КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра моделювання складних систем

Кваліфікаційна робота бакалавра

за спеціальністю 113 Прикладна математика

на тему:

ЗАСТОСУВАННЯ МЕТОДІВ КОНТИНУАЛЬНОГО ЛІНІЙНОГО ПРОГРАМУВАННЯ ДО ВИРІШЕННЯ КОНКРЕТНИХ ПРОБЛЕМ

Виконав студент 4-го курсу
Тимур КРАСНОСЛОБОДЦЕВ

Науковий керівник
доцент, кандидат фізико-математичних наук
Марина КОРОБОВА

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань
Студент

Роботу розглянуто на засіданні кафедри моделювання складних систем та рекомендовано до захисту в ЕК. Протокол № 10 від 5 червня 2023 р.

Завідувач кафедри МСС доц. Дмитро ЧЕРНІЙ

Київ – 2023

РЕФЕРАТ

Обсяг роботи 43 сторінки, 5 рисунків, 7 використаних джерел.

Об’єктом роботи є методи континуального лінійного програмування до вирішення конкретних проблем.

Предметом роботи є застосування методів для розв’язання задач континуального лінійного програмування.

Метою роботи є дослідження алгоритмів вирішення задач континуального лінійного програмування.

Результати роботи: Досліджено алгоритми вирішення задач континуального лінійного програмування. Здійснена реалізація програми.

Отримані результати повністю відповідають темі дипломної роботи.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта дослідження або розробки. Не варто навіть намагатися заперечити, що математика займає надзвичайно важливе та особливе місце серед наук про світ. Ще зі шкільних років кожен з нас пам’ятає фразу «Математика – цариця наук», але не кожен з нас тоді коректно розумів цю фразу. Математика – це інструмент для пізнання світу, як і всі інші науки, вона була створена для виконання практичних завдань, поставлених людством. Та крім цього, математика структурує, знаходить зв’язки, об’єднує усі інші науки, даючи можливість описувати хімічні, фізичні та навіть суспільні процеси математичною мовою, яка буде зрозуміла для всіх, незалежно від їхньої належності до певної мовленнєвої групи.

Прикладна ж математиками, будучи окремим розділом математики, також є невід’ємною частиною наук про світ. Суть прикладної математики полягає в застосуванні математичних методів у різних галузях, таких як фізика, механіка, медицина, біологія, фінанси, бізнес, інформатика, інженерна справа тощо. Таким чином, прикладну математику можна назвати поєднанням математичної науки і спеціальних технічних знань [1].

Розвиваючись на сторінках робіт відомих науковців-математиків, таких як Чебишев, Ляпунов та Марков, прикладна математика не губить свою актуальність і на сьогоднішній день, охоплюючи проблеми від розробки алгоритмів шифрування до використання при побудові моделей нейронних мереж.

Одним з головних аспектів, яким також займається дисципліна прикладної математики, є теорія оптимізації, що знаходить своє широке використання в економіці, військовій справі, медицині тощо. Задачі, пов’язані з необхідністю визначення найбільш оптимального та ефективного варіанту розв’язку, є, напевно, найбільш поширеними задачами прикладної математики, а для їхнього

вирішення було розроблено математичні методи, що отримали загальну назву методів математичного програмування, і включають у себе методи нелінійного програмування, змішаного цілочисельного програмування, нелінійного програмування тощо [2].

У рамках написання даної кваліфікаційної роботи розглядаються методи континуального лінійного програмування, що є надбудовою (узагальненням) задач лінійного програмування, адже рішення у даному випадку шукається не на конкретному наборі дискретних величин, а на неперервній множині. Застосування методів континуального лінійного програмування є подібним до застосування методів лінійного програмування, з єдиним уточненням, що цільова функція та обмеження задачі задаються функціоналами зі скінченними або нескінченними границями інтегрування, лінійними відносно невідомої невід'ємної функції неперервного аргументу [3].

Актуальність роботи та підстави для її виконання. Не варто навіть намагатися заперечити, що задачі оптимізації мають неймовірно широке поле застосування. Та, разом із розвитком технологій та конкретних галузей науки та промисловості загалом, розміри та складності практичних проблеми оптимізації зростають, а питання моделювання, формулювання та розробки різних алгоритмів оптимізації стають ще більш затребуваними. Отже, дослідження у сфері теорії оптимізації, у конкретному випадку – сфера континуального моделювання, були і будуть актуальними.

Мета й завдання роботи. Метою роботи є дослідження методів рішення задач континуального лінійного програмування, а також розробка програми для вирішення конкретної задачі: вибір моменту пострілу у безшумній дуелі. Для цього необхідно:

1. Ознайомитися з методами вирішення задач континуального лінійного програмування.

2. Встановити необхідні модулі та компоненти.

3. Розробити програму для вирішення задачі про вибір моменту пострілу у безшумній дуелі шляхом побудови дискретного аналогу та методом послідовного покращення план-функції.

4. Порівняти результати роботи обох методів.

Об’єкт і методи дослідження або розроблення. Об’єктом розробки є процес створення програми для вирішення задачі континуального лінійного програмування. На вхід подаються цільова функція та обмеження, в даному випадку – функції щільностей.

Після виконання програми отримуємо рішення задачі двома методами.

Можливі сфери застосування. Значення важливості теорії оптимізації у сучасному світі важко переоцінити. Вони знаходять застосування у низці галузей, таких як економіка та управління, військова справа, інженерне проектування, медицина тощо. До того ж, як показує аналіз задач континуального лінійного програмування, до лінійних континуальних моделей зводиться величезна кількість задач управління в умовах невизначеності, розподілу безкінечних ресурсів, пошуку розподілів ймовірностей, що мають скінченні екстремальні властивості [3]. Також у рамках розгляду теорії задач континуального лінійного програмування особливе місце займають ігрові задачі на одиничному квадраті – так звані дуелі, що можуть враховувати ряд додаткових обмежень, що на них накладаються, задачі оцінювання інтервалів та наближення інтервалу невизначеності параметрів на довільний момент часу, задачі розподілу енергії радіолокаційних систем при огляді тощо.

Ще однією перевагою використання теорії континуального лінійного програмування є можливість формулювання і чисельного розв’язання на єдиній методологічній основі задач перевірки статистичних гіпотез, у тому числі з використанням континуального програмування стає можливим навіть перевірка

тих гіпотез та розв’язання тих задач, що в теорії статистичних рішень не розглядаються [3].

Таким чином, поле застосування розглянутих у рамках цієї роботи методів стає ще ширшим, захоплюючи найрізноманітніші задачі теорії надійності, теорії виявлення сигналів та розпізнавання образів.

Отже, враховуючи вище наведені приклади застосування задач континуального моделювання (список прикладів не є вичерпним), їхнє значення неможливо переоцінити, а методи, що були розглянути в основній частині кваліфікаційної роботи, можуть слугувати як науковою базою для подальшого поглибленого вивчення даних методологій, так і для використання у ряді конкретних задач як дослідницького, так і суто практичного характеру.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ КОНТИНУАЛЬНОГО ЛІНІЙНОГО ПРОГРАМУВАННЯ

Як і зазначалося вище, предметом дослідження даної роботи є методи континуального лінійного програмування.

Континуальне лінійне програмування є невід'ємною частиною теорії оптимізації та широко використовується у найрізноманітніших сферах людської діяльності від економіки та фінансів до логістики та виробництва. Так, наприклад, задачу континуального лінійного програмування можна використати для створення плану виробництва маючи за мету мінімізацію витрат або максимізацію прибутку при обмеженнях, що будуть враховувати наявні ресурси, потужності, матеріали тощо.

Задача континуального лінійного програмування є узагальненням задачі лінійного програмування, але, на відміну від лінійного програмування, розв'язок задачі шукається не на конкретному наборі дискретних величин, а на неперервній множині [7].

1.1 Задача лінійного програмування

Формулювання звичайної задачі лінійного програмування можна записати так [3]:

Знайти набір змінних $\{x_1, \dots, x_n\}$, за якого лінійна функція

$$L(x_1, x_2, \dots, x_n) = \sum_{j=1}^{n} c_j \cdot x_j \tag{1.1}$$

набуває екстремальних (максимального чи мінімального) значень, та задовільняє обмеження:

$$\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i, \;\; i = 1, 2, \dots, m; \tag{1.2}$$

$$x_j \geq 0, \qquad j = 1, 2, \ldots, n. \tag{1.3}$$

1.2 Задача континуального лінійного програмування

1.2.1 Загальна задача континуального лінійного програмування

Формулювання задачі (1.1)-(1.3) із попереднього пункту можна подати у більш узагальненому вигляді. Тепер задача буде задаватися так:

знайти вектор-функцію

$$X(t) = [x_1(t), x_2(t), \ldots, x_n(t)], \tag{1.4}$$

яка надає функціоналу

$$L(x_1(t), x_2(t), \ldots, x_n(t)) = \sum_{j=1}^{n} \int_{r_{1_j}}^{r_{2_j}} c_j(t) \cdot x_j(t) dt \tag{1.5}$$

екстремальних значень, та задовольняє обмеження

$$\sum_{j=1}^{n} \int_{r_{1_j}}^{r_{2_j}} a_{ij}(t) \cdot x_j(t) dt = b_i, \;\; i = 1, 2, \ldots, m, \tag{1.6}$$

$$x_j \geq 0, \qquad t \in \left[r_{1_j};\, r_{2_j}\right], \qquad j = 1, 2, \ldots, n, \tag{1.7}$$

де $c_j(t)$ та $a_{ij}(t)$ – інтегровані на $\left[r_{1_j};\, r_{2_j}\right]$ функції.

Таку задачу називатимемо загальною задачею континуального лінійного програмування [3].

1.2.2 Канонічна задача континуального лінійного програмування

Якщо у попереднього формулювання задачі (1.4)-(1.7) покласти $n = 1$, то ми отримаємо канонічну задачу континуального лінійного програмування, математичне формулювання якої виглядатиме так:

знайти функцію $x(t)$, яка надає функціоналу

$$L\big(x(t)\big) = \int_{r_1}^{r_2} c(t) \cdot x(t)dt \tag{1.8}$$

екстремального значення, та задовольняє обмеження:

$$\int_{r_1}^{r_2} a_i(t) \cdot x(t)dt = b_i, \qquad i = 1, 2, \ldots, m, \tag{1.9}$$

$$x(t) \geq 0, \qquad t \in [r_1;\ r_2]. \tag{1.10}$$

1.3 Поняття та властивості план-функцій

Довизначимо функцію $x(t)$:

$$x(t) = \xi(t) + \sum_{j=1}^{p} x_j \cdot \delta(t - t_j), \tag{1.11}$$

де $\xi(t)$ – функція неперервна на $[r_1;\ r_2]$,

$\delta(t - t_j)$ – дельта-функція Дірака.

Введемо такі означення:

$L\big(x(t)\big) = \int_{r_1}^{r_2} c(t) \cdot x(t)dt -$ цільовий функціонал,

$A^T(t)$ – функція умов,

$B^T(t)$ – вектор обмежень.

Якщо $x(t)$ задовольняє вищезазначені умови (1.9)-(1.10), то ця функція називається план-функцією, множину таких функцій позначимо за U. Якщо така функція надає функціоналу найбільше або найменше значення, то вона називатиметься оптимальною план-функцією, і буде розв’язком задачі континуального лінійного програмування.

Якщо план-функція $x(t)$ є крайньою точкою U, то вона називається опорною план-функцією.

Для кожної задачі континуального лінійного програмування можна побудувати двоїсту задачу:

знайти набір $\Lambda = \{\lambda_i\}$, що мінімізує функцію

$$Z(\Lambda) = \sum_{i=1}^{m} b_i \cdot \lambda_i \tag{1.12}$$

за умов

$$\sum_{i=1}^{m} \lambda_i \cdot a_i(t) \geq c(t). \tag{1.13}$$

1.3.1 Властивості задач континуального лінійного програмування

Також план-функції задач континуального лінійного програмування мають такі властивості [3]:

- Множина план-функцій є випуклою.
- Якщо

$$x(t) = \sum_{j=1}^{m} x_j \cdot \delta(t - t_j) \tag{1.14}$$

є план-функцією задачі, то вона опорна.

- Нехай $x(t)$ – опорна, тоді $\xi(t) = 0$, а сама план-функція набуває вигляду (1.14).
- Якщо задача (1.8)-(1.10) має єдине рішення, то воно досягається на опорній план-функції.
- Якщо задачу (1.8)-(1.10) можливо розв'язати, то у множині розв'язків цієї задачі існує $x(t)$, яке є крайньою точкою U .
- Якщо $x(t)$ – довільна план-функція прямої (1.8)-(1.10) задачі, Λ – довільний план двоїстої задачі (1.12)-(1.13), то

$$L\big(x(t)\big) = \int_{r_1}^{r_2} c(t) \cdot x(t)dt \leq \sum_{j=1}^{m} b_j \cdot \lambda_j = \tilde{Z}(\Lambda). \quad (1.15)$$

- Якщо $x(t)$ – план-функція прямої задачі (1.8)-(1.10), Λ – план двоїстої задачі (1.12)-(1.13), та

$$\int_{r_1}^{r_2} c(t) \cdot x^*(t)dt \leq \sum_{j=1}^{m} b_j \cdot \lambda_j^*, \quad (1.16)$$

то $x(t)$ та Λ є рішеннями відповідних прямої та двоїстої задач.

- Якщо цільовий функціонал прямої задачі (1.8)-(1.10) континуального лінійного програмування не обмежений зверху, то двоїста задача не має допустимих планів. Те ж саме працює і навпаки.
- Якщо одна задача з двоїстої пари має рішення, то і друга теж, причому значення цільових функцій рівні.

1.3.2 Критерій оптимальності план-функції задачі континуального лінійного програмування

Для оптимальності план-функції необхідне й достатнє існування набору $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ такого, що

$$\sum_{i=1}^{m} a_i(t) \cdot \lambda_i \geq c(t), \;\; t \in [r_1;\, r_2]; \quad (1.17)$$

$$\sum_{j=1}^{m} a_i\big(t_j^*\big) \cdot \lambda_j = c\big(t_j^*\big), \;\; j = 1, 2, \ldots, m. \quad (1.18)$$

Наслідок:

План-функція є оптимальною, якщо

$$\max_{r_1 \leq t \leq r_2} \left\{ c(t) - \sum_{j=1}^{m} a_j(t) \cdot \lambda_j \right\} \leq 0\,. \quad (1.19)$$

1.4 Канонічна задача континуального лінійного програмування з двосторонніми обмеженнями

Загальною задачею континуального лінійного програмування з двосторонніми обмеженнями на план-функції називатимемо задачу, яка формулюється так:

знайти функцію $y^*(t)$, що максимізує функціонал

$$Z(y) = \int_{r_1}^{r_2} g(t) \cdot y(t) dt \tag{1.20}$$

та задовільняє умови:

$$\int_{r_1}^{r_2} f_i(t) \cdot y(t) dt = S_i, \tag{1.21}$$

$$d^-(t) \leq y(t) \leq d^+(t), \tag{1.22}$$

де $g(t)$ – відома функція, обмежена на інтервалі $[r_1, r_2]$,

$f_i(t)$ – система лінійно незалежних функцій,

$d^-(t), d^+(t)$ – обмежені на інтервалі $[r_1, r_2]$ функції,

$$d^-(t) < d^+(t) \in [r_1, r_2], \tag{1.23}$$

S_i – невід'ємні числа

Множину точок, які належать закритому інтервалу $[r_1, r_2]$ позначимо через R[3].

Введемо функцію

$$x(t) = \frac{y(t) - d^-(t)}{d^+(t) - d^-(t)}, \tag{1.24}$$

і трансформуємо вище описану задачу (1.20)-(1.22) у наступну форму:

знайти функцію $x^*(t)$, що максимізує

$$Z(x) = \int_{r_1}^{r_2} c(t) \cdot x(t) dt \tag{1.25}$$

та задовольняє обмеження:

$$\int_{r_1}^{r_2} a_1(t) \cdot x(t) dt = b_1, \;\; i \in M, \tag{1.26}$$

$$0 \leq x(t) \leq 1, \tag{1.27}$$

де

$$c(t) = g(t)[d^+(t) - d^-(t)], \qquad t \in R, \tag{1.28}$$

$$a_1(t) = f_1(t)[d^+(t) - d^-(t)], \qquad t \in R, \qquad i \in M, \tag{1.29}$$

$$b_i = S_i - \int_{r_1}^{r_2} f_i(t) d^-(t) dt, \qquad t \in M, \tag{1.30}$$

$$b_i \leq \int_{r_1}^{r_2} a_i(t) dt, \qquad i \in M. \tag{1.31}$$

Цю задачу назвемо канонічною задачею континуального лінійного програмування з двосторонніми обмеженнями. Дана модель відрізняється від звичайної канонічної задачі континуального лінійно програмування тим, що разом з вимогами невід'ємності, значення шуканої функції обмежуються зверху.

Задачі континуального лінійного програмування з двосторонніми обмеженнями мають (1.25)-(1.27) такі властивості:

- План-функція задачі континуального лінійного програмування з двосторонніми обмеженнями є опорною тоді й тільки тоді, коли вона має вигляд

$$x_0(t) = \begin{cases} 1, & t \in \tau_1, \\ 0, & t \in \tau_0, \end{cases} \tag{1.32}$$

де $\tau_1 \subset R,\ \tau_2 \subset R,\ \tau_1 \cap \tau_2 = \emptyset,\ \tau_1 \cup \tau_2 = R.$

- Нехай $x(t)$ – довільна план-функція прямої задачі, а $\Lambda^T = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ та $\lambda(t)$ – довільні план-вектор та план-функція двоїстої задачі. Тоді

$$Z(x) = \int_{r_1}^{r_2} c(t)x(t)dt \le \sum_{i=1}^{m} b_i\lambda_i + \int_{r_1}^{r_2} \lambda(t)d(t) = \hat{Z}(\Lambda, \lambda). \tag{1.33}$$

Нехай для план-функцій $x^*(t)$ і $\lambda^*(t)$ та план-вектора $\Lambda^T = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ має місце рівність:

$$\int_{r_1}^{r_2} c(t)x^*(t) = \sum_{i=1}^{m} b_i\lambda_i + \int_{r_1}^{r_2} \lambda^*(t). \tag{1.34}$$

Тоді план-функція $x^*(t)$ і набір $\{\Lambda^*, \lambda^*(t)\}$ будуть розв'язками відповідних прямої та двоїстої задач

- Для оптимальності опорної план-функції достатньо існування вектора $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$, який би задовольняв умови:

$$\sum_{i=1}^{m} a_i(t)\lambda_i < c(t), \qquad t \in \{t : x^*(t) > 0\}, \tag{1.35}$$

$$\sum_{i=1}^{m} a_i(t)\lambda_i \ge c(t), \qquad t \in \{t : x^*(t) = 0\}. \tag{1.36}$$

1.5 Наближені методи розв'язування задачі континуального лінійного програмування

Розглянемо два найпоширеніші методи для знаходження наближеного розв'язку задачі континуального програмування.

1.5.1 Дискретизація

Метод дискретизації задачі континуального лінійного програмування включає наступні кроки:

1. Розбити проміжок $[r_1;\ r_2]$ на $n \geq m$ підінтервалів $[t_0;\ t_1], [t_1;\ t_2], \ldots, [t_{n-1};\ t_n]$, де $r_1 = t_1 < t_2 < \ldots < t_n = r_2$. Як бачимо, інтервали не перетинаються та є суміжними.

2. Визначимо клас функцій $x(t)$ наступним чином:

$$x(t) = \begin{cases} x_1, & t \in [t_0, t_1], \\ x_2, & t \in (t_1, t_2], \\ & \ldots \\ x_n, & t \in (t_{n-1}, t_n]. \end{cases} \tag{1.37}$$

Після виконаних маніпуляцій канонічна задача континуального програмування (1.20)-(1.22) може бути записана у вигляді звичайної задачі лінійного програмування. Запишемо цю задачу.

Знайти такий набір $x^* = \{x_j^*\}$, що максимізує функцію

$$Z(x) = \sum_{j=1}^{n} c_j \cdot x_j \tag{1.38}$$

та задовольняє умови

$$\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i,\ \ i = 1, 2, \ldots, m; \tag{1.39}$$

$$x_j \geq 0, \qquad j = 1, 2, \ldots, n, \tag{1.40}$$

і де

$$c_j = \int_{t_{j-1}}^{t_j} c(t)dt, \tag{1.41}$$

$$a_{ij} = \int_{t_{j-1}}^{t_j} a_i(t)dt\,, i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, n. \tag{1.42}$$

Розв'язок $x^* = \{x_j^*\}$ визначає функцію $x(t)$, що при достатньо великому значення n може прирівнюватися до розв'язку початкової континуальної задачі.

Даний підхід використовується тільки за умови, що задача лінійного програмування (1.38)-(1.40), яка утворюється, є коректною. Точність даного методу залежить від довжини підінтервалу $\Delta t = \frac{r_2 - r_1}{n}$.

1.5.2 Вирішення задачі континуального лінійного програмування у заданому класі інтегрованих функцій

Введемо клас інтегрованих функцій $x(t)$, який можна однозначно визначити за допомогою набору параметрів $x_1, x_2, \ldots, x_m$

$$x(t) = f(x_1, x_2, \ldots, x_m, t) = f(x, t). \tag{1.43}$$

Припущення, щодо належності функції $x(t)$ до параметричного сімейства функцій приводить сформовану вище задачу континуального лінійного програмування (1.20)-(1.22) до такої задачі нелінійного математичного програмування:

знайти такий набір $X = \{x_1, x_2, \ldots, x_m\}$, що буде максимізувати функцію

$$F(x) = \int_{r_1}^{r_2} c(t) \cdot f(x,t)dt \tag{1.44}$$

та задовольняти умови

$$\int_{r_1}^{r_2} a_i(t) \cdot f(x,t)dt = b_1, \qquad f(x,t) \geq 0. \tag{1.45}$$

Процес розв'язання задачі можна значно спростити, обравши за функцію $f(x,t)$ лінійну відносно змінних $(x_1, x_2, \ldots, x_m)$, тобто

$$f(x,t) = \sum_{j=1}^{n} x_j \cdot f_j(t), \tag{1.46}$$

де $f_1(t), f_2(t), \ldots, f_n(t)$ – деяка система інтегрованих функцій, яку ми називатимемо базисною. При цьому початкова задача набуває вигляду:

$$F(x) = \sum_{j=1}^{n} \int_{r_1}^{r_2} c(t) \cdot f_j(t)dt \to \max, \tag{1.47}$$

$$\sum_{j=1}^{n} \int_{r_1}^{r_2} a_i(t) \cdot f_j(t)dt = b_i, \quad i = 1,2,\ldots,m, \tag{1.48}$$

$$\sum_{j=1}^{n} x_j \cdot f_j(t) \geq 0. \tag{1.49}$$

А якщо, до того ж, ще й накласти умову $x_j \geq 0$, то ми отримаємо звичайну задачу лінійного програмування такого вигляду:

знайти такий набір $X = \{x_1, x_2, \ldots, x_m\}$, що максимізує функцію

$$F(x) = \sum_{j=1}^{n} c_j \cdot x_j \tag{1.50}$$

і задовольняє обмеження:

$$\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i, \qquad i = 1,2,\dots,m; \tag{1.51}$$

$$x_j \geq 0, \qquad j = 1,2,\dots,n, \tag{1.52}$$

де

$$c_j = \int_{r_1}^{r_2} c(t) \cdot f_j(t)dt\,, \quad j = 1,2,\dots,n, \tag{1.53}$$

$$a_{ij} = \int_{r_1}^{r_2} a_i(t) \cdot f_j(t)dt\,, \qquad i = 1,2,\dots,m, \qquad j = 1,2,\dots,n. \tag{1.54}$$

Ефективність застосування даного методу досить суттєво залежить від того, наскільки вдало було підібрано набір базисних функцій, однак простота цього методу в деяких випадках може бути більш важливою за точність.

1.6 Метод послідовного покращення план-функції

Метод послідовного покращення план-функції бере за основу викладені вище властивості план-функцій. Як і його аналог для задач лінійного програмування (метод послідовного покращення плану) метод є ітераційним.

Нехай на $(q-1)$ ітерації отримана така план-функція має вигляд:

$$x^{(q-1)}(t) = \sum_{j=1}^{m} x_j^{(q-1)} \cdot \delta(t - t_j^{(q-1)}). \tag{1.55}$$

На першому кроці q-ї ітерації проходить перевірка план-функції, і або підтверджується знайдене рішення, або з’ясовується неоптимальність план-функції, і на другому кроці отримується нова, більш оптимальна план-функція.

Перший крок

Перш за все потрібно перевірити план-функцію на оптимальність. Для цього розв'яжемо систему алгебраїчних рівнянь:

$$\sum_{i=1}^{m} a_i\left(t_j^{(q-1)}\right)\cdot\lambda_i = c(t_j^{(q-1)}). \tag{1.56}$$

Нехай $\Lambda^{(q-1)} = \{\lambda_1^{(q-1)}, \dots, \lambda_m^{(q-1)}\}$ – розв'язок цієї системи. Згідно з вище викладеними властивостями, план-функція оптимальна якщо

$$c^{(q)}(t) = c^{(q-1)}(t) - \sum_{i=1}^{m} a_i(t)\lambda_i^{(q-1)} > 0, \tag{1.57}$$

інакше переходимо до другого кроку

Другий крок

Нову опорну план-функцію отримаємо шляхом заміни одного з векторів у базисному наборі, $\left\{A\left(t_1^{(q-1)}\right), \dots, A\left(t_m^{(q-1)}\right)\right\}$, отриманому на попередній ітерації, на новий вектор, відповідний точці $\check{t}$, для якої, власне, не справдились умови оптимальності:

$$c^{(q)}(\check{t}) = c^{(q-1)}(\check{t}) - \sum_{i=1}^{m} a_1(\check{t})\lambda_i^{(q-1)} \le 0. \tag{1.58}$$

Так як $x^{(q-1)}(t)$ – опорна план-функція, то вектор $A(t^0)$ може бути єдиним чином розкладено по векторам, тому

$$\sum_{i=1}^{m} a_i\left(t_j^{(q-1)}\right)x_j^0 = a_i(t^0). \tag{1.59}$$

Розглянемо отриманий вектор X^0 Якщо всі коефіцієнти у розкладі від'ємні, то цільовий функціонал не обмежений зверху, і розв'язку немає. Якщо ж це не так, то може бути побудована нова план-функція, яка буде надавати функціоналові значення більше, ніж на попередній ітерації. Для цього обчислимо

$$\theta_0^{(q)} = \min_{j, x_j^0 > 0} \frac{x_j^{(q-1)}}{x_j^0} = \frac{x_l^{(q-1)}}{x_l^0} > 0. \tag{1.60}$$

Вектор $A\left(t_l^{(q-1)}\right)$ тепер може бути виключено з базису $\left\{A\left(t_1^{(q-1)}\right), \ldots, A\left(t_{l-1}^{(q-1)}\right), A\left(t_{l+1}^{(q-1)}\right), \ldots, A\left(t_m^{(q-1)}\right),\ A(t^0)\right\}$.

Тепер підставимо

$$A\left(t_l^{(q-1)}\right) = \frac{1}{x_l^0}\left[A(t^0) - \sum_{j \neq l} A\left(t_j^{(q-1)}\right) x_j^0\right] \tag{1.61}$$

до

$$\sum_{j=1}^{m} A\left(t_j^{(q-1)}\right) x_j^{(q-1)} = B, \tag{1.62}$$

і отримаємо

$$\begin{aligned} &\sum\nolimits_{j \neq l} A\left(t_j^{(q-1)}\right) x_j^{(q-1)} + \frac{x_l^{(q-1)}}{x_l^0}\left[A(t^0) - \sum_{j \neq l}^{m} A\left(t_j^{(q-1)}\right) x_j^0\right] = \\ &= \sum_{j \neq l}\left(x_j^{(q-1)} - \frac{x_l^{(q-1)}}{x_l^0} x_j^0\right) A\left(t_j^{(q-1)}\right) + \frac{x_l^{(q-1)}}{x_l^0} A(t^0) = B. \end{aligned} \tag{1.63}$$

Таким чином, отримаємо нову опорну план-функцію вигляду:

$$t_j^{(q)} = \begin{cases} t_j^{(q-1)}, & j \neq l, \\ t^0, & j = l; \end{cases} \tag{1.64}$$

$$x_j^{(q)} = \begin{cases} x_j^{(q-1)} - \theta_0^{(q)} x_j^0, & j \neq l, \quad \theta_0^{(q)} = \dfrac{x_l^{(q-1)}}{x_l^0}, \\ \theta_0^{(q)}, & j = l. \end{cases} \tag{1.65}$$

та значення цільового функціоналу, яке буде більшим, ніж на попередній ітерації

$$Z^{(q)}\left(x^{(q)}\right) - Z^{(q-1)}\left(x^{(q-1)}\right) = \theta_0^{(q)}\left(c(t^0) - \Lambda^{(q-1)} A(t^0)\right) = \tag{1.66}$$

$$= \theta_0^{(q)} c^{(q)}(t_0) > 0.$$

На відміну від симплекс-методу, кількість ітерацій у методі послідовного покращення план-функції може бути нескінченною. Але гарна новина полягає у тому, що значення цільового функціоналу обмежене зверху, тому послідовність значень цільового функціоналу має границю. Тому ітераційну процедуру можна обірвати, щойно різниця послідовних двох значень цільових функціоналів стане меншою за задане число.

Залишилась одна проблема. Збіжність описаної ітераційної процедури істотно залежить від обраної початкової опорної план-функції. Наприклад, якщо взяти випадковим чином з інтервалу $[r_1, r_2]$ m точок, і розв'язати систему рівнянь:

$$\begin{gathered} x_1 a_1\left(t_1^{(0)}\right) + x_2 a_1\left(t_2^{(0)}\right) + \cdots + x_m a_1\left(t_m^{(0)}\right) = b_1, \\ x_1 a_2\left(t_1^{(0)}\right) + x_2 a_2\left(t_2^{(0)}\right) + \cdots + x_m a_2\left(t_m^{(0)}\right) = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_1 a_m\left(t_1^{(0)}\right) + x_2 a_m\left(t_2^{(0)}\right) + \cdots + x_m a_m\left(t_m^{(0)}\right) = b_m, \end{gathered} \tag{1.67}$$

то відсутність у рішенні від'ємних значень x_j зовсім не гарантована. Це означає, що обирати навмання точки $\{t_j^{(0)}\}$ і обчислювати відповідні їм значення $\{x_j\}$ – це не найкращий спосіб знаходження початкової опорної план-функції задачі. Тому розглянемо один алгоритм формування початкової опорної план-функції.

Алгоритм формування початкової опорної план-функції.

У ході даного алгоритму послідовно вишукуватимемо компоненти $(x_1, \ldots, x_m)$ і $(t_1, \ldots, t_m)$ початкової опорної план-функції

$$x(t) = \sum_{j=1}^{m} x_j \delta(t - t_j). \tag{1.68}$$

Ідея методу полягає в тому, що для нашої план-функції задача континуального лінійного програмування зводиться до дискретної, тобто до знаходження наборів $\underline{X} = (x_1, \dots, x_m)$, $\underline{T} = (t_1, \dots, t_m)$, які б максимізували функцію

$$L(\underline{X}, \underline{T}) = \sum_{j=1}^{m} x_j c(t_j) \tag{1.69}$$

з такими обмеженнями:

$$\sum_{j=1}^{m} x_j a_i(t_j) = b_i, \tag{1.70}$$

$$x_j \geq 0, t_j \in [r_1, r_2]. \tag{1.71}$$

Така задача розв'язується за допомогою ітераційної процедури, з кожним кроком якої набори X та T довизначаються, при цьому щодо їхніх компонент ставляться такі умови:

1. Отримувані набори X та T повинні залишатись у класі допустимих.
2. Обмеження, які попередньо виконувались, не можуть бути порушені.
3. Довизначені компоненти мають бути обрані таким чином, щоб забезпечити найбільший можливий приріст цільової функції.

Наведемо приклад такої ітераційної процедури, яка б задовольняла вказані вище обмеження.

Нехай p кроків послідовного покращення векторів рішення вже позаду, і в результаті ми наразі маємо набори $\underline{X}^{(p)}$ та $\underline{T}^{(p)}$, які містять p ненульових компонентів та план-функцію

$$x^{(p)}(t) = \sum_{j=1}^{p} x_j^{(p)} \delta(t - t_j). \tag{1.72}$$

Позначимо

$N^{(p)}$ – множина індексів ненульових компонент векторів розв'язку;

M – множина індексів обмежень;

$T^{(p)}$ – множина значень t, що відповідають ненульовим значенням план-функції $x^{(p)}(t)$:

$$L^{(p+1)} = \left\{ i: i \in M, \sum_{j \in M} x_j^{(p)} a_i^{(p)}(t_j) = b_i^{(p)} \right\}, \tag{1.73}$$

$$\overline{L}^{(p+1)} = \left\{ i: i \in M, \sum_{j \in M} x_j^{(p)} a_i^{(p)}(t_j) < b_i^{(p)} \right\}. \tag{1.74}$$

Обчислимо

$$t_0^{(p+1)} = arg \max_{t \in R^{(p)}} \left\{ c^{(p)}(t) \min_{\substack{i \in \overline{L}^{(p+1)} \\ a_i^{(p)}(t) > 0}} \left\{ \frac{b_i^{(p)}}{a_i^{(p)}(t)} \right\} \right\}, \tag{1.75}$$

$$R^{(p)} = \left\{ t: t \in R, \min_{i \in \overline{L}^{(p+1)}} \left\{ \frac{b_i^{(p)}}{a_i^{(p)}(t)} \right\} < \min_{i \in L^{(p+1)}} \left\{ \frac{b_i^{(p)}}{a_i^{(p)}(t)} \right\} \right\}. \tag{1.76}$$

де $R^{(p)}$ – область допустимих значень t після проведення p ітерацій, у відповідності з якою при виборі $t_0^{(p-1)}$ вживаються необхідні заходи для того, щоб не були порушені вже виконані обмеження.

Нехай k – номер обмеження, для якого виконується (1.75). Тоді покращення план-функції полягає у додаванні компоненти $t_0^{(p+1)}$ до множини $T^{(p)}$ та компонента

$$x_{p+1} = \frac{b_k^{(p)}}{a_k^{(p)}\left(t_0^{(p+1)}\right)} \tag{1.77}$$

до множини $N^{(p)}$.

Вибір однорозмірного покращення забезпечує виконання вимог 1 та 3. Для виконання вимоги 2 необхідно перетворити рівняння обмежень за схемою повного виключення, взявши за направляючий елемент $a_k^{(p)}\left(t_0^{(p+1)}\right)$. При цьому

$$a_i^{(p+1)}(t) = a_i^{(p)}(t) - a_k^{(p)}(t)\frac{a_i^{(p)}\left(t_0^{(p+1)}\right)}{a_k^{(p)}\left(t_0^{(p+1)}\right)}, i \neq k; \tag{1.78}$$

$$a_k^{(p+1)}(t) = \frac{a_k^{(p)}(t)}{a_k^{(p)}\left(t_0^{(p+1)}\right)}; \tag{1.79}$$

$$b_i^{(p+1)}(t) = b_i^{(p)}(t) - b_k^{(p)}(t)\frac{a_i^{(p)}\left(t_0^{(p+1)}\right)}{a_k^{(p)}\left(t_0^{(p+1)}\right)}, i \neq k; \tag{1.80}$$

$$b_k^{(p+1)}(t) = \frac{b_k^{(p)}}{a_k^{(p)}\left(t_0^{(p+1)}\right)}, \tag{1.81}$$

а нова часткова план-функція матиме вигляд

$$x^{(p+1)}(t) = \sum_{j=1}^{p} x_j^{(p)}\delta\left(t - t_j\right) + x_{p+1}\delta\left(t - t_0^{(p+1)}\right). \tag{1.82}$$

З метою виключення впливу виконаного обмеження за тією ж схемою перетворюється цільова функція:

$$c^{(p+1)}(t) = c^{(p)}(t) - a_k^{(p)}(t)\,\frac{c^{(p)}\left(t_0^{(p+1)}\right)}{a_k^{(p)}\left(t_0^{(p+1)}\right)}. \qquad (1.83)$$

Після проведення m кроків ми матимемо початкову опорну план-функцію

$$x(t) = \sum_{j=1}^{m} x_j \delta\left(t - t_j\right). \qquad (1.84)$$

РОЗДІЛ 2. ЗАДАЧА ВИБОРУ МОМЕНТІВ ВІДКРИТТЯ ВОГНЮ

Для демонстрації алгоритмів вирішення задач континуального лінійного програмування оберемо задачу вибору моменту відкриття вогню у безшумній дуелі. Спочатку розглянемо простіший варіант постановки такої задачі.

2.1 Проста задача вибору моменту відкриття вогню у безшумній дуелі

Нехай двоє дуелянтів наближаються один до одного, кожен з них має всього один шанс вистрілити. Перед кожним з дуелянтів стоїть задача обрати найоптимальніший момент для пострілу. Складність вибору полягає у тому, що, хоча з наближенням і збільшується шанс вразити суперника, але так само збільшується і вірогідність бути ураженим. Дуелянти не чують пострілу свого суперника (тому дуель і називають безшумною), і відповідно вибір моменту пострілу не залежить від моменту пострілу опонента.

Нехай $p_1(t_1)$ – ймовірність ураження другого дуелянта першим, якщо перший вистрілить у момент часу t_1, $p_2(t_2)$ – ймовірність ураження першого дуелянта другим, якщо той вистрілить у момент часу t_2.

Тоді умовна ймовірність виживання для першого дуелянта буде обчислюватись за формулою:

$$M_1(t_1, t_2) = \begin{cases} 1 - P_2(t_2), & \text{якщо } t_2 \leq t_1, \\ 1 - P_2(t_2) \cdot \big(1 - P_1(t_1)\big), & \text{якщо } t_2 > t_1. \end{cases} \quad (2.1)$$

Дана формула враховує те, що дуель є безшумною і дуелянти не знають моменту пострілу свого противника.

Нехай $f_1(t_1)$ – щільність розподілу випадкового моменту відкриття вогню першим дуелянтом, $f_2(t_2)$ – щільність розподілу випадкового моменту відкриття вогню другим учасником дуелі.

В такому випадку математичне сподівання ймовірності виживання першого дуелянта дорівнює:

$$M_1 = \int_0^{\infty} \int_0^{\infty} M_1(t_1, t_2) \cdot f_1(t_1) \cdot f_2(t_2)\, dt_1 dt_2. \tag{2.2}$$

З властивостей функції щільностей розподілу випадкової величини маємо:

$$\int_0^{\infty} f_1(t_1) dt_1 = 1, \qquad f_1(t_1) \geq 0, \qquad t_1 \in [0, \infty), \tag{2.3}$$

$$\int_0^{\infty} f_2(t_2) dt_2 = 1, \qquad f_2(t_2) \geq 0, \qquad t_2 \in [0, \infty). \tag{2.4}$$

Наведені вище співвідношення дають достатню кількість інформації для того, щоб сформулювати задачу з вибору моменту відкриття вогню [3]. Наприклад, задачу з вибору моменту часу, постріл в який забезпечить першому дуелянтові максимальну ймовірність не бути враженим у результаті дуелі, за умови, що відома функція щільності розподілу $f_2(t_2)$ другого дуелянта.

Математичне формулювання даної задачі матиме такий вигляд:

знайти функцію $f_1(t_1)$, що максимізує функціонал

$$P_1(f_1) = \int_0^{\infty} c_1(t_1) \cdot f_1(t_1) dt_1, \tag{2.5}$$

де

$$c_1(t_1) = \int_0^{\infty} M_1(t_1, t_2) \cdot f_2(t_2)\, dt_2\,, \tag{2.6}$$

при цьому задовольняючи умови:

$$\int_0^{\infty} f_1(t_1) dt_1 = 1, \qquad f_1(t_1) \geq 0, \qquad t_1 \in [0, \infty), \tag{2.7}$$

$$\int_0^{\infty} f_2(t_2) dt_2 = 1, \qquad f_2(t_2) \geq 0, \qquad t_2 \in [0, \infty). \tag{2.8}$$

Описана вище модель демонструє «обережну» тактику першого дуелянта, проте існує й інший підхід. Полягає він у тому, щоб поставити у пріоритет не власне виживання, а ураження ворога. У цьому випадку умовна ймовірність враження супротивника першим дуелянтом, за умови, що той виконує постріл у момент часу t_2, буде дорівнювати

$$N_1(t_1, t_2) = \begin{cases} P_1(t_1), & \text{якщо } t_1 \leq t_2, \\ \big(1 - P_2(t_2)\big) \cdot \big(P_1(t_1)\big), & \text{якщо } t_1 > t_2. \end{cases} \quad (2.9)$$

За тієї умови, що стратегію другого дуелянта відомо, ризикована стратегія першого дуелянта у формі задачі континуального лінійного програмування матиме вигляд:

знайти функцію $f_1(t_1)$, що максимізує функціонал

$$P_1(f_1) = \int_0^\infty c_2(t_1) \cdot f_1(t_1) dt_1, \quad (2.10)$$

де

$$c_2(t_1) = \int_0^\infty N_1(t_1, t_2) \cdot f_2(t_2)\, dt_2\,, \quad (2.11)$$

та задовольняє умови (2.7), (2.8).

2.2 Узагальнена задача вибору моменту відкриття вогню у безшумній дуелі

У даній роботі ми розглянемо узагальнення наведеної у підрозділі 2.1 задачі. Тепер один дуелянт буде у ролі нападника, а другий – захисника, якому потрібно захистити певну кількість об’єктів від атаки противника.

Нехай для кожного об’єкту маємо функцію щільності випадкового моменту відкриття по ньому вогню противником, який атакує: $f_i(t)$, $i = 1, 2, \ldots, m,$ де m – кількість об’єктів, які необхідно захистити. Також відомий розподіл ймовірностей $p_1, p_2, \ldots, p_m$ ураження об’єктів тим дуелянтом,

що атакує. Тоді безумовна щільність розподілу моменту відкриття вогню нападником по об'єктах матиме такий вигляд:

$$f_0(t) = \sum_{i=1}^{m} p_i \cdot f_i(t), \qquad \sum_{i=1}^{m} p_i = 1. \tag{2.12}$$

Тепер введемо ймовірність враження нападника $c(t)$; b_i – необхідні ймовірності збереження об'єктів $(i = 1, 2, \ldots, m)$: вони залежать від важливості об'єктів; а також ймовірності збереження об'єктів $a_i(t)$ як функції моменту здійснення пострілу тим, хто захищається: $a_i(t) = (1 - p_i) \cdot f_i(t)$, $i = 1, 2, \ldots, m$.

Таким чином, отримаємо таке формулювання задачі:

знайти функцію $f(t)$, яка максимізує функціонал

$$P(t) = \int_{-\infty}^{\infty} c(t) \cdot f(t) dt, \tag{2.13}$$

задовольняючи умови

$$\int_{-\infty}^{\infty} a_i(t) \cdot f(t) dt \geq b_i, \qquad i = 1, 2, \ldots, m, \tag{2.14}$$

$$f(t) \geq 0, \quad t \in (-\infty; +\infty). \tag{2.15}$$

РОЗДІЛ 3. МЕТОДИ РЕАЛІЗАЦІЇ УЗАГАЛЬНЕНОЇ ЗАДАЧІ ВИБОРУ МОМЕНТУ ВІДКРИТТЯ ВОГНЮ

Для розробки програми було обрано мову програмування Python. Також було використано такі бібліотеки як Numpy, Scipy та Sympy.

В якості прикладу, було взято задачу з двома об'єктами, які треба захищати. Функції щільності розподілу випадкового моменту пострілу по об'єктам:

$$f_1(t) = \frac{1}{5}, \tag{3.1}$$

$$f_2(t) = \frac{4t^3}{625}. \tag{3.2}$$

Функція щільності розподілу випадкового моменту пострілу по противнику, що атакує:

$$f(t) = \frac{t}{12,5}. \tag{3.3}$$

В якості необхідних мінімальних ймовірностей збереження об'єктів візьмемо такі значення:

$$b_1 = 0,3; \quad b_2 = 0,7. \tag{3.4}$$

Нехай ймовірності нанесення удару по об'єктам будуть однаковими:

$$p_1 = 0,5; \quad p_2 = 0,5. \tag{3.5}$$

3.1 Огляд технологій, що використовувалися

Мова програмування Python була обрана для реалізації алгоритмів через свою простоту, гнучкість та широкі можливості. Для полегшення розробки та виконання різних обчислювальних завдань, були використані деякі бібліотеки, такі як NumPy, SciPy та SymPy.

Розглянемо детальніше кожну з цих бібліотек.

NumPy [4] є однією з основних бібліотек для обробки числових даних у Python. Вона надає потужні функції для роботи з багатовимірними масивами і матрицями. Основна перевага NumPy полягає у тому, що вона оптимізована для швидкої виконання операцій на масивах, що робить її ідеальним вибором для наукових обчислень. Крім того, NumPy надає багатий набір математичних функцій, що значно спрощує вирішення алгебраїчних рівнянь та інших математичних задач.

Бібліотека *SciPy* [6] є розширенням NumPy і надає багато додаткових функцій для наукових обчислень. Вона включає інструменти для інтеграції, градієнтної оптимізації, розв’язання звичайних диференціальних рівнянь, паралельного програмування та іншого. SciPy забезпечує високорівневі інтерфейси для багатьох складних обчислювальних задач і є потужним інструментом для наукових досліджень та інженерних розрахунків.

SymPy [5] є символьною математичною бібліотекою, яка надає можливості символьних обчислень в Python. Вона дозволяє виконувати алгебраїчні операції, диференціювання, інтегрування, розв’язування рівнянь та багато іншого за допомогою символьних виразів. SymPy використовується для математичного аналізу, символьного моделювання, дискретної математики, квантової фізики та інших галузей науки.

Використання цих бібліотек значно спрощує реалізацію складних обчислювальних алгоритмів у Python. Вони надають широкий набір функцій та

інструментів, які дозволяють здійснювати чисельні розрахунки, розв'язувати математичні задачі та виконувати символьні обчислення з високою точністю та ефективністю.

3.2 Метод 1: дискретизація задачі континуального лінійного програмування

Загальний алгоритм можна описати так:

1. Розбиваємо проміжок на потрібну кількість інтервалів n (рис. 1).
2. Обраховуємо

$$c_j = \int_{t_{j-1}}^{t_j} c(t)dt \tag{3.6}$$

та

$$a_{ij} = \int_{t_{j-1}}^{t_j} a_i(t)dt \tag{3.7}$$

на заданих проміжках.

Розв'язуємо звичайну задачу лінійного програмування з цільовою функцією

$$Z(x) = \sum_{j=1}^{n} c_j \cdot x_j \tag{3.8}$$

та умовами

$$\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i, \quad i = 1, 2, \ldots, m; \tag{3.9}$$

$$x_j \geq 0, \qquad j = 1, 2, \ldots, n, \tag{3.10}$$

за допомогою модуля scipy.optimize. У тому прикладі, який наразі реалізується, $m = 2$.

```
def discrete(cdf, b, d, p):
    A = []
    b = -b
    c = len(cdf[0]) // d
    for g in cdf:
        temp = []
        for i in range(d - 1):
            temp.append((1 - p[i]) * integral(g, i * c, (i + 1) * c) * -1)
        A.append(temp)
    obj = A[0].copy()
    a = [1] * (d - 1)
    b1 = [1]
    A.pop(0)
    bnd = [(0, float("inf"))] * len(obj)
    opt = linprog(c=obj, A_ub=A, b_ub=b, A_eq=[a], b_eq=b1, bounds=bnd, method="interior-point")
    res = [opt.x, opt.fun]
    return res
```

Рисунок 1. Функція discrete: реалізує алгоритм дискретизації континуальної задачі лінійного програмування

Результати обчислень з різними інтервалами розбиття:

```
Z = 1.053333333 кількість проміжків 5
Z = 1.233 кількість проміжків 20
Z = 1.25 кількість проміжків 100
Z = 1.25332 кількість проміжків 500
```

3.3 Метод 2: послідовне покращення план-функції

Опишемо загальний алгоритм реалізації даного методу.

Підготовчий етап:

1. Обчислення початкових параметрів план-функції $t_j,\ x_j$.
2. Обчислення значення цільової функції $Z\big(x(t)\big)$.
3. Обчислення початкових множників Λ.

Основний етап:

1. Визначити

$$\max_{r\epsilon[t_1,t_2]}\left\{c(t)-\sum_{i=1}^{m}a_i(t)\cdot\lambda_i\right\}=c^{(q)}(t^*). \tag{3.11}$$

Якщо $c(t)\le 0$, то план-функція оптимальна.

2. Обчислити $a_j(t^*)$.
3. Вирішити систему:

$$\sum_{j=1}^{m}a_i\left(t_j^{(q-1)}\right)\cdot\theta_j=-a_i(t^*),\ \ i=1,2,\ldots,m, \tag{3.12}$$

за допомогою модулю np.linalg.

4. Обчислити $\frac{x_j}{\theta_j'}$.
5. Знайти

$$\max_{j\in\{j:\frac{x_j}{\theta_j}<0\}}\{\frac{x_j}{\theta_j}\}=\theta_0=\frac{x_{j_1}}{\theta_{j_1}'}. \tag{3.13}$$

6. Обчислити

$$\theta_0\cdot\theta_j'. \tag{3.14}$$

7. Визначити параметри нової план-функції $t_j,\ x_j$.
8. Обчислити

$$a_j\left(t_i^{(q)}\right), \quad j = 1, 2, \ldots, m, \quad i = 1, 2, \ldots, m; \qquad (3.15)$$
$$c(t_i), \quad i = 1, 2, \ldots, m.$$

9. Вирішити систему

$$\sum_{j=1}^{m} a_j\left(t_i^{(q)}\right) \cdot \tilde{\lambda}_i = c\left(t_i^{(q)}\right), \quad i = 1, 2, \ldots, m. \qquad (3.16)$$

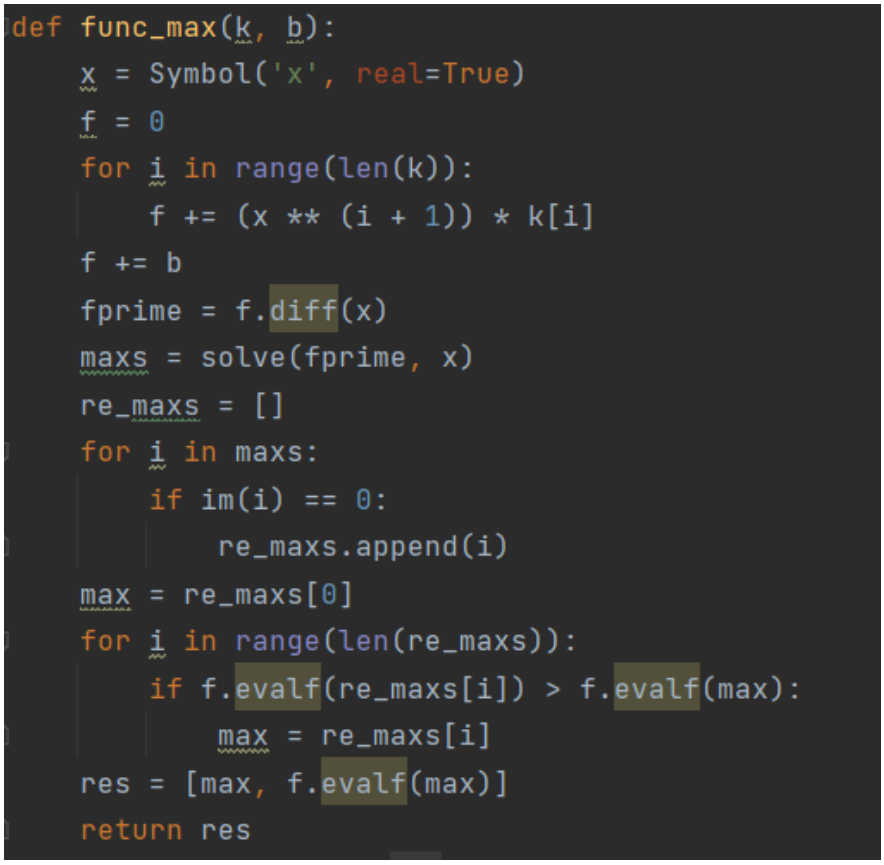
10. Обчислити значення цільової функції

$$Z = \sum_{j=1}^{m} x_j^{(q)} \cdot c\left(t_j^{(q)}\right). \qquad (3.17)$$

На рис. 2 продемонстровано реалізацію процедури знаходження максимуму цільової функції задачі, а також знаходження аргументу, на якому максимум досягається.

На рис. 3 наведено частину коду, в якому проводиться підготовчий етап для методу послідовного покращення план-функції.

На рис. 4, 5 наведено реалізацію алгоритму послідовного покращення план-функції.



```
def func_max(k, b):
    x = Symbol('x', real=True)
    f = 0
    for i in range(len(k)):
        f += (x ** (i + 1)) * k[i]
    f += b
    fprime = f.diff(x)
    maxs = solve(fprime, x)
    re_maxs = []
    for i in maxs:
        if im(i) == 0:
            re_maxs.append(i)
    max = re_maxs[0]
    for i in range(len(re_maxs)):
        if f.evalf(re_maxs[i]) > f.evalf(max):
            max = re_maxs[i]
    res = [max, f.evalf(max)]
    return res
```

Рисунок 2. Реалізація функції func_max: знаходить максимум функції та точку в якій функція його набуває

```
def start_plan(c_k, c_b, a_k, a_b, b, ti):
    t_size = len(a_k)
    tt = ti / t_size
    t = np.arange(0, ti, tt)
    A = []
    for i in range(len(b)):
        a = a_k[i] * t + a_b[i]
        A.append(a.tolist())
    x_0 = np.linalg.lstsq(A, b, rcond=None)[0]
    L = np.array(A).T
    C = c_k * t + c_b
    l_0 = np.linalg.lstsq(L, C, rcond=None)[0]
    C1 = np.where(t == 0, c_k, 0)
    res = [l_0.tolist(), C1.tolist(), L.tolist(), A, x_0.tolist()]
    return res
```

Рисунок 3. Реалізація функції start_plan: проводить підготовчий етап для методу послідовного покращення план-функції

```
def mppfe(L, l, C, c_k, c_b, a_k, a_b, x_0, t):
    C1 = c_k - l
    t_max, c_max = func_max(C1, c_b)

    if c_max < 0:
        return x_0

    a = a_k * t_max + a_b
    O = np.linalg.solve(L, -a)
    x_O = x_0 / O

    x_O = x_O[x_O >= 0]

    x_max = np.max(x_O)
    i_max = np.argmax(x_O)
    O_max = x_max
    t[i_max] = t_max

    x_01 = np.empty_like(x_0)
```

Рисунок 4. Реалізація функції mppfe(1 частина): реалізує алгоритм послідовного покращення план-функції

```
    for i in range(len(x_0)):
        if i != i_max:
            x_01[i] = x_0[i] - O[i] * O_max
        else:
            x_01[i] = -O_max

    A1 = []
    for i in range(len(C)):
        a = a_k[i] * t + a_b[i]
        A1.append(a.tolist())

    C2 = c_k * t + c_b

    l_res = np.linalg.lstsq(A1, C2, rcond=None)[0]

    Z = np.dot(x_01, C2)

    return [Z, x_01.tolist(), t.tolist(), l_res.tolist()]
```

Рисунок 5. Реалізація функції mppfe(1 частина): реалізує алгоритм послідовного покращення план-функції

Результат обчислень для числових даних (3.1)-(3.5), наведених на початку розділу:

```
Z = 1.253 Точність: 0.01 Кількість ітерацій: 5
Z = 1.25333 Точність: 0.0001 Кількість ітерацій: 12
```

Як випливає з наведених обчислень, результати методу дискретизації задачі континуального лінійного програмування не сильно відрізняються від результатів методу послідовного покращення план-функції, якщо обрати досить малий інтервал дискретизації. Але очевидно, що такий підхід не є оптимальним через більше споживання розрахункових потужностей.

ВИСНОВКИ

У ході роботи були опрацьовані та програмно реалізовані два різних за своїм підходом методи розв’язання задачі континуального лінійного програмування: дискретизації та метод послідовного покращення план-функції. Перший зводиться до інтегрування за інтервалами дискретизації та перетворення на звичайну задачу лінійного програмування, яку вже можна вирішувати будь-яким чисельним методом оптимізації. Другий, в свою чергу, враховує специфіку задач континуального лінійного програмування, і зводиться до ітераційної процедури, направленої на побудову та покращення початкової опорної план-функції.

У результаті реалізації обох алгоритмів на прикладі узагальненої задачі вибору моменту відкриття вогню, було отримано дані для порівняння ефективності вищезгаданих методів. Як можна, зокрема, бачити, через своє спрощення задачі алгоритм дискретизації дає хороші результати лише на великій частоті дискретизації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is Applied Mathematics? | About | Engineering Sciences [Електронний ресурс] – Режим доступу до ресурсу: https://www.mccormick.northwestern.edu/applied-math/about/what-is-applied-mathematics.html .
2. Mathematical Programming — AIMMS Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://documentation.aimms.com/platform/math-program/mathematical-programming.html.
3. Раскін Л.Г. Континуальне лінійне програмування / Л.Г. Раскін. – Харків, 2005.
4. NumPy Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org/doc/.
5. SymPy 1.12 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.sympy.org/latest/index.html.
6. SciPy documentation SciPy v1.10.1 Manual [Електронний ресурс] – Режим доступу до ресурсу: https://docs.scipy.org/doc/scipy/.
7. Levinson N.A. Class continuous linear programming problems / N. Levinson., 1966. – (Journal of Mathematical Analysis and Applications).