодія -унікальність -Витрати на обчилювання	Постійне оновлення(вдосконаленн) Найкраща якість

Результатом виконання підрозділу має стати узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначатиме напрями роботи стартап-компанії на ринку.

5.5 Розроблення маркетингової програми стартап-проекту

1) Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 5.5.1 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.5.110. Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Якісна оптимізація	Точне знаходження найкоротшої траєкторії руху	У конкурентів є такий функціонал, але він не є настільки якісним
2	Швидкість	Витрача часу на обчислення мінімальна	Конкуренти програють у швидкості роботи
3	Малі затрати обчслювальної техніки	Мінімальні затрати електроенергії	Конкуренти програють у швидкості роботи
4	Робота з багатовимірними об’єктами	Можливість знаходження оптимальних рішення для будь-яких об’єктів	Конкуренти не працюють з цією задачею, але як-би виконували. Це не було ефективно.

2) Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 5.5.2). В таблиці М/Нм – монотонні/немонотонні; Вр/Тх/Тл/Е/Ор – вартісні/ технічні/ технологічні/ ергономічні/ органолептичні;

Таблиця 5.5.211. Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>І. Товар за задумом</td><td colspan="3">Можна виділити наступні вигоди від використання продукуту:
-Оптимання оптимальний пошук траєкторії руху для багатовимірних об’єктів;
- Малі затрати обчислювальних потужностей;
-Малі витрати електроенергії;</td></tr>
<tr><td rowspan="8"></td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1.Алгоритм пошуку оптимальний</td><td>М</td><td>Тх/Тл</td></tr>
<tr><td>2.Робота з багатовимірними об’єктами</td><td>М</td><td>Вр/Тх/Тл</td></tr>
<tr><td>3.Швидкість знаходження траєкторії руху об’єкту</td><td>М</td><td>Тх/Тл</td></tr>
<tr><td>4. Універсальний для систем оптимізації</td><td>Нм</td><td>Вр/Тх/Тл</td></tr>
<tr><td>5. Відсутній термін дії</td><td>М</td><td>Вр/Тл</td></tr>
<tr><td colspan="3">Якість: відповідає нормам розробки програмного забезпечення.</td></tr>
<tr><td colspan="3">Пакування: Продукт представлений на сайті у вигляді .exe – файлу. На сайті міститься:
• загальна назва продукту, власна назва;
• опис продукту;
• функції, які представлені;
• інструкція для корестування;
• контакти для зв’язку з розробником;
• підтримка для клієнтів;</td></tr>
<tr><td colspan="3">Марка: ПП «Оптимізація»</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: Товар буде запатентований, та захищений за допомогою шифрів.</td></tr>
</table>

Після формування маркетингової моделі товару слід особливо відмітити – чим саме проект буде захищено від копіювання. Захист може бути організовано за рахунок

захисту ідеї товару (захист інтелектуальної власності), або ноу-хау, чи комплексне поєднання властивостей і характеристик, закладене на другому та третьому рівнях товару.

3) Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (таблиця 5.5.3). Аналіз проводиться експертним методом.

Таблиця 5.5.312. Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Відсутні	20 тис. грн	600 тис. грн	17 тис. грн – 30 тис. грн

4) Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (таблиця 5.5.4):

— проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);

—вибір та обґрунтування оптимальної глибини каналу збуту;

—вибір та обґрунтування виду посередників.

Таблиця 5.5.4. Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Знаходження на сайті продукту, оплата продукту, відсилання на електрону пошту ключа активації.	Зберігання, доставка на електрону пошту ключа активації.	Виробни - споживач	Web-сайт

5) Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 5.5.5).

Таблиця 5.5.5. Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Дуже ретельно обирають товар, багато порівнюють	Веб-сайт, телефон	Підтримка, індивідуальний підхід, постійне оновлення	Донесення переваг до клієнтів	Оптимізація для будь-якого об’єкта за доступною ціною

Результатом пункту 5 має стати ринкова (маркетингова) програма, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей

ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого буде впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

Висновки: Було проведено розробленого програмний засіб оптимізації методу покоординатного спуску для задачі динамічного програмування. За результатами аналізу було виявлено сильні та слабкі сторони даного продукту, конкурентні позиції на ринку. Аналіз ринку та ситуації, що склалася на ньому, дозволив виявити головні тенденції подальшого розвитку продукту, зміну споживчих переваг, головні напрямки подальшої діяльності на ринку.

Також було виявлено, в яких елементах ринково-продуктової стратегії криється проблема та невідповідність поточній ринковій ситуації.

Відповідно до виявлених невідповідностей маркетингової стратегії підприємства ринковій ситуації, що склалася, а також виявлених загроз і можливостей, сильних і слабких сторін компанії, були запропоновані коригувальні дії щодо змін в ринково-продуктовій стратегії підприємства. Попит на ринку наявний, та динаміка ринку зростає. Проект конкурентно спроможний, пряма конкуренція відсутня, потрібно швидко розвивати продукт. Продукт пропонує уже нові рішення відомих задач. Проект можна розвивати та виходити на міжнародний рівень. Програмний засіб можна розвивати, удосконалювати.

ВИСНОВКИ

В дисертації було досліджено методи оптимального керування. Детально було опрацьовано метод динамічного програмування для безперервних та дискретних моделей об’єктів. Було вивчено математичні методи оптимізації, зокрема метод половинного ділення, рівномірного поділу, золотого перетину, покординатного спуску. Було розроблено алгоритми та написано програмний засіб на базі пакету MATLAB для МДП з використанням методів оптимізації для першого та другого порядків об’єків. У дисе було вперше поєднано та виконаний порівняльний аналіз методу динамічного програмування та алгоритми оптимізації: рівномірного та половиного поділу, золотого перетину для об’єктів першого порядку, а для другого ще було використано метод багатовимірної оптимізації – покоординатного спуску.

В пакеті MATLAB/Simulink було побудовано моделі оптимальних систем. На об’єкти подавалось знайдене керування, а оптимані результати керувань моделювання підтвердити правильність виконаннь розрахунків. Виконати порівняння алгоритмів за швидкодією, та кількісттю ітерацій.

Найпростіший в реалізації був метод рівномірного поділу, але він виявився найменш ефективний – алгоритм працює дуже довго, щоб пришвидшити роботу, потрібні великі обчислювальні потужності. А от алгоритми половинного ділення та золотого перетину набагато швидші, та мають майже однакову швидкодію для складних задач, алгоритм золотого переитину ефективніший. Також можна зробити висновок, що для багатовимірних об’єктів застосовувати одновимірні методи оптимізації не є найкращим рішенням, тому було використано метод покоординатного спуску. Він показав хороші результати, його великою перевагою є те, що його ефективно можна застосовувати для багатовимірних об’єктів.

Також було розроблено стартап проект в якому було досліджено привабливість та спроможність виходу програмного продукту на ринок.

Було виконано наступні задачі:

— огляд літератури та аналіз основних відомих результатів;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МРП для об’єкту першого порядку;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МПП для об’єкту першого порядку;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МЗП для об’єкту першого порядку;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МРП для об’єкту другого порядку;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МПП для об’єкту другого порядку;

— розроблення програмного засобу на базі пакету MATLAB для МДП з використанням МЗП для об’єкту другого порядку;

— розроблення моделі дискретної системи керуання для перевірки отриманих результатів;

— дослідження швидкодії алгоритмів оптимізації в методі динамічного програмування дискретними об’єктами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Оптимальні системи [Електронний ресурс] – Режим доступу до ресрсу: https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0

2. Оптимальні системи. Критерій якості [Електронний ресурс] – Режим доступу до ресрсу: https://studfiles.net/preview/5725769/

3. Denardo E. V. Dynamic Programming: Models and Applications / Eric V. Denardo. – NY: Dover Publications, 2003.

4. Чураков, Е. П. Оптимальные и адаптивные системы : учебное пособие для вузов по специальности "Автоматика и телемеханика" / Е. П. Чураков . – М.: Энергоатомиздат, 1987 . – 256 с.

5. Акулич И. Л. Математическое программирование в примерах и задачах: Учеб. пособие для студентов эконом. спец. вузов. — М.: Высшая школа, 1986. – 317 с.

6. Алгоритм рівномірного поділу [Електронний ресурс] – Режим доступу до ресрсу: http://www.mathros.net.ua/minimizacija-funkcii-odnijei-zminnoi-metodom-rivnomirnogo-poshuku.html

7. Алгоритм половинного поділу [Електронний ресурс] – Режим доступу до ресрсу: http://www.mathros.net.ua/minimizacija-funkcii-odnijei-zminnoi-metodom-dyhotomii.html

8. Алгоритм золотого перетину [Електронний ресурс] – Режим доступу до ресрсу: http://www.mathros.net.ua/minimizacija-funkcii-odnijei-zminnoi-metodom-zolotogo-peretynu.html

9. Атанс М., Фалб Н. Оптимальное управление. М.: Машиностроение, 1968. – 784 с.

10. Иванов В. А., Фалдин Н. В. Теория оптимальных систем автоматического управления М. Наука, 1981.- 336 с.

11. Олейников В.А., Зотов Н.С., Пришвин А.М. Основы оптимального и

экстремального управления. М.: Высшая школа, 1969. – 296 с.

12. Чураков Е.П. Оптимальные и адаптивные системы. М.: Энергоиздат, 1987. - 256 с.

13. Александров А.Г. Оптимальные и адаптивные системы. М.: Высшая школа, 1989. – 263 с.

14. Теория оптимизации систем автоматического управления, под ред. К.А. Пупкова. М.: Издательство МГТУ им. Н.Э. Баумана, 2004. – 656 с.

15. Оптимальні та адаптивні системи–1. Методи теорії оптимального керування: метод. Вказівки до викон. Лабораторн. робіт для студ. спец. 7,8.05020101 “Комп’ютеризовані системи управління та автоматика” / Уклад. А.В. Писаренко, Н.Б. Репнікова. – К.: НТУУ “КПІ”, 2013. – 128с.

16. Костюк В.И., Ажогин В.В. Оптимальные системы цифрового управления технологическими процессами. К.: Техника, 1982. – 170 с.

17. Болтянский В.Г. Математические методы оптимального управления. М.: Наука, 1968. – 408 с.

18. Куропаткин П.В. Оптимальные и адаптивные системы. М.: Высшая школа, 1980. – 287 с.

19. Клюев А.С., Колесников А.А. Оптимизация автоматических систем управления по быстродействию. М.: Энергоиздат, 1982. – 239 с.

20. Оптимальні та робото спроможні системи управління [Електронний ресурс] – Режим доступу до ресрсу: http://matlab.exponenta.ru/optimrobast/

21. Корн Г., Корн Т. Довідник з математики для науковців та інженерів. - М .: Наука, 1970. - С. 575-576.

22. Оптимальні системи [Електронний ресурс] – Режим доступу до ресрсу: http://alnam.ru/book_ads.php?id=3

23. Г. Реклейтис, А. Рейвиндран, К. Рэгсдел Оптимизация в технике: В2-х кн. Кн.1. Пер с англ.- М.: Мир, 1986.-350 с.

24. Кіні Р. Л., Райфа Х. Прийняття рішень при багатьох критеріях: переваги і заміщення. - М .: Радио и связь, 1981. - 560 с.

25. Г. Реклейтис, А. Рейвиндран, К. Рэгсдел Оптимизация в технике: В2-х кн. Кн.2. Пер с англ.- М.: Мир, 1986.-320 с.

26. Т. Кармен, Ч. Лейзерсон, Р. Ривест Алгоритмы: построение и анализ. М.: МЦНМО, 2001.- 960с., 263 іл.

27. Скиена С. Алгоритмы. Руководство по разработке.-2-е узд.: Пер с англ..-СПб.: БХВ-Петербург, 2011.- 720с.

28. Седжвик Роберт. Фундаментальные алгоритмы на С. Анализ/Структуры данных/ Сортировка/Поиск/Алгоритмы на графах: Пер.с англ./ Роберт Седжвик.- СПб: ООО «Диа СофтЮП», 2003.- 1136 с.

29. Мінімізація функції однієї змінної методом рівномірного пошуку [Електронний ресурс] – Режим доступу до ресрсу: http://www.mathros.net.ua/minimizacija-funkcii-odnijei-zminnoi-metodom-rivnomirnogo-poshuku.html

30. Оптимізація функції багатьох змінних методом покоординатного спуску [Електронний ресурс] – Режим доступу до ресрсу: http://www.mathros.net.ua/optymizacija-funkcii-bagatoh-zminnyh-metodom-pokoordynatnogo-spusku.html

Додаток А. Тези

CONFERENCE PROCEEDINGS

МАТЕРІАЛИ КОНФЕРЕНЦІЇ

summer infoCom 2018

Summer InfoCom Advanced Solutions 2018

VI МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ
20-21 травня 2018 року

ISBN 978-966-2344-65-3

Україна, Київ

Алгоритм покоординатного спуску в методі динамічного програмування для задач цифрового оптимального керування

Майер І.В.
КПІ ім. Ігоря Сікорського
Київ, Україна

Писаренко А.В.
КПІ ім. Ігоря Сікорського
Київ, Україна

Анотація. **Запропоновано модифікацію алгоритму розв'язання задачі оптимального керування дискретними об'єктами методом динамічного програмування шляхом застосування методу покоординатного спуску.**

Ключові слова: динамічне програмування, оптимальне керування, метод рівномірного пошуку, метод покоординатного спуску, моделювання, MATLAB /Simulink.

Значне число задач, що виникають в суспільстві, пов'язані з процесами прийняття рішень. Параметри цих процесів вибираються таким чином, щоб забезпечити екстремальне значення певного показника (вартість, маса, час роботи, тощо) при умові наявних обмежень на параметри процесу. Сьогодні більшість систем керування проектують на основі різного роду цифрових обчислювальних пристроїв, що забезпечує високу точність, швидкодію, гнучкість при налаштуванні та збалансовану вартість. Розроблення та вдосконалення алгоритмів, що реалізують (зокрема) методи оптимального керування для побудови високошвидкісних цифрових систем автоматичного керування є актуальної задачею сьогодення.

При проектуванні і оптимізації системи важливим моментом є формування цілі оптимізації, яка математично виражається як вимога забезпечення екстремального значення деякого критерію оптимальності.

Динамічне програмування – це підхід до розв'язання складних задач шляхом їх розбиття на більш прості підзадачі. Даний метод заснований на принципі оптимальності Беллмана, який дозволяє скоротити перебір рішень в багатоетапних нелінійних задачах [1].

Метод динамічного програмування для задачі оптимального керування полягає у тому що потрібно з початкової заданої точки траєкторії об'єкту керування оптимально потрапити в задану кінцеву точку, частіше всього це 0. Для цього задачу ділять на підзадачі і починають вирішувати її з кінця – це називається зворотній прохід. Він полягає в тому, що на кожному кроці знаходять відповідно до правил обчислень усі можливі оптимальні рішення. Далі застосовується прямий прохід, який з усіх можливих знайдених рішень для кожної підзадачі дозволяє сформувати оптимальне рішення основної задачі.

Відомий підхід до розв'язання задачі оптимального керування об'єктом першого порядку методом динамічного програмування [2] для знаходження на кожному проміжку часу мінімального рішення використовує метод рівномірного пошуку. Хоча даний метод і є легким у реалізації, але він не є ефективним, оскільки витрачається багато часу та обчислювальних потужностей для вирішення задачі. Для позбавлення від вказаних недоліків замість методу рівномірного пошуку для функції однієї змінної було обрано метод покоординатного спуску для функції багатьох змінних.

Застосуємо метод покоординатного спуску для розв'язання підзадач зворотного проходу методу динамічного програмування.

На першому кроці методу покоординатного спуску в якості початкового наближення обираємо одну із координат $M_0(x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})$ на заданій області допустимих значень. Підставляємо в критерій оптимальності I усі точки початкового наближення крім першої. Далі отримаємо функцію однієї змінної $I = \int(x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})$. Знайшовши для даної функції точку мінімуму, переходимо від точки M_0 до точки $M_1(x_1^{(1)}, x_2^{(0)}, ..., x_n^{(0)})$, в якій критерій оптимальності I приймає мінімальне значення по координаті x_1. Це перший крок оптимізації, що складається з спуску по координаті x_1. Для знаходження мінімуму можна використовувати метод золотого перетину, метод Фібоначі, тощо. Далі потрібно підставити в критерій I всі координати точки M_1 крім x_2 та розглянути функцію виду $I = \int(x_1^{(1)}, x_2^{(0)}, ..., x_n^{(0)})$. Знову вирішити одновимірну задачу оптимізації та знайти нову точку $M_2(x_1^{(1)}, x_2^{(1)}, ..., x_n^{(0)})$, в якій критерій I приймає мінімальне значення по координаті x_2. Аналогічно проводиться пошук за координатами $x_3, x_4, \ldots x_n$. Після цього все повторюється від x_1 до x_n. В результаті буде отримана послідовність точок $M_0, M_1, \ldots M_n$ в яких значення цільової функції складають монотонно спадну послідовність $I(M_0) \geq I(M_1) \geq I(M_2) \geq \ldots$. Також на будь-якому і-му кроці можна даний процес призупинити та в якості мінімального значення використовувати значення критерію в точці M_i [3].

Таким чином, метод покоординатного спуску зводить задачу про знаходження найменшого

значення функції багатьох змінних до багаторазового рішення одновимірних задач оптимізації по кожній з цих змінних.

В середовищі MATLAB було написано скрипт для розв'язання задачі оптимального керування методом динамічного програмування з використанням методу покоординатного спуску.

Розглянемо приклад. Задано об'єкт другого порядку у вигляді дискретної моделі у просторі станів:

$$\begin{cases} x_1[k+1] = 0,1765x_1[k] - 0,7686x_2[k] + 0,25u[k], \\ x_2[k+1] = 0,5x_1[k] + 0,5x_2[k], \\ y[k] = x_1[k] + x_2[k]. \end{cases} \quad (1)$$

Необхідно знайти таку оптимальну керуючу послідовність u, що переведе об'єкт керування з початкового стану $Xstart = [5\ ; -5]$ в кінцевий стан $Xend = [0\ ;0]$ за $N = 13$ кроків квантування, а значення критерію якості $I = \int (x_1^2 + x_2^2 + u^2)$ буде мінімальним. Область допустимих керувань: $\Omega(u) : |u| \le 1$. Області допустимих станів: $V(x_1) : x_1 c [-8; 12]$, $V(x_2) : x_2 c [-9\ ;12]$.

В результаті роботи скрипту оптимано послідовність керувань та станів об'єкту в кожний момент квантування :

k=0; x1=-5.000; x2=5.000; U=-0.0345
k=1; x1=-4.733; x2=0.000; U=-0.0345
k=2; x1=-0.843; x2=-2.366; U=-0.0345
k=3; x1=1.662; x2=-1.605; U=-0.0345
k=4; x1=1.518; x2=0.000; U=-0.0345
k=5; x1=0.238; x2=0.773; U=-0.0345
k=6; x1=-0.561; x2=0.505; U=0.0345
k=7; x1=-0.478; x2=0.000; U=-0.0345
k=8; x1=-0.072; x2=-0.253; U=-0.0345
k=9; x1=0.175; x2=-0.162; U=0.0345
k=10; x1=0.175; x2=0.000; U=-0.0345
k=11; x1=0.016; x2=0.085; U=-0.0345
k=12; x1=-0.071; x2=0.0505; U=0.0345
k=13; x1=0.000; x2=0.000; U=-0.0345

На рис. 1 представлено побудовану в MATLAB /Simulink модель дискретної системи керування. Отриману послідовність керувань подаємо на об'єкт (Discrete State-Space) за допомогою блоку From Workspace. На рис. 2 представлено графіки перехідних процесів за станами x_1 (суцільна лінія) та x_2 (штрихова лінія). На рис. 3 зображено оптимальну траєкторію об'єкту керування, а на рис. 4 – графік керування системи.

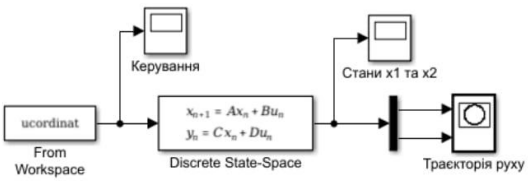


Рис. 1. Модель дискретної системи керування у MATLAB/Simulink

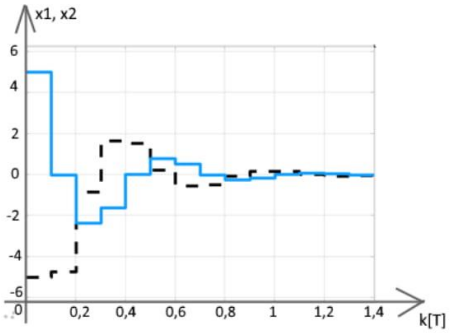


Рис. 2. Перехідні процеси за станами x1 та x2

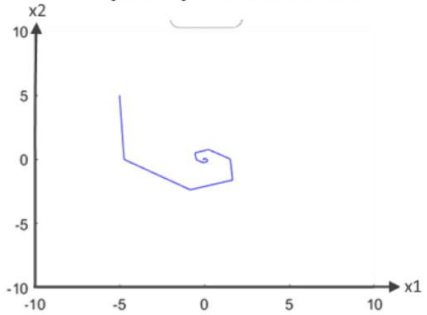


Рис. 3. Траєкторія руху об'єкту

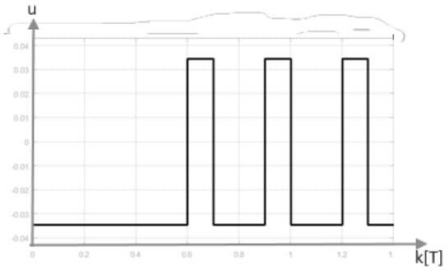


Рис. 4. Графік керування

З отриманих графіків видно, що знайдена оптимальна послідовність керування переводить його з заданого початкового стану у заданий кінцевий стан за визначену кількість тактів квантування.

ЛІТЕРАТУРА

1. Denardo E. V. Dynamic Programming: Models and Applications / Eric V. Denardo. – NY: Dover Publications, 2003.

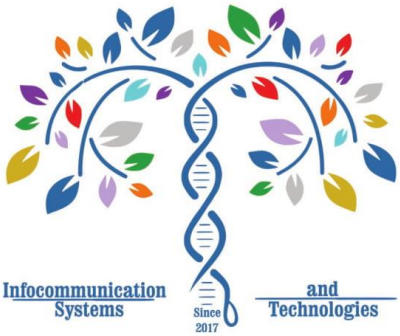
2. Чураков, Е. П. Оптимальные и адаптивные системы : учебное пособие для вузов по специальности "Автоматика и телемеханика" / Е. П. Чураков . – М.: Энергоатомиздат, 1987 . – 256 с.

3. Акулич И. Л. Математическое программирование в примерах и задачах: Учеб. пособие для студентов эконом. спец. вузов. — М.: Высшая школа, 1986.

Додаток Б. Стаття

Number 2(2) May 2018 ISSN 2520-6257

Infocommunication Systems and Technologies



Управление городским транспортом

Дослідження однорідності кабельних ліній

Швидкодія алгоритмів оптимізації в методі динамічного програмування для задач цифрового оптимального керування

Майер І.В.
КПІ ім. Ігоря Сікорського
Київ, Україна

Писаренко А.В.
КПІ ім. Ігоря Сікорського
Київ, Україна

Анотація. **Виконано дослідження швидкості алгоритму розв'язання задачі оптимального керування дискретними об'єктами методом динамічного програмування шляхом застосування методів оптимізації: рівномірного поділу, половинного ділення та золотого перетину.**

Ключові слова: динамічне програмування, оптимальне керування, метод рівномірного поділу, метод половинного ділення, метод золотого перетину, MATLAB /Simulink.

Ключовою проблемою сучасної теорії управління є оптимізація. Розв'язання цієї проблеми вимагає розробки та практичного застосування методів оптимізації, що базуються на використанні можливостей сучасних ЕОМ. Оптимізація алгоритму – це процес поліпшення різних характеристик і процесів обчислення. Основними характеристики, які вимагають покращення в процесі оптимізації алгоритму є:

- час виконання;
- кількість ітерацій;
- об'єм пам'яті, використовуваної алгоритмом у процесі обчислень.

Одним з основних критеріїв, що визначає перевагу того чи іншого обчислювального алгоритму є швидкодія. Швидкодія процесу – одна з найважливіших його характеристик, яка визначає ефективність роботи всієї системи, визначається часом виконання операторів і кількістю ітерацій алгоритму.

Серед прикладних задач оптимального керування важливе місце займають задачі динамічного програмування. В задачі динамічного програмування задано математичну модель об'єкту керування, його початковий стан, критерій оптимальності, та область допустимих керувань. Необхідно знайти таку послідовність керуючих впливів, що належить до області допустимих керувань, при якій критерій якості досягає мінімального значення, а об'єкт переводиться із заданого початкового стану у кінцевий. Критерій оптимальності є мірою наближення розв'язку до поставленої мети. В задачах, як правило, таким критерієм виступає показник ефективності функціонування системи або показник витрат.

Для розв'язання складної оптимізаційної задачі методом динамічного програмування її ділять на підзадачі. Розв'язання задачі починається з кінця – зворотній прохід. Вважаємо, що керування u[0], u[1], …, u[N-1] відомі. На N-кроці обчислюємо усі можливі значення критерію для кожного можливого значення станів та керування. З них визначаємо мінімальне значення критерію для кожного стану об'єкту та відповідне йому керування. На кожному наступному кроці, починаючи з N-1 по 2 крок, знаходимо значення критерію якості, що відповідає кожному можливому значенню змінних стану, керуванню та мінімального значення критерію з попередніх кроків алгоритму. Після формування таблиць залежності мінімальних значень критеріїв від станів та керувань переходять до прямого проходу – на кожному кроці підставляємо в рівняння стану об'єкту значення керування та станів і знаходимо з усіх можливих значень в таблиці найближче значення змінних стану та відповідне їм значення керування u. Таким чином остаточно отримуємо оптимальну послідовність керувань u[0], u[1], …, u[N -1], яка подається на об'єкт.

Розглянемо алгоритми, які використовуються для мінімізації функції на інтервалі. Перший з них – метод рівномірного поділу, який відноситься до пасивних стратегій пошуку точки екстремуму. Він полягає в тому, що задається інтервал невизначеності [xmin; xmax] та кількість інтервалів n. Обчислення проходять на відстані o n дна від іншої точках. При цьому інтервал ділиться на n+1 відрізків (рис. 1). Далі на кожному кроці відбувається перебір визначеного значення стану x з усіма можливими значеннями керування u, обчислюється значення критерію та знаходиться мінімальне [1].

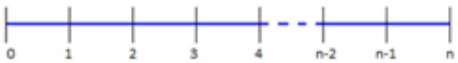


Рис 1. Метод рівномірного поділу

Метод половинного ділення відноситься до послідовних стратегій і дозволяє виключати з подальшого розгляду на кожній ітерації половину поточного інтервалу невизначеності. В даному алгоритмі на початку задано критерій та обмеження [xmin; xmax]. На даному проміжку потрібно знайти точку мінімуму критерію з заданою точністю ε. Для цього потрібно знайти x_1 та x_2 за формулами (1), (2):

$$x_1 = \frac{x_{min} + x_{max} - \delta}{2} \qquad (1)$$

$$x_2 = \frac{x_{min} + x_{max} + \delta}{2} \qquad (2)$$

де $\delta < \varepsilon$. Після чого обчислюємо значення критерію в знайдених точках з усіма можливими керуваннями u та знаходимо мінімальне значення критерію в точках x_1 та x_2. Порівнюємо два значення критерію і якщо $I(x_1) < I(x_2)$, то значення правого обмеження змінюємо на значення в точці x_2, тобто $x_{max} = x_2$. В протилежному випадку, якщо $I(x_1) > I(x_2)$, то змінюємо значення лівого обмеження на значення в точці x_1. Перейшовши до наступної ітерації обчислюємо знову x_1 та x_2 так визначаємо новий інтервал обмеження. Графічну інтерпретацію даного методу можна спостерігати на рис. 2. Пошук закінчується, якщо довжина поточного інтервалу невизначеності менша заданого числа ε [2].

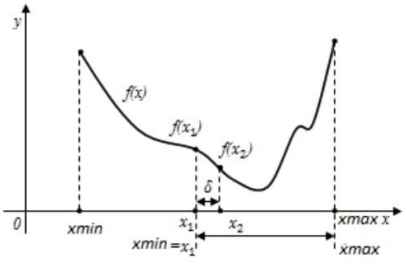


Рис. 2. Метод половиного ділення

Метод золотого перетину полягає у поділі відрізка [xmin; xmax] точкою *c* на дві нерівні частини таким чином, щоб відношення усього відрізка до більшої частини дорівнювало відношенню більшої частини до меншої (рис. 3), тобто:

$$(x_{max} - x_{min})/(x_{max} - c) = (x_{max} - c)/(c - x_{min}) = r \qquad (3)$$

r – золоте відношення, значення відоме (r ≈ 1,618).

Рис 3. Метод золотого перетину (співвідношення)

На першому кроці початковий відрізок [xmin; xmax] ділимо точкою *c* (5) та *d* (4) за правилом золотого перетину.

$$d = x_{min} + \frac{x_{max} - x_{min}}{r} = x_{max} - \frac{3 - \sqrt{5}}{2}(x_{max} - x_{min}) \qquad (4)$$

$$c = x_{min} + \left(1 - \frac{2}{r}\right)(x_{max} - x_{min}) = x_{min} + \frac{3 - \sqrt{5}}{2}(x_{max} - x_{min}) \qquad (5)$$

Далі обчислюємо значення критерію *I(c)* та *I(d)*. При порівняні цих значень можна відкинути інтервал [xmin ; c], при умові, якщо $I(c) > I(d)$. Довжина відрізка який залишається для подальших обчислень зменшується в r разів.

Далі процес повторюється, на інтервалі є лише одна точка, що робить його золотий перетин: *c* – друга точка золотого перетину відрізка [c ; x_{max}], а *d*-перша точка золотого перетину відрізка [c ; x_{max}]. Знаючи одну з точок золотого перетину, іншу можна знайти за однією із вищезгаданих формул та обчислити значення критерію у знову знайденій точці (значення в іншій точці вже обчислено на попередньому кроці). Таким чином, на кожному кроці, починаючи з другого, потрібно лише одне обчислення і інтервал невизначеності зменшується в r разів. Процес обчислень за методом золотого перетину продовжуємо до тих пір, поки довжина інтервалу невизначеності не стане меншою деякого заданого числа ε [3].

В середовищі MATLAB було розроблено скрипт методу динамічного програмування для задач цифрового оптимального керування з використанням алгоритмів: рівномірного ділення, половинного та золотого перетину.

Розглянемо як змінюється швидкодія, кількість ітерацій в залежності від алгоритму пошуку на конкретному прикладу з використаними наступних даних. Об'єкт керування задано рівнянням стану

$$x[n+1] = x[n] + u[n]. \qquad (6)$$

N = 7 – кількість тактів квантування.
Xmax = 10 – максимальне значення змінної стану.
Xmin = -19 – мінімальне значення змінної стану.
U = [-1; 1] – область допустимих керувань.
M = 4 – кількість дискретних значень керування.
Xstart = 5 – початковий стан об'єкту.
Xend = 0 – кінцевий стан об'єкту.

$I = \sum_{n=0}^{N} \left(x^2[n] + u^2[n] \right)$ – критерій оптимальності.

Для алгоритму рівномірного поділу використовується R = 100 – кількість дискретних значень змінної стану.

В результаті виконання алгоритму отримаємо послідовність представлену в табл. 1.

Таблиця 1

Результати обчислень алгоритму рівномірного поділу в задачі дискретного програмування

n (ітерація)	x – стан об'єкта	u – керування
0	5,0000	-1,0000
1	4,14141	-1,0000
2	2,96970	-1,0000
3	2,09091	-1,0000
4	0,99919	-0,3333
5	0,62626	-0,3333
6	0,33333	-0,3333
7	0,04040	-0,3333

В пакеті MATLAB/Simulink було побудовано модель системи (рис. 4) на якій присутня дискретна модель об'єкту (блок Discrete State Space), керуюча послідовність (блок From Workspace), та осцилограф (блок Scope). Результати виконання розробленого

скрипту (обчислень алгоритму) задаються в блоці From Workspace. Результати моделювання подано на рис. 5. Швидкодію (кількість ітерацій алгоритму) представлено в табл. 4 для усіх розглянутих алгоритмів.

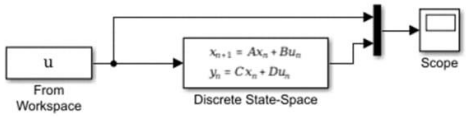


Рис. 4. Модель системи для задачі цифрового оптимального керування

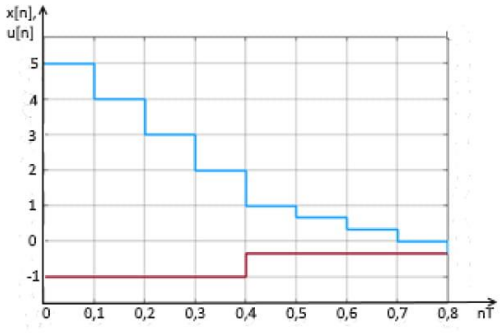


Рис. 5. Результати виконання задачі дискретного програмування з використанням алгоритму рівномірного поділу

Для алгоритму половинного ділення (вхідні данні $\varepsilon = 0.07$) результати відображено в табл. 2 та на рис. 6.

Таблиця 2

Результати обчислень алгоритму половинного поділу в задачі дискретного програмування

n (ітерація)	x – стан об'єкта	u – керування
0	5,0000	-0,3333
1	4,6750	-1,0000
2	3,6750	-1,0000
3	2,6750	-1,0000
4	1,6750	-1,0000
5	0,6750	-1,0000
6	-0,3859	0,3333
7	0,00917	0,3333

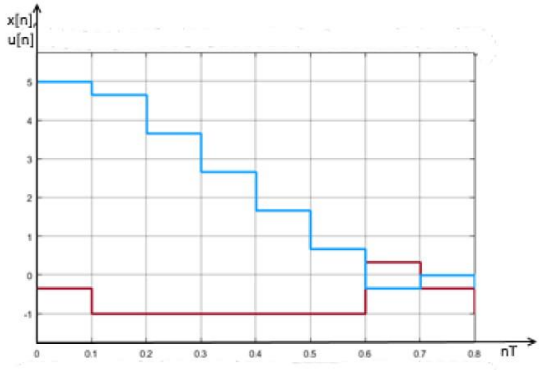


Рис. 6. Результати виконання задачі дискретного програмування з використанням алгоритму половинного поділу

Результати обчислень методом золотого перетину представлені у табл. 3, а моделювання – на рис. 7.

Таблиця 3

Результати обчислень алгоритму золотого перетину в задачі дискретного програмування

n (ітерація)	x – стан об'єкта	u – керування
0	5,0000	-0,3333
1	4,6750	-0,3333
2	4,36750	-0,3333
3	4,0050	-1,0000
4	3,0075	-1,0000
5	2,01075	-1,0000
6	1,01009	-1,0000
7	0,00917	0,3333

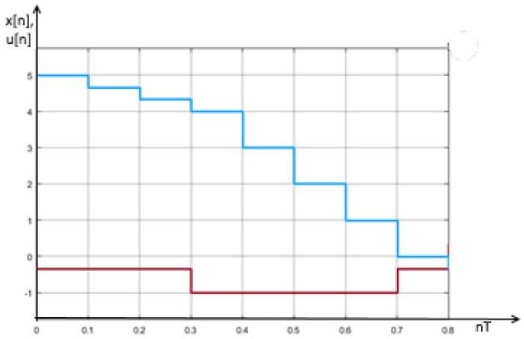


Рис. 7. Результати виконання задачі дискретного програмування з використанням алгоритму золотого перетину

Таблиця 4

Якісні показники алгоритмів

	Алгоритм рівномірного ділення	Алгоритм половинного ділення	Алгоритм золотого перетину
Час виконання, с	5,14	0,14	0,14
Кількість ітерацій	4909	78	71
Швидкодія, ітер/с	955	557	597

Аналізуючи роботу усіх трьох скриптів в табл. 4 виявлено, що найефективніший алгоритм для знаходження оптимуму в методі динамічного програмування для задач цифрового оптимального керування є алгоритм золотого перетину. Алгоритм рівномірного поділу є найпростіший в реалізації, але найменш ефективний. Даний алгоритм можна використовувати для дрібних задач, у яких не важлива швидкість виконання. Алгоритми половинного ділення та золотого перетину ефективні швидкі в роботі методи пошуку, які використовуються коли потрібні швидкі та ефективні рішення.

ЛІТЕРАТУРА

1. Алгоритм рівномірного поділу [Електронний ресурс] – Режим доступу до ресрсу: http://www.mathros.net.ua/minimizacija-funkcii-odnijei-zminnoi-metodom-rivnomirnogo-poshuku.html

2. Алгоритм половинного поділу [Електронний ресурс] – Режим доступу до ресрсу: http://www.mathros.net.ua/minimizacija-funkcii-odnijei-

zminnoi-metodom-dyhotomii.html

3. Алгоритм золотого перетину [Електронний ресурс] – Режим доступу до ресрсу: http://www.mathros.net.ua/minimizacija-funkcii-odnijei-zminnoi-metodom-zolotogo-peretynu.html

4. Denardo E. V. Dynamic Programming: Models and Applications / Eric V. Denardo. – NY: Dover Publications, 2003.

5. Чураков, Е. П. Оптимальные и адаптивные системы : учебное пособие для вузов по специальности "Автоматика и телемеханика" / Е. П. Чураков . – М.: Энергоатомиздат, 1987 . – 256 с.

6. Акулич И. Л. Математическое программирование в примерах и задачах: Учеб. пособие для студентов эконом. спец. вузов. — М.: Высшая школа, 1986.

Додаток В. Скритпм МДП з МРП для об’єкту першого порядку

```
clc;
disp('******* Start *******');

N=10;
Xstart=5;
Xend=0;

M=4;
Umin=-1;
Umax=1;
Ustep=((abs(Umin)+abs(Umax))/(M-1));
u=(Umin:Ustep:Umax);

R=100;
xMin=-19;
xMax=10;
xStep=((abs(xMin)+abs(xMax))/(R-1));
xr=(xMin:xStep:xMax);

G= @(u,x) (x^2+u^2);

iter=0;
tstart = tic;
% end
disp(['---------------------- ',num2str(N),' ----------------------']);
for r=1:R
   g=zeros(M);
   for m=1:M
      iter=iter+1;
      g(m,1)=G(u(m),xr(r));
      g(m,2)=u(m);
   end
   index=minimum(g(:,1));
   result(N,r)=ResultPoint(xr(r),g(index,2),g(index,1));
   fprintf('[%d/%d] Time: %0.2f s; n=%d; r=%d; xr=%0.5f; u=%0.5f;
S=%0.5f\n',iter,R*M*N,toc(tstart),N,r,xr(r),g(index,2),result(N,r).S);
end
```

```
disp('----------------------------------------');

% middle

for n=1:N-1
   disp(['---------- ',num2str(N-n),' ----------']);
   for r=1:R
      iter=iter+1;
      g=zeros(M);
      for m=1:M
         iter=iter+1;
         xnm=xr(r)+u(m);
         index=findclosestkvantum(xnm,result(N-n+1,:));
         g(m,1)=G(u(m),xr(r))+result(N-n+1,index).S;
         g(m,2)=u(m);
    %      fprintf('S=%0.5f; u=%0.5f;\n',g(m,1),g(m,2));
      end

      index=minimum(g(:,1));
      result(N-n,r)=ResultPoint(xr(r),g(index,2),g(index,1));
      fprintf('[%d/%d] Time: %0.2f s; n=%d; r=%d; xr=%0.20f; u=%0.3f;
S=%0.2f;\n',iter,R*M*N,toc(tstart),N-n,r,xr(r),g(index,2),result(N-n,r).S);
   end
end
disp('----------------------------------------');

% first
disp(['---------- ',num2str(0),' ----------']);
g=zeros(M);
for m=1:M
   iter=iter+1;
   x1m=Xstart+u(m);
   ind=findclosestkvantum(x1m,result(1,:));
   g(m,1)=G(u(m),Xstart)+result(1,ind).S;
   g(m,2)=u(m);
end

index=minimum(g(:,1));
result0=ResultPoint(Xstart,g(index,2),g(index,1));
```

```
fprintf('[%d/%d] Time: %0.2f s; n=%d; xr=%0.5f; u=%0.5f; S=%0.5f;\n',iter,R*M*N,toc(tstart),1,result0.X,result0.U,result0.S);
disp('----------------------------------------');

% priamoy
disp('Direct walk');
disp(['n=0; x=',num2str(Xstart),'; U=',num2str(result0.U)]);

un=result0.U;
xn=result0.X+un;
index=findclosestkvantum(xn,result(1,:));
fprintf('n=%d; x=%0.5f; U=%0.3f\n',1,result(n,index).X,result(1,index).U);
un=result(1,index).U;

for n=2:N
   xn=xn+un;
   index=findclosestkvantum(xn,result(n,:));
   fprintf('n=%d; x=%0.5f; U=%0.3f\n',n,result(n,index).X,result(n,index).U);
   un=result(n,index).U;
end

disp('******* End *******');
```

Функція знаходження найближчого значення:

```
function s = findclosestkvantum(x,mas)
   sz=size(mas);
   for i=1:sz(2)
      t = abs(mas(i).X - x);
      if ~exist('m','var')
         m = t;
         s=i;
      elseif m > t
        m=t;
        s=i;
      end
   end
end
```

Функція знаходження мінімум:

```
function s = minimum(x)
    for i=1:size(x)
        if ~exist('m','var')
            m = x(i);
            s=i;
        else
            if m > x(i)
               m=x(i);
               s=i;
            end
        end
    end
end
```

Функція результатної точки

```
classdef ResultPoint
  properties
     X;
     U;
     S;
  end
  methods
     function obj = ResultPoint(x,u,s)
       if nargin > 0
            obj.X = x;
            obj.U = u;
            obj.S = s;
       end
     end
  end
end
```

Додаток Г. МДП з МПП для об’єкту першого порядку

```
clc;
clear;
disp('******* Start *******');

N=7;
Xstart=5;
Xend=0;

M=4;
Umin=-1;
Umax=1;
Ustep=((abs(Umin)+abs(Umax))/(M-1));
u=(Umin:Ustep:Umax);

xMin=-19;
xMax=10;

G= @(u,x) (x^2+u^2);

beta=0.1;
epsilon=0.07;
iter=0;
tic;
tstart = tic;
% end
disp(['---------------------- ',num2str(N),' ----------------------']);

a=xMin;
b=xMax;
i=1;
while 1
   iter=iter+1;
   x1=(a+b-beta)/2;
   x2=(a+b+beta)/2;

   g=zeros(M);
   for m=1:M
      g(m,1)=G(u(m),x1);
```

```
      g(m,2)=u(m);
   end
   index=minimum(g(:,1));
   f1 = ResultPoint(x1,g(index,2),g(index,1));

   g=zeros(M);
   for m=1:M
      g(m,1)=G(u(m),x2);
      g(m,2)=u(m);
   end
   index=minimum(g(:,1));
   f2 = ResultPoint(x2,g(index,2),g(index,1));

   if f1.S > f2.S
      a=f1.X;
      result(N,i)=f2;
      fprintf('[%d] Time: %0.2f s n=%d; i=%d; x=%0.5f; u=%0.5f;
S=%0.5f\n',iter,toc(tstart),N,i,f2.X,f2.U,f2.S);
   end
   if f2.S > f1.S
      b=f2.X;
      result(N,i)=f1;

      fprintf('[%d] Time: %0.2f s n=%d; i=%d; x=%0.5f; u=%0.5f;
S=%0.5f\n',iter,toc(tstart),N,i,f1.X,f1.U,f1.S);
   end
   if (b-a)/2 < epsilon
      break
   end

  i=i+1;
end
disp('----------------------------------------');

% middle

for n=1:N-1
   disp(['---------- ',num2str(N-n),' ----------']);
   a=xMin;
```

```
    b=xMax;
    i=1;
    while 1
        iter=iter+1;
        x1=(a+b-beta)/2;
        x2=(a+b+beta)/2;

        f1 = findMin(x1,u,M,N-n,result,G);
        f2 = findMin(x2,u,M,N-n,result,G);

        if f1.S > f2.S
            a=x1;
            result(N-n,i)=f2;
           fprintf('[%d] Time: %0.2f s n=%d; i=%d; x=%0.10f; u=%0.10f;
S=%0.10f\n',iter,toc(tstart),N-n,i,f2.X,f2.U,f2.S);
        end
        if f2.S > f1.S
            b=x2;
            result(N-n,i)=f1;
           fprintf('[%d] Time: %0.2f s n=%d; i=%d; x=%0.10f; u=%0.10f;
S=%0.10f\n',iter,toc(tstart),N-n,i,f1.X,f1.U,f1.S);
        end
        if (b-a)/2 < epsilon
            break
        end
      i=i+1;
    end
end
disp('----------------------------------------');

% first
disp(['---------------------- ',num2str(0),' ----------------------']);
iter=iter+1;
g=zeros(M);
for m=1:M
    x1m=Xstart+u(m);
    ind=findclosestkvantum(x1m,result(1,:));
    g(m,1)=G(u(m),Xstart)+result(1,ind).S;
    g(m,2)=u(m);
```

```
end
index=minimum(g(:,1));
result0=ResultPoint(Xstart,g(index,2),g(index,1));

fprintf('[%d] Time: %0.2f s n=%d; xr=%0.10f; u=%0.10f;
S=%0.10f;\n',iter,toc(tstart),1,result0.X,result0.U,result0.S);
disp('-----------------------------------------');

% priamoy
disp('Direct walk');
disp(['n=0; x=',num2str(Xstart),'; U=',num2str(result0.U)]);

un=result0.U;
xn=result0.X+un;
index=findclosestkvantum(xn,result(1,:));
fprintf('n=%d; x=%0.10f; U=%0.10f\n',1,result(n,index).X,result(1,index).U);
un=result(1,index).U;

for n=2:N
   xn=xn+un;
   index=findclosestkvantum(xn,result(n,:));
   fprintf('n=%d; x=%0.10f; U=%0.10f\n',n,result(n,index).X,result(n,index).U);
   un=result(n,index).U;
end
t1=toc;
 disp('******** End ********');
```

Додаток Д. МДП з МЗП для об’єкту першого порядку

```
tic;
clc;
clear;
disp('******** Start ********');

N=7;
Xstart=5;
Xend=0;

M=4;
Umin=-1;
Umax=1;
Ustep=((abs(Umin)+abs(Umax))/(M-1));
u=(Umin:Ustep:Umax);

xMin=-19;
xMax=10;

G= @(u,x) (x^2+u^2);

epsilon=0.08;
tau=double((1+sqrt(5))/2);

iter=0;
tstart = tic;

% end
disp(['---------------------- ',num2str(N),' ----------------------']);

a=xMin;
b=xMax;
i=1;
while 1
   iter=iter+1;
   x1=b-(b-a)/tau;
   x2=a+(b-a)/tau;

   g=zeros(M);
```

```
    for m=1:M
        g(m,1)=G(u(m),x1);
        g(m,2)=u(m);
    end
    index=minimum(g(:,1));
    f1=ResultPoint(x1,g(index,2),g(index,1));

    g=zeros(M);
    for m=1:M
        g(m,1)=G(u(m),x2);
        g(m,2)=u(m);
    end

    index=minimum(g(:,1));
    f2=ResultPoint(x2,g(index,2),g(index,1));

    if f1.S >= f2.S
        a=x1;
        result(N,i)=f2;
        fprintf('[%d] Time: %0.2f s n=%d; i=%d; x=%0.5f; u=%0.5f;
S=%0.5f\n',iter,toc(tstart),N,i,f2.X,f2.U,f2.S);
    else
        b=x2;
        result(N,i)=f1;
        fprintf('[%d] Time: %0.2f s n=%d; i=%d; x=%0.5f; u=%0.5f;
S=%0.5f\n',iter,toc(tstart),N,i,x1,f1.U,f1.S);
    end
    if abs(b-a)/2 < epsilon
        break
    end

   i=i+1;
end
disp('----------------------------------------');

% middle

for n=1:N-1
    disp(['---------------------- ',num2str(N-n),' ----------------------']);
```

```
    a=xMin;
    b=xMax;
    i=1;
    while 1
        iter=iter+1;
        x1=b-((b-a)/tau);
        x2=a+((b-a)/tau);

        f1 = findMin(x1,u,M,N-n,result,G);
        f2 = findMin(x2,u,M,N-n,result,G);

        if f1.S >= f2.S
            a=x1;
            result(N-n,i)=f2;
            fprintf('[%d] Time: %0.2f s n=%d; i=%d; x=%0.10f; u=%0.10f;
S=%0.10f\n',iter,toc(tstart),N-n,i,f2.X,f2.U,f2.S);
        else
            b=x2;
            result(N-n,i)=f1;
            fprintf('[%d] Time: %0.2f s n=%d; i=%d; x=%0.10f; u=%0.10f;
S=%0.10f\n',iter,toc(tstart),N-n,i,f1.X,f1.U,f1.S);
        end
        if abs(b-a)/2 < epsilon
            break
        end
      i=i+1;
    end
end
disp('-----------------------------------------');

% first
disp(['---------------------- ',num2str(0),' ----------------------']);
iter=iter+1;
g=zeros(M);
for m=1:M
    x1m=Xstart+u(m);
    ind=findclosestkvantum(x1m,result(1,:));
    g(m,1)=G(u(m),Xstart)+result(1,ind).S;
    g(m,2)=u(m);
```

```
end

index=minimum(g(:,1));
result0=ResultPoint(Xstart,g(index,2),g(index,1));
fprintf('[%d] Time: %0.2f s n=%d; xr=%0.10f; u=%0.10f;
S=%0.10f;\n',iter,toc(tstart),1,Xstart,g(index,2),result0.S);
disp('-----------------------------------------');

% priamoy
disp('Direct walk');
disp(['n=0; x=',num2str(Xstart),'; U=',num2str(result0.U)]);

un=result0.U;
xn=result0.X+un;
index=findclosestkvantum(xn,result(1,:));
fprintf('n=%d; x=%0.10f; U=%0.10f\n',1,result(n,index).X,result(1,index).U);
un=result(1,index).U;

for n=2:N
   xn=xn+un;
   index=findclosestkvantum(xn,result(n,:));
   fprintf('n=%d; x=%0.10f; U=%0.10f\n',n,result(n,index).X,result(n,index).U);
   un=result(n,index).U;
end
t2=toc;
disp('******** End ********');
```

Додаток Е. МДП з МРП для об’єкту другого порядку

```
clc;
clear;
disp('******* Start *******');

N=10;
X1start=3;
X1end=0;
X2start=-3;
X2end=0;

M=10;
Umin=-1;
Umax=1;
Ustep=((abs(Umin)+abs(Umax))/(M-1));
u=(Umin:Ustep:Umax);

R=10;
x1Min=-20;
x1Max=30;
x2Min=-20;
x2Max=30;
x1Step=((abs(x1Min)+abs(x1Max))/(R-1));
x2Step=((abs(x2Min)+abs(x2Max))/(R-1));
x1=(x1Min:x1Step:x1Max);
x2=(x2Min:x2Step:x2Max);

Fi1= @(x1,x2,u) (0.1764*x1-0.7686*x2+0.25*u);
Fi2= @(x1,x2,u) (x1);

G= @(u,x1,x2) (x1^2+x2^2+u^2);

% end
disp(['---------------------- ',num2str(N),' ----------------------']);
for r1=1:R
   for r2=1:R
      g=zeros(M);
      for m=1:M
```

```
            g(m,1)=G(u(m),x1(r1),x2(r2));
            g(m,2)=u(m);
            %fprintf('G=S=%0.5f; u=%0.5f;\n',g(m,1),g(m,2));
        end
        index=minimum(g(:,1));
        result(N,r1,r2)=ResultPoint2D(x1(r1),x2(r2),g(index,2),g(index,1));
      % fprintf('n=%d; r1=%d; r2=%d; x1=%0.5f; x2=%0.5f; u=%0.15f;
S=%0.5f\n',N,r1,r2,x1(r1),x2(r2),g(index,2),S);
    end
end
disp('----------------------------------------');

% middle
for n=1:N-1
    disp(['---------------------- ',num2str(N-n),' ----------------------']);
    for r1=1:R
        for r2=1:R
            g=zeros(M);
            for m=1:M
                xnm1=Fi1(x1(r1),x2(r2),u(m));
                xnm2=Fi2(x1(r1),x2(r2),u(m));
                [index1,index2]=findclosestkvantum2d_2(xnm1,xnm2,result(N-n+1,:,:));
                g(m,1)=G(u(m),x1(r1),x2(r2))+result(N-n+1,index1,index2).S;
                g(m,2)=u(m);
              % fprintf('xnm1=%0.10f; xnm2=%0.10f; G=%0.5f; privS=%0.5f; S=%0.5f;
u=%0.5f;\n',xnm1,xnm2,G(u(m),x1(r1),x2(r2)),result(N-
n+1,index1,index2,4),g(m,1),g(m,2));
            end
            index=minimum(g(:,1));
            result(N-n,r1,r2)=ResultPoint2D(x1(r1),x2(r2),g(index,2),g(index,1));
          %  fprintf('n=%d; r1=%d; r2=%d; x1=%0.5f; x2=%0.5f; u=%0.15f; S=%0.5f\n',N-
n,r1,r2,x1(r1),x2(r2),g(index,2),S);
        end
    end
end
disp('----------------------------------------');

% first
disp(['---------------------- ',num2str(0),' ----------------------']);
```

```
g=zeros(M);
for m=1:M
   x1m=Fi1(X1start,X2start,u(m));
   x2m=Fi2(X1start,X2start,u(m));
   [ind1,ind2]=findclosestkvantum2d_2(x1m,x2m,result(1,:,:));
   g(m,1)=G(u(m),X1start,X2start)+result(1,ind1,ind2).S;
   g(m,2)=u(m);
  % fprintf('S=%0.5f; u=%0.5f;\n',g(m,1),g(m,2));
end
index=minimum(g(:,1));
result0=ResultPoint2D(X1start,X2start,g(index,2),g(index,1));
%fprintf('n=%d;  x1=%0.5f;  x2=%0.5f;  u=%0.15f;
S=%0.5f\n',1,X1start,X2start,g(index,2),S);
disp('----------------------------------------');

% priamoy
disp('Direct walk');
disp(['n=0; x1=',num2str(X1start),'; x2=',num2str(X2start),'; U=',num2str(result0.U)]);

un=result0.U;
xn1=Fi1(result0.X1,result0.X2,un);
xn2=Fi2(result0.X1,result0.X2,un);
[index1,index2]=findclosestkvantum2d_2(xn1,xn2,result(1,:,:));
un=result(1,index1,index2).U;
fprintf('n=%d;  x1=%0.10f;  x2=%0.10f;
U=%0.10f\n',1,result(1,index1,index2).X1,result(1,index1,index2).X2,un);

for n=2:N
   xn1=Fi1(xn1,xn2,un);
   xn2=Fi2(xn1,xn2,un);
   [index1,index2]=findclosestkvantum2d_2(xn1,xn2,result(n,:,:));
   un=result(n,index1,index2).U;
   fprintf('n=%d;  x1=%0.10f;  x2=%0.10f;
U=%0.10f\n',n,result(n,index1,index2).X1,result(n,index1,index2).X2,un);
end

disp('******* End *******');
```

Функція знаходження найближчого значення

```
function index = findclosestkvantum2d(x,mas)
   sz=size(mas);
   for i=1:sz(2)
      t = sqrt((x(1)-mas(i).X1)^2+(x(2)-mas(i).X2)^2);
      if ~exist('m','var')
         m = t;
         index=i;
      elseif m >= t
         m = t;
         index=i;
      end
   end
end
```

Функція знаходження мінімального значення:

```
function f = findMin2D(x1,x2,u,M,n,mas,G,Fi1,Fi2)
   g=zeros(M);
   for m=1:M
      xnm1=Fi1(x1,x2,u(m));
      xnm2=Fi2(x1,x2,u(m));
      [index1]=findclosestkvantum2d([xnm1 xnm2],mas(n+1,:));
      g(m,1)=G(x1,x2,u(m))+mas(n+1,index1).S;
      g(m,2)=u(m);
      %fprintf('S=%0.5f; u=%0.5f;\n',g(m,1),g(m,2));
   end

   index=minimum(g(:,1));
   f = ResultPoint2D(x1,x2,g(index,2),g(index,1));
end
```

Додаток Ж. МДП з МПП для об’єкту другого порядку

```
function result = polovinoe2D(priv,xmin,xmax,fx)
   result=[];
   beta=0.01;
   epsilon=0.02;

   if beta >= (xmax(1)-xmin(1))/2 || beta >= (xmax(2)-xmin(2))/2
      disp('beta must be less then ',str((xmax(1)-xmin(1))/2),' and ',str((xmax(2)-
xmin(2))/2));
      return
   end

   a1=xmin(1);
   b1=xmax(1);
   i1=1;
   while 1
      x11=(a1+b1-beta)/2;
      x12=(a1+b1+beta)/2;

      a2=xmin(2);
      b2=xmax(2);
      while 1
         x21=(a2+b2-beta)/2;
         x22=(a2+b2+beta)/2;

         f11=func2D([x11 x21],priv,fx);
         f12=func2D([x11 x22],priv,fx);

         if f11.S >= f12.S
            a2=f11.X2;
            f1=f12;
         else
            b2=f12.X2;
            f1=f11;
         end
         if abs(b2-a2) < epsilon
            break
         end
```

```
    end

    a2=xmin(2);
    b2=xmax(2);
    while 1
        x21=(a2+b2-beta)/2;
        x22=(a2+b2+beta)/2;

        f11=func2D([x12 x21],priv,fx);
        f12=func2D([x12 x22],priv,fx);

        if f11.S >= f12.S
            a2=f11.X2;
            f2=f12;
        else
            b2=f12.X2;
            f2=f11;
        end
        if abs(b2-a2) < epsilon
            break
        end
    end

   if f1.S >= f2.S
        a1=f1.X1;
        result=[result,f2];
        fprintf('i=%d;          x1=%0.5f;          x2=%0.5f;          u=%0.5f;
S=%0.5f\n',i1,f2.X1,f2.X2,f2.U,f2.S);
    else
        b1=f2.X1;
        result=[result,f1];
        fprintf('i=%d;          x1=%0.5f;          x2=%0.5f;          u=%0.5f;
S=%0.5f\n',i1,f1.X1,f1.X2,f1.U,f1.S);
    end
    if abs(b1-a1) < epsilon
        break
    end
   i1=i1+1;
 end
```

```
end
```

Додаток К. МДП з МЗП для об’єкту другого порядку

```
clc;
clear;
disp('******* Start *******');

N=10;
X1start=3;
X1end=0;
X2start=-3;
X2end=0;

M=100;
Umin=-1;
Umax=1;
Ustep=((abs(Umin)+abs(Umax))/(M-1));
u=(Umin:Ustep:Umax);

x1Min=-20;
x1Max=30;
x2Min=-20;
x2Max=30;

Fi1= @(x1,x2,u) (0.1764*x1-0.7686*x2+0.25*u);
Fi2= @(x1,x2,u) (x1);
G= @(x1,x2,u) (x1^2+x2^2+u^2);

beta=0.1;
epsilon=0.7;
tau=double((1+sqrt(5))/2);
iter=0;
tic;
tstart = tic;

% end
disp(['---------------------- ',num2str(N),' ----------------------']);
a1=x1Min;
b1=x1Max;
i1=1;
while 1
```

```
iter=iter+1;
x11=b1-(b1-a1)/tau;              % обчислюємо значення x1 та x2
x12=a1+(b1-a1)/tau;

a2=x2Min;
b2=x2Max;
i2=1;
while 1
   iter=iter+1;
   x21=b2-(b2-a2)/tau;           % обчислюємо значення x1 та x2
   x22=a2+(b2-a2)/tau;

   g=zeros(M);
   for m=1:M
      g(m,1)=G(x11,x21,u(m));
      g(m,2)=u(m);
   end
   index=minimum(g(:,1));
   f11 = ResultPoint2D(x11,x21,g(index,2),g(index,1));

   g=zeros(M);
   for m=1:M
      g(m,1)=G(x11,x22,u(m));
      g(m,2)=u(m);
   end
   index=minimum(g(:,1));
   f12 = ResultPoint2D(x11,x22,g(index,2),g(index,1));

   if f11.S >= f12.S
      a2=x21;
      f1=f12;
   else
      b2=x22;
      f1=f11;
   end
   if abs(b2-a2)/2 < epsilon
      break
   end
  i2=i2+1;
```

```
 end

 a2=x2Min;
 b2=x2Max;
 while 1
    iter=iter+1;
    x21=b2-(b2-a2)/tau;          % обчислюємо значення x1 та x2
    x22=a2+(b2-a2)/tau;

    g=zeros(M);
    for m=1:M
       g(m,1)=G(x12,x21,u(m));
       g(m,2)=u(m);
    end
    index=minimum(g(:,1));
    f11 = ResultPoint2D(x12,x21,g(index,2),g(index,1));

    g=zeros(M);
    for m=1:M
       g(m,1)=G(x12,x22,u(m));
       g(m,2)=u(m);
    end
    index=minimum(g(:,1));
    f12 = ResultPoint2D(x12,x22,g(index,2),g(index,1));

    if f11.S >= f12.S
       a2=x21;
       f2=f12;
    else
       b2=x22;
       f2=f11;
    end
    if abs(b2-a2)/2 < epsilon
       break
    end
 end

if f1.S >= f2.S
    a1=f1.X1;
```

```
      result(N,i1)=f2;
      fprintf('[%d] Time: %d min n=%d; i=%d; x1=%0.5f; x2=%0.5f; u=%0.5f;
S=%0.5f\n',iter,toc(tstart)/60,N,i1,f2.X1,f2.X2,f2.U,f2.S);
  else
      b1=f2.X1;
      result(N,i1)=f1;
      fprintf('[%d] Time: %d min n=%d; i=%d; x1=%0.5f; x2=%0.5f; u=%0.5f;
S=%0.5f\n',iter,toc(tstart)/60,N,i1,f1.X1,f1.X2,f1.U,f1.S);
   end
   if abs(b1-a1)/2 < epsilon
      break
   end
  i1=i1+1;
end
disp('-----------------------------------------');

% middle
for n=1:N-1
   disp(['---------------------- ',num2str(N-n),' ----------------------']);
   a1=x1Min;
   b1=x1Max;
   i1=1;
   while 1
      iter=iter+1;
      x11=b1-(b1-a1)/tau;            % обчислюємо значення x1 та x2
      x12=a1+(b1-a1)/tau;

      a2=x2Min;
      b2=x2Max;
      i2=1;
      while 1
         iter=iter+1;
         x21=b2-(b2-a2)/tau;            % обчислюємо значення x1 та x2
         x22=a2+(b2-a2)/tau;

         f11 = findMin2D(x11,x21,u,M,N-n,result,G,Fi1,Fi2);
         f12 = findMin2D(x11,x22,u,M,N-n,result,G,Fi1,Fi2);

         if f11.S >= f12.S
```

```
        a2=x21;
        f1=f12;
    else
        b2=x22;
        f1=f11;
    end
    if abs(b2-a2)/2 < epsilon
        break
    end
   i2=i2+1;
end

a2=x2Min;
b2=x2Max;
i2=1;
while 1
    iter=iter+1;
    x21=b2-(b2-a2)/tau;          % обчислюємо значення x1 та x2
    x22=a2+(b2-a2)/tau;

    f11 = findMin2D(x12,x21,u,M,N-n,result,G,Fi1,Fi2);
    f12 = findMin2D(x12,x22,u,M,N-n,result,G,Fi1,Fi2);

    if f11.S >= f12.S
        a2=x21;
        f2=f12;
    else
        b2=x22;
        f2=f11;
    end
    if abs(b2-a2)/2 < epsilon
        break
    end
   i2=i2+1;
end

if f1.S >= f2.S
    a1=x11;
    result(N-n,i1)=f2;
```

```
        fprintf('[%d] Time: %d min n=%d; i=%d; x1=%0.5f; x2=%0.5f; u=%0.5f; S=%0.5f\n',iter,toc(tstart)/60,N-n,i1,f2.X1,f2.X2,f2.U,f2.S);
      else
        b1=x12;
        result(N-n,i1)=f1;
        fprintf('[%d] Time: %d min n=%d; i=%d; x1=%0.5f; x2=%0.5f; u=%0.5f; S=%0.5f\n',iter,toc(tstart)/60,N-n,i1,f1.X1,f1.X2,f1.U,f1.S);
       end
       if abs(b1-a1)/2 < epsilon
          break
       end
      i1=i1+1;
   end
end
disp('-----------------------------------------');

% first
disp(['---------------------- ',num2str(0),' ----------------------']);
iter=iter+1;
g=zeros(M);
for m=1:M
   x1m=Fi1(X1start,X2start,u(m));
   x2m=Fi2(X1start,X2start,u(m));
   ind=findclosestkvantum2d([x1m x2m],result(1,:));
   g(m,1)=G(X1start,X2start,u(m))+result(1,ind).S;
   g(m,2)=u(m);
end
index=minimum(g(:,1));
result0=ResultPoint2D(X1start,X2start,g(index,2),g(index,1));

fprintf('[%d] Time: %d min n=%d; x1=%0.10f; x2=%0.10f; u=%0.10f; S=%0.10f;\n',iter,toc(tstart)/60,1,result0.X1,result0.X2,result0.U,result0.S);
disp('-----------------------------------------');

% priamoy
disp('Direct walk');
disp(['n=0; x1=',num2str(X1start),'; x2=',num2str(X2start),'; U=',num2str(result0.U)]);

un=result0.U;
```

```
xn1=Fi1(result0.X1,result0.X2,un);
xn2=Fi2(result0.X1,result0.X2,un);
[index1]=findclosestkvantum2d([xn1 xn2],result(1,:));
un=result(1,index1).U;
fprintf('n=%d;                    x1=%0.10f;                    x2=%0.10f;
U=%0.10f\n',1,result(1,index1).X1,result(1,index1).X2,un);

for n=2:N
   xn1=Fi1(xn1,xn2,un);
   xn2=Fi2(xn1,xn2,un);
   [index1]=findclosestkvantum2d([xn1 xn2],result(n,:));
   un=result(n,index1).U;
   fprintf('n=%d;                    x1=%0.10f;                    x2=%0.10f;
U=%0.10f\n',n,result(n,index1).X1,result(n,index1).X2,un);
end

t1=toc;

disp('******** End ********');
```

Додаток Л. МДП з МПС для об’єкту другого порядку

Генеруючий файл

```
clc;
clear;
disp('******* Start *******');

N=14;
Xstart=[-5 5];
Xend=[0 0];

Xmin=[-8 -9];
Xmax=[12 12];

Fx1= @(x,u) (0.1764*x(1)-0.07686*x(2)+0.25*u);
Fx2= @(x,u) (0.5*x(1)+0.5*x(2));
NextX=@(x,u) ([Fx1(x,u) Fx2(x,u)]);

f=@(priv,xmin,xmax,fx) (coordinate_descent(priv,xmin,xmax,fx));
%f=@(priv,xmin,xmax,fx) (polovinoe2D(priv,xmin,xmax,fx));

tstart = tic;
% end
disp(['---------------------- ',num2str(N),' ----------------------']);
t=f([],Xmin,Xmax,NextX);
result(N,:)=t;
sz=size(t);
disp('-----------------------------------------');

% middle
for n=1:N-1
   disp(['---------------------- ',num2str(N-n),' ----------------------']);
   t=f(result(N-n+1,:),Xmin,Xmax,NextX);
   sz2=size(t);
   if sz2(2) > sz(2)
      result(N-n,:)=t(1:sz(2));
   else
      t(sz(2))=ResultPoint2D;
      result(N-n,:)=t;
   end
```

```
    disp('----------------------------------------');
end

% start
disp(['---------------------- ',num2str(0),' ----------------------']);
result0=func2D(Xstart,result(1,:),NextX);
disp('----------------------------------------');

% direct
disp('Direct walk');
fprintf('n=%d;\tx1=%0.15f;\tx2=%0.15f;\tU=%0.15f\n',0,result0.X1,result0.X2,result0.U);

un=result0.U;
xn=NextX([result0.X1 result0.X2],un);
index=findclosestkvantum2d(xn,result(1,:));
un=result(1,index).U;
fprintf('n=%d;\tx1=%0.15f;\tx2=%0.15f;\tU=%0.15f\n',1,result(1,index).X1,result(1,index
).X2,result(1,index).U);

for n=2:N
    xn=NextX(xn,un);
    index=findclosestkvantum2d(xn,result(n,:));
    resultN=result(n,:);
    un=resultN(index).U;

fprintf('n=%d;\tx1=%0.15f;\tx2=%0.15f;\tU=%0.15f\n',n,resultN(index).X1,resultN(index
).X2,resultN(index).U);
end
disp('******* End *******');
```

Алгоритм МПС

```
function result = coordinate_descent(priv,xmin,xmax,fx)
    eps=0.008;
    result=[];

    x1p=xmin(1);
    x2p=xmin(2);

    k=true;
```

```
    fmin=0;
    while true
        if k == false
            break;
        end
        i=1;
        [x1p,r]=coordinate_descent_min(x2p,x2p,xmax(2),i,priv,fx,result);
        f=func2D([x1p x2p],priv,fx);
        fmin1=f.S;
        if abs(fmin1-fmin)<eps
            break;
        end
        result=[result,r];
        fmin=fmin1;
        [x2p,r]=coordinate_descent_min(x1p,x1p,xmax(1),i,priv,fx,result);
        f=func2D([x1p x2p],priv,fx);
        fmin1=f.S;
        if abs(fmin1-fmin)<eps
            break;
        end
        result=[result,r];
        fmin=fmin1;
    end
end
```

Функція мінімізації для даного методу

```
function [Xmin,result] = coordinate_descent_min(x,a,b,i,priv,fx,result)
    eps=0.00008;
    t=(3-sqrt(5))/2;
    c=a+t*(b-a);
    d=a+b-c;

    while true
        if i==1
            f1=func2D([c x],priv,fx);
            k1=f1.S;
            f2=func2D([d x],priv,fx);
            k2=f2.S;
```

```
        else
            if i==2
                f1=func2D([x c],priv,fx);
                k1=f1.S;
                f2=func2D([x d],priv,fx);
                k2=f2.S;
            end
        end
         %fprintf('f1                                                    x1=%0.10f;
x2=%0.10f;\tS=%0.20f;\tu=%0.10f;\n',f1.X1,f1.X2,f1.S,f1.U);
        %fprintf('f2                                                    x1=%0.10f;
x2=%0.10f;\tS=%0.20f;\tu=%0.10f;\n',f2.X1,f2.X2,f2.S,f2.U);

        if k1 >= k2
            a=c;
            c=d;
            d=d-t*(b-a);
        else
            b=d;
            d=c;
            c=a+b-d;
        end
         if abs(b-a)<eps
            break;
        end

        if i==1
            f=func2D([(a+b)/2 x],priv,fx);
        else
            if i==2
                f=func2D([x (a+b)/2],priv,fx);
            end
        end
        result=[result,f];
        fprintf('f  x1=%0.10f; x2=%0.10f;\tS=%0.20f;\tu=%0.10f;\n',f.X1,f.X2,f.S,f.U);
    end
        Xmin = (a+b)/2;
end
```