**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

**Факультет прикладної математики**

**Кафедра прикладної математики**

**ПРОЕКТ**

з дисципліни "Математичне моделювання"

**Тема:** Оптимізація ресурсів з метою підвищення прибутку підприємства

**Виконавець:**

Студентка групи КМ-73

Кисла В.О.

2020

Анотація

В основі даного проекту лежить програмне забезпечення , яке дозволяє промоделювати та оптимізувати наявні ресурси в залежності від різноманітності та запасів цих ресурсів, кількості виробів, що виготовляє підприємство, кількості ресурсів кожного виду на одиницю виробу та вартості виробів.

Перелік прийнятих скорочень

1. ПЗ – програмне забезпечення;
2. ОЗЛП – оптимальна задача лінійного програмування;
3. ЗЛП – задача лінійного програмування.

ЗМІСТ

[Вступ 5](#_Toc3329181)

[1 Постановка задачі 6](#_Toc3329182)

[2 Проектування математичного забезпечення 7](#_Toc3329183)

[2.1 Огляд літератури 7](#_Toc3329184)

[2.2 Математичний метод розв’язання задачі 7](#_Toc3329185)

[2.3 Архітектура розроблених програмних засобів 9](#_Toc3329186)

[2.4 Опис користувацького інтерфейсу 10](#_Toc3329187)

[3 Програмна реалізація 12](#_Toc3329188)

[3.1 Опис розроблених алгоритмів 12](#_Toc3329189)

[3.2 Опис розроблених програм 13](#_Toc3329190)

[3.3 Мова управління програмою 14](#_Toc3329191)

[3.4 Керівництво користувача 14](#_Toc3329192)

[4 Випробування програмних засобів 15](#_Toc3329193)

[4.1 Опис контрольних прикладів 15](#_Toc3329194)

[4.2 Опис експериментальних розрахунків 17](#_Toc3329195)

[4.3. Рекомендації щодо використання 20](#_Toc3329196)

[Перелік використаних джерел 22](#_Toc3329197)

[Додаток А Лістинг програм 23](#_Toc3329198)

Вступ

Задача оптимізації є досить актуальна на виробництві, де потрібно використати наявні ресурси з максимальною вигодою для компанії, заводу тощо. Неправильний розподіл ресурсів може призвести до зменшення прибутку, а можливо навіть і до збитків. На підприємстві, якщо воно виробляє багато різних товарів і використовує для цього багато різних ресурсів, необхідно точно знати як оптимально використати наявні ресурси, щоб отримати найбільший прибуток для свого підприємства.

Автоматизація процесу значно спростить та пришвидшить процес оптимізації ресурсів. Також зменшиться ймовірність помилку обчислень через людський фактор. Це усе допоможе підприємству отримати найбільший прибуток.

1 Постановка задачі

Розробити ПЗ, за допомогою якого можна виходячи з особливостей технологічних процесів підприємства та наявних виробничих ресурсів, знайти таку виробничу програму, яка б забезпечувала отримання максимального прибутку від реалізації виготовленої продукції.

2 Проектування математичного забезпечення

2.1 Огляд літератури

Для рішення задачі розподілу ресурсів використовують методи, наведені в оглянутій літературі [1 – 4] та представлені нижче:

1. Симплекс-метод[1 – 6]
2. Геометричний симплекс метод[1 – 5]
3. Симплекс-метод зі штучним базисом[2 – 4]
4. Модифікований симплекс-метод[6]

Для реалізації даного проекту обрано симплекс-метод зі штучним базисом.

Перевага симплекс-методу зі штучним базисом полягає в тому, що цей метод дозволяє розв’язувати задачі будь-якої розмірності. Цей метод застосовується навіть тоді, коли дуже важко знайти початковий опорний план, коли звичайний симплекс-метод не розв’яже задачу. Симплекс-метод зі штучним базисом охоплює значно більше задач ніж інші методи.[6]

2.2 Математичний метод розв’язання задачі

ОЗЛП:



i=1,…,m

xj≥0 j=1,…,n.

Область допустимих розв’язків ЗЛП представляє собою випуклий многогранник. Оптимальний розв’язок знаходиться в вершинах многогранника. В основі симплекс-метода[1-6] лежить ціле направлений перебір вершин області допустимих розв’язків. При цьому рух до оптимальної точки здійснюється по сусіднім вершинам області.

**Канонічна форма ЗЛП**

КЗЛП:

Мінімізувати

i=1,…,m

xj≥0 j=1,…,n.

max(z)=-min(-z);

Перетворення нерівностей в рівності здійснюється шляхом введення додаткових невід’ємних змінних:

→ , xn+i≥0;

→ , xn+i≥0;

Всі вільні члени bi≥0.

**Пошук оптимального значення СМ**

Приведення ОЗЛП до КЗЛП.

Вибір початкового допустимого базисного розв’язку .

Перехід від початкового базисного розв’язку до суміжного допустимого базисного розв’язку з кращим значенням цільової функції.

Продовження пошуку допустимих базисних розв’язків, що покращують значення цільової функції. Якщо існуючий допустимий базисний розв’язок покращити не можна, він є оптимальним.

2.3 Архітектура розроблених програмних засобів

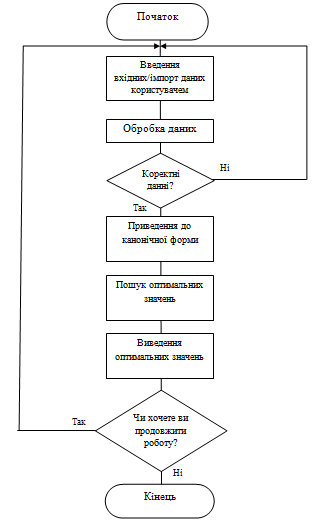


Рисунок 2.3.1. Схема взаємодії програмних засобів

2.4 Опис користувацького інтерфейсу

На рисунках 2.4.1 – 2.4.3 зображений користувацький інтерфейс розробленого програмного забезпечення.

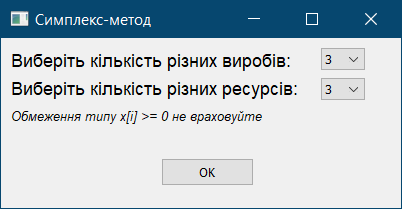


Рис. 2.4.1. Задання розмірності вхідної задачі

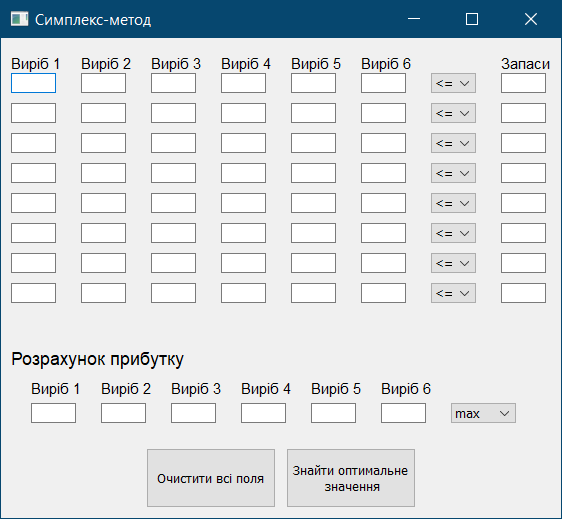


Рис. 2.4.2. Введення даних

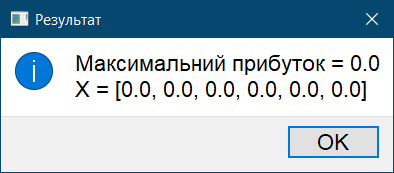


Рис. 2.4.3. Отриманий результат

3 Програмна реалізація

3.1 Опис розроблених алгоритмів

ні

так

ні

Кінець

Вибір генерального елемента

Перетворення симплексної таблиці методом Гаусса

Усі

Початок

Введення вхідних даних: n, m, aij, bi, cj

Вибір генерального стовпчика

Усі

Вивід: оптимального плану та оптимума

Вивід: цільова функція необмежена

так

Рисунок 3.1.1. Блок-схема алгоритму розроблюваної програми

На рисунку 3.1.1 зображено алгоритм роботи програми. Програма працює у відповідності методу, описаного в підпункті 2.2 розділу 2. Значення функцій та змінних зберігаються у вигляді масивів між якими існує відповідність за індексами.

* 1. Опис розроблених програм

ПЗ написане мовою програмування Python, в якій представлені програмні засоби, що дозволяють легко запрограмовувати методів оптимізації та представити зручний інтерфейс для користувача. Розроблене ПЗ складається з 1 модулю:

1. interface.py – графічний інтерфейс користувача, введення даних, виведення результатів роботи;
2. simplex.py – реалізація методу, вибір методів в залежності від задачі.

Програмні коди відповідного файлу наведені в додатку А.

3.3 Мова управління програмою

Користувач може ввести:

* кількість обмежень ;
* кількість ресурсів, що оптимізуються;
* коефіцієнти системи рівнянь, що записані у вигляді матриці;
* коефіцієнти цільової функції, що записані у вигляді вектору.

Після введення вищезазначених даних, користувач натискає кнопку «оптимальний розв’язок» для отримання результатів. Усі введені дані підлягають перевірці на відповідність бізнес-логіці та відсутність синтаксичних помилок.

3.4 Керівництво користувача

Спочатку користувачеві потрібно обрати з випадаючого меню розмірність задачі, а саме кількість різних виробів та кількість різних наявних ресурсів. Потім він побачить нове вікно, де матиме змогу ввести свої дані у відповідні комірки. Після натискання кнопки «Оптимальний розв’язок» користувач побачить перед собой вікно з отриманим результатом. Також натиснувши «Очистити» можна очистити усі поля. Закрити вікно можна натиснувши хрестик в правому верхньому куті.

4 Випробування програмних засобів

4.1 Опис контрольних прикладів

Таблиця 4.1.1. Приклад 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ресурси | Норми витратів ресурсів на одиницю виробу | | | | Наявність ресурсів |
| Килим №1 | Килим №2 | Килим №3 | Килим №4 |
| Праця (люд/днів) | 7 | 2 | 2 | 6 | 80 |
| Сировина (кг) | 5 | 8 | 4 | 3 | 480 |
| Обладнання (станко/год) | 2 | 4 | 1 | 8 | 130 |
| Ціна (тис. грн.) | 3 | 4 | 3 | 1 |  |

Обмеження:

Таблиця 4.1.2. Приклад 2

|  |  |  |  |
| --- | --- | --- | --- |
| Вид деревини | Вид виробу | | Запаси |
| Стіл | Шафа |
| Д1 (m3) | 0,15 | 0,2 | 60 |
| Д2 (m3) | 0,2 | 0,1 | 40 |
| Дохід (у. е.) | 12 | 15 |  |

Обмеження:

Таблиця 4.1.3. Приклад 3

|  |  |  |  |
| --- | --- | --- | --- |
| Сировина | Продукція | | Наявність ресурсів |
| Виріб 1 | Виріб 2 |
| Ресурс 1 | 2,4 | 12,2 | 500 |
| Ресурс 2 | 8 | 5,4 | 470 |
| Ресурс 3 | 6,2 | 2,2 | 340 |
| Дохід (у. е.) | 50 | 40 |  |

Обмеження:

4.2 Опис експериментальних розрахунків

Результати виконання контрольних прикладів 1 – 3 з п. 1 представлені на рисунках 2.1 – 2.6. На даних рисунках представлені результати роботи програмного забезпечення та скріншоти результатів.

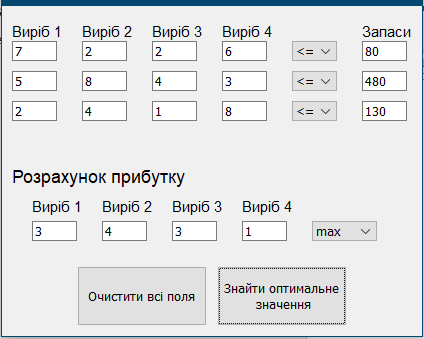


Рисунок 2.1. Введення даних прикладу 1.

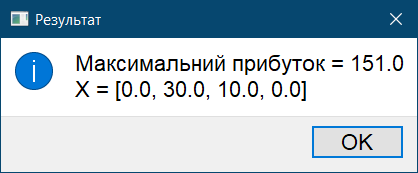


Рисунок 2.2. Результати виконання прикладу 1.

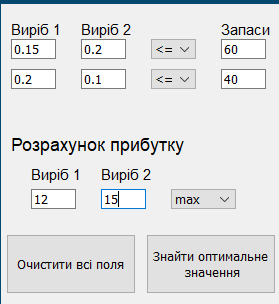


Рисунок 2.3. Введення даних прикладу 2.

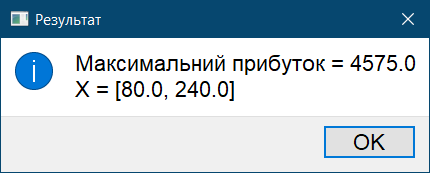


Рисунок 2.4. Результати виконання прикладу 2.

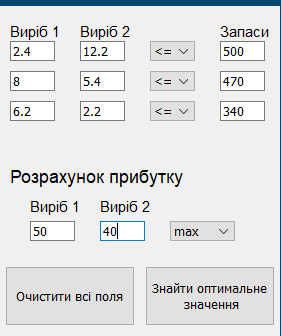


Рисунок 2.5. Введення даних прикладу 3.

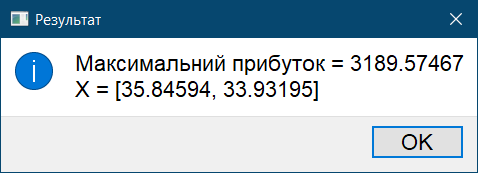


Рисунок 2.6. Результати виконання прикладу 3.

Під час випробування розробленого програмного забезпечення шляхом виконання вищенаведених контрольних прикладів, було встановлено, що результат, тобто виготовлення тієї чи іншої кількості певного продукту для отримання прибутку залежить від двох чинників:

1. Кількість виробу прямопропорційно залежить від тієї суми грошей, яку можна отримати від реалізації одиниці цього виробу. Це гарно демонструє другий приклад. На ньому видно, що для максимального прибутку потрібно виробити «виріб 1» всього 80 одиниць, тому що він найдешевший. Натомість «виріб 2», який є найдорожчим, потрібно виготовити 240 одиниць, що є втричі більше за «виріб 1».
2. Кількість виробу також залежить від кількості затрачених на нього ресурсів. Тобто чим більше витрачаємо ресурсів на якийсь виріб, тим меншу кількість потрібно виготовити, щоб заощадити ресурси для інших виробів. Як видно з прикладу 1, маємо два вироби, вартість яких однакова. Проте «виріб 1» виробляти взагалі не доцільно, тому що нам потрібно використати більше ресурсів на нього і отримаємо менший прибуток, тому краще виготовляти «виріб 3».

4.3. Рекомендації щодо використання

Дане ПЗ розроблене для використання на персональних комп’ютерах користувачів, не має аналогової версії для мобільних або консольних пристроїв, а також не представлене у вигляді web-застосунку.

Також алгоритм даної програми передбачає наявність логічного взаємозв’язку між вхідними даними, запрограмований метод доцільно використовувати в реальних умовах виробничого процесу без перевірки шляхом введення фіктивних даних.

Висновки

В ході виконання проекту було розроблено ПЗ, яке дозволяє оптимізувати наявні ресурси, а саме розподілити їх для виробництва певних виробів таким чином, щоб отримати найбільший прибуток після реалізації цих виробів. Дане ПЗ є основою для моделювання фінансових стратегій на підприємствах. В програмному забезпеченні використовуються методи математичної статистики та моделювання.

Під час випробування розробленого програмного забезпечення шляхом виконання вищенаведених контрольних прикладів, було встановлено, що результат, тобто виготовлення тієї чи іншої кількості певного продукту для отримання прибутку залежить від двох чинників:

1. Кількість виробу прямопропорційно залежить від тієї суми грошей, яку можна отримати від реалізації одиниці цього виробу;
2. Кількість виробу також залежить від кількості затрачених на нього ресурсів. Тобто чим більше витрачаємо ресурсів на якийсь виріб, тим меншу кількість потрібно виготовити, щоб заощадити ресурси для інших виробів;

Перелік використаних джерел

1. Васильев, Ф. П. Линейное программирование / Ф.П. Васильев, А.Ю. Иваницкий. - М.: Факториал Пресс, 2016. - 352 c.
2. Линейное и нелинейное программирование / И.Н. Ляшенко и др. - М.: Вища школа, 2016. - 372 c.
3. Палий, И.А. Линейное программирование / И.А. Палий. - М.: Эксмо, 2014**.** - 256 c.
4. Юдин, Д. Б. Задачи и методы линейного программирования. Задачи транспортного типа / Д.Б. Юдин, Е.Г. Гольштейн. - М.: Либроком, 2013. - 184 c.
5. Юдин, Д. Б. Задачи и методы линейного программирования. Конечные методы / Д.Б. Юдин, Е.Г. Гольштейн. - М.: Либроком, 2016. - 264 c.
6. Юдин, Д. Б. Задачи и методы линейного программирования. Математические основы и практические задачи / Д.Б. Юдин, Е.Г. Гольштейн. - М.: Либроком, **2016**. - 322 c.

Додаток А  
Лістинг програм

Лістинг файлу interface.py:

import sys

from PyQt5.QtWidgets import QWidget

from PyQt5.QtWidgets import QLabel

from PyQt5.QtWidgets import QComboBox

from PyQt5.QtWidgets import QMainWindow

from PyQt5.QtWidgets import QApplication

from PyQt5.QtWidgets import QPushButton

from PyQt5.QtWidgets import QLineEdit

from PyQt5.QtWidgets import QMessageBox

from PyQt5.QtGui import QIcon, QFont

from PyQt5 import QtCore, QtGui

from simplex import Simplex, simplex\_method\_fictitious\_basis

import numpy as np

class MainWindow(QWidget):

def \_\_init\_\_(self, parent=None):

super().\_\_init\_\_(parent)

self.number\_of\_variables = "3"

self.number\_of\_limitations = "3"

self.secondWin = None

self.initUI()

def initUI(self):

self.lbl\_1 = QLabel(self)

self.lbl\_1.move(10, 12)

self.lbl\_1.setFont(QFont("Arial", 11))

self.lbl\_1.setText("Виберіть кількість різних виробів: ")

self.combo\_1 = QComboBox(self)

for i in range(11):

self.combo\_1.addItem(str(i+2), i)

self.combo\_1.move(320, 10)

self.combo\_1.setCurrentIndex(1)

self.combo\_1.activated.connect(self.set\_number\_of\_variables)

self.lbl\_2 = QLabel(self)

self.lbl\_2.move(10, 40)

self.lbl\_2.setFont(QFont("Arial", 11))

self.lbl\_2.setText("Виберіть кількість різних ресурсів: ")

self.text\_label = QLabel(self)

self.text\_label.move(10, 70)

self.text\_label.setText("Обмеження типу x[i] >= 0 не враховуйте")

self.text\_label.setFont(QFont("Arial", 8, italic=True))

self.combo\_2 = QComboBox(self)

for i in range(12):

self.combo\_2.addItem(str(i+1), i)

self.combo\_2.move(320, 40)

self.combo\_2.setCurrentIndex(2)

self.combo\_2.activated.connect(self.set\_number\_of\_limitations)

self.okbutton = QPushButton("OK", self)

self.okbutton.move(160, 120)

self.okbutton.clicked.connect(self.openWin)

self.setGeometry(150, 150, 400, 170)

self.setWindowTitle('Симплекс-метод')

self.setWindowIcon(QIcon('icon.jpg'))

self.show()

def set\_number\_of\_variables(self, index):

self.number\_of\_variables = self.combo\_1.itemText(index)

def set\_number\_of\_limitations(self, index):

self.number\_of\_limitations = self.combo\_2.itemText(index)

def openWin(self):

if self.secondWin is not None:

self.secondWin = None

self.secondWin = SecondWindow(self.number\_of\_variables,

self.number\_of\_limitations)

self.secondWin.show()

class SecondWindow(QMainWindow):

def \_\_init\_\_(self, number\_of\_variables,

number\_of\_limitations, parent=None):

super().\_\_init\_\_(parent, QtCore.Qt.Window)

self.matrix\_entry = []

self.x\_str = []

self.sign = []

self.b = []

self.z = []

self.z\_str = []

self.number\_of\_variables = int(number\_of\_variables)

self.number\_of\_limitations = int(number\_of\_limitations)

self.initUI()

def initUI(self):

reg = QtCore.QRegExp("^[-]?[0-9]{1,8}(\.[0-9]{1,5})?")

validator = QtGui.QRegExpValidator(reg)

for i in range(self.number\_of\_variables):

self.x\_str.append(QLabel("Виріб "+str(i+1), self))

self.x\_str[i].move(10+70\*i, 10)

self.x\_str[i].setFont(QFont("Arial", 9))

self.b\_lbl = QLabel("Запаси", self)

self.b\_lbl.move(10+70\*(i+2), 10)

self.b\_lbl.setFont(QFont("Arial", 9))

for i in range(self.number\_of\_variables):

self.matrix\_entry.append([])

for j in range(self.number\_of\_limitations):

self.matrix\_entry[i].append(QLineEdit(self))

self.matrix\_entry[i][j].resize(45, 20)

self.matrix\_entry[i][j].move(10+70\*i, 35+30\*j)

self.matrix\_entry[i][j].setValidator(validator)

for j in range(self.number\_of\_limitations):

self.sign.append(QComboBox(self))

item = ["<=", "=", ">="]

for itr in range(3):

self.sign[j].addItem(item[itr], itr)

self.sign[j].move(10+70\*(i+1), 35+30\*j)

self.sign[j].resize(45, 20)

for j in range(self.number\_of\_limitations):

self.b.append(QLineEdit(self))

self.b[j].resize(45, 20)

self.b[j].move(10+70\*(i+2), 35+30\*j)

self.b[j].setValidator(validator)

self.f\_lbl = QLabel("Розрахунок прибутку", self)

self.f\_lbl.resize(200, 30)

self.f\_lbl.move(10, 35+30\*(j+2))

self.f\_lbl.setFont(QFont("Arial", 11))

for i in range(self.number\_of\_variables):

self.z\_str.append(QLabel("Виріб "+str(i+1), self))

self.z\_str[i].move(30+70\*i, 35+30\*(j+3))

self.z\_str[i].setFont(QFont("Arial", 9))

# self.z\_str.append(QLabel("C", self))

# self.z\_str[-1].move(30+70\*(i+1), 35+30\*(j+3))

# self.z\_str[-1].setFont(QFont("Arial", 9))

# self.z\_str.append(QLabel("extr", self))

# self.z\_str[-1].move(30+70\*(i+2), 35+30\*(j+3))

# self.z\_str[-1].setFont(QFont("Arial", 9))

for i in range(self.number\_of\_variables):

self.z.append(QLineEdit(self))

self.z[i].resize(45, 20)

self.z[i].move(30+70\*i, 35+30\*(j+4))

self.z[i].setValidator(validator)

self.sign.append(QComboBox(self))

self.sign[-1].addItem("max", 0)

self.sign[-1].addItem("min", 1)

self.sign[-1].move(30+70\*(i+1), 35+30\*(j+4))

self.sign[-1].resize(65, 20)

but = QPushButton("Знайти оптимальне\n значення", self)

but.resize(130, 60)

but.move((80+70\*(self.number\_of\_variables+1))//2, 50+30\*(j+5))

but.clicked.connect(self.solve)

clear\_line\_button = QPushButton("Очистити всі поля", self)

clear\_line\_button.resize(130, 60)

clear\_line\_button.move((70\*(self.number\_of\_variables+1)-200)//2,

50+30\*(j+5))

clear\_line\_button.clicked.connect(self.clear\_line)

self.setGeometry(150, 150, 70+70\*(self.number\_of\_variables+1),

90+30\*(j+6))

self.setWindowTitle('Симплекс-метод')

self.setWindowIcon(QIcon('icon.jpg'))

def clear\_line(self):

for i in range(self.number\_of\_variables):

for j in range(self.number\_of\_limitations):

self.matrix\_entry[i][j].clear()

for i in range(self.number\_of\_limitations):

self.b[i].clear()

for i in range(self.number\_of\_variables+1):

self.z[i].clear()

def isEmpty(self):

empty = False

for i in np.reshape(self.matrix\_entry,

self.number\_of\_variables\*

self.number\_of\_limitations):

if i.text() == "":

empty = True

break

if empty:

return empty

for i in self.b:

if i.text() == "":

empty = True

break

if empty:

return empty

for i in self.z:

if i.text() == "":

empty = True

break

return empty

def solve(self):

isMax = False

isEqual = False

if self.isEmpty():

QMessageBox.warning(self, "Message",

"Всі поля мають бути заповнені")

else:

signs = []

for i in range(self.number\_of\_limitations+1):

signs.append(self.sign[i].currentText())

X = np.zeros((self.number\_of\_limitations+1,

self.number\_of\_variables+1))

for i in range(self.number\_of\_limitations):

X[i][0] = float(self.b[i].text())

X[i+1][0] = float(self.z[-1].text())

for i in range(self.number\_of\_limitations):

for j in range(self.number\_of\_variables):

X[i][j+1] = float(self.matrix\_entry[j][i].text())

for i in range(self.number\_of\_variables):

X[-1][i+1] = float(self.z[i].text())

for i in range(self.number\_of\_limitations):

if signs[i] == ">=":

X[i] = (-1)\*X[i]

if signs[-1] == "min":

X[-1][1:] = (-1)\*X[-1][1:]

else:

X[-1][0] = (-1)\*X[-1][0]

isMax = True

initial\_shape = X.shape[1]

if "=" in signs:

isEqual = True

rows\_with\_fict\_vars = []

X = np.insert(X, X.shape[0], np.zeros(X.shape[1]), axis=0)

for i in range(self.number\_of\_limitations):

if X[i][0] < 0 and signs[i] != "=":

X = np.concatenate((X, np.zeros((X.shape[0], 1))), axis=1)

X[i] = (-1)\*X[i]

X[i][-1] = -1.

X[-1] += X[i]

rows\_with\_fict\_vars.append(i)

elif X[i][0] < 0 and signs[i] == "=":

X[i] = (-1)\*X[i]

X[-1] += X[i]

rows\_with\_fict\_vars.append(i)

elif X[i][0] >= 0 and signs[i] == "=":

X[-1] += X[i]

rows\_with\_fict\_vars.append(i)

rows\_with\_fict\_vars = np.array(rows\_with\_fict\_vars)

if isMax:

if not isEqual:

res, vector = Simplex(X).use\_appropriate\_simplex\_method()

if vector is not None:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Information)

msg.setText("Maксимальний прибуток = {}\nX = {}".format(-res, vector))

msg.setWindowTitle("Результат")

msg.setFont(QFont("Arial", 13))

msg.exec\_()

else:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Information)

msg.setText(res.replace("мінімум", "максимум"))

msg.setWindowTitle("Результат")

msg.setFont(QFont("Arial", 13))

msg.exec\_()

else:

res, vector = simplex\_method\_fictitious\_basis(

X, rows\_with\_fict\_vars, initial\_shape

)

if vector is not None:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Information)

msg.setText("Maксимальний прибуток {}\nX = {}".format(-res, vector))

msg.setWindowTitle("Результат")

msg.setFont(QFont("Arial", 13))

msg.exec\_()

else:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Information)

msg.setText(res.replace("мінімум", "максимум"))

msg.setWindowTitle("Результат")

msg.setFont(QFont("Arial", 13))

msg.exec\_()

else:

if not isEqual:

res, vector = Simplex(X).use\_appropriate\_simplex\_method()

if vector is not None:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Information)

msg.setText("Min(Z) = {}\nX = {}".format(res, vector))

msg.setWindowTitle("Результат")

msg.setFont(QFont("Arial", 13))

msg.exec\_()

else:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Information)

msg.setText(res)

msg.setWindowTitle("Результат")

msg.setFont(QFont("Arial", 13))

msg.exec\_()

else:

res, vector = simplex\_method\_fictitious\_basis(

X, rows\_with\_fict\_vars, initial\_shape

)

if vector is not None:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Information)

msg.setText("Min(Z) = {}\nX = {}".format(res, vector))

msg.setWindowTitle("Результат")

msg.setFont(QFont("Arial", 13))

msg.exec\_()

else:

msg = QMessageBox(self)

msg.setIcon(QMessageBox.Information)

msg.setText(res)

msg.setWindowTitle("Результат")

msg.setFont(QFont("Arial", 13))

msg.exec\_()

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

w = MainWindow()

sys.exit(app.exec\_())

Лістинг файлу simplex.py:

import numpy as np

class Simplex:

def \_\_init\_\_(self, X):

self.unable\_g\_row = False

self.isAdmissible = all((np.round(X.T[0][:-1], 5)) >= 0)

self.isOptimal = all((np.round(X[-1][1:], 5)) <= 0)

self.X = X

self.free = {}

self.basis = {}

for i in range(self.X.shape[1]-1):

self.free['x'+str(i+1)] = 0.

for i in range(self.X.shape[1], self.X.shape[1]+self.X.shape[0]-1):

self.basis['x'+str(i)] = self.X[i-self.X.shape[1]][0]

def result(self):

if self.isAdmissible and self.isOptimal:

return (round(self.X[-1][0], 5),

[round(dict(self.free.items() | \

self.basis.items())['x'+str(i)], 5) \

for i in range(1, self.X.shape[1])])

if self.isAdmissible and self.unable\_g\_row:

return "Необмежений мінімум", None

if not self.isAdmissible and self.unable\_g\_row:

return "Система обмежень несумісна", None

def use\_appropriate\_simplex\_method(self):

if self.isAdmissible:

self.simplex\_method\_for\_admissible\_solution()

return self.result()

else:

self.simplex\_method\_for\_inadmissible\_solution()

return self.result()

def swap(self, g\_row, g\_col):

free\_keys = []

for i, j in zip(self.free.keys(), range(1, self.X.shape[1])):

if j != g\_col:

free\_keys.append(i)

else:

toDelete\_free = i

self.free.pop(toDelete\_free)

basis\_keys = []

for i, j in zip(self.basis.keys(), range(self.X.shape[0]-1)):

if j != g\_row:

basis\_keys.append(i)

else:

toDelete\_basis = i

self.basis.pop(toDelete\_basis)

free\_keys = np.insert(free\_keys, g\_col-1, toDelete\_basis)

basis\_keys = np.insert(basis\_keys, g\_row, toDelete\_free)

self.free.clear()

self.basis.clear()

for i in free\_keys:

self.free[i] = 0.

for i, j in zip(basis\_keys, range(self.X.shape[0]-1)):

self.basis[i] = self.X[j][0]

def simplex\_method\_for\_admissible\_solution(self):

self.isOptimal = all((np.round(self.X[-1][1:], 5)) <= 0)

while self.isOptimal == False:

g\_col = np.argmax(self.X[-1][1:]) + 1

self.unable\_g\_row = all((np.round(self.X.T[g\_col][:-1], 5)) <= 0)

if self.unable\_g\_row:

break

g\_row = np.argmin(np.array(

[self.X[i][0]/self.X[i][g\_col] \

if self.X[i][g\_col] > 0 else np.inf \

for i in range(self.X.shape[0]-1)]

))

X1 = np.zeros((self.X.shape[0], self.X.shape[1]))

X1[g\_row][g\_col] = 1 / self.X[g\_row][g\_col]

for j in range(self.X.shape[1]):

if j == g\_col:

continue

X1[g\_row][j] = self.X[g\_row][j] / self.X[g\_row][g\_col]

for i in range(self.X.shape[0]):

if i == g\_row:

continue

X1[i][g\_col] = (-1) \* self.X[i][g\_col] / self.X[g\_row][g\_col]

for i in range(self.X.shape[0]):

if i == g\_row:

continue

for j in range(self.X.shape[1]):

if j == g\_col:

continue

X1[i][j] = self.X[i][j] + X1[i][g\_col] \* self.X[g\_row][j]

self.X = X1

self.swap(g\_row, g\_col)

self.isOptimal = all((np.round(self.X[-1][1:], 5)) <= 0)

def simplex\_method\_for\_inadmissible\_solution(self):

while self.isAdmissible == False:

g\_row = None

for i in range(self.X.shape[0]-1):

if self.X[i][0] < 0:

for j in range(1, self.X.shape[1]):

if self.X[i][j] < 0:

g\_row = i

g\_col = j

break

break

if g\_row is None:

self.unable\_g\_row = True

break

else:

X1 = np.zeros((self.X.shape[0], self.X.shape[1]))

X1[g\_row][g\_col] = 1 / self.X[g\_row][g\_col]

for j in range(self.X.shape[1]):

if j == g\_col:

continue

X1[g\_row][j] = self.X[g\_row][j] / self.X[g\_row][g\_col]

for i in range(self.X.shape[0]):

if i == g\_row:

continue

X1[i][g\_col] = (-1)\*self.X[i][g\_col]/self.X[g\_row][g\_col]

for i in range(self.X.shape[0]):

if i == g\_row:

continue

for j in range(self.X.shape[1]):

if j == g\_col:

continue

X1[i][j] = self.X[i][j]+X1[i][g\_col]\*self.X[g\_row][j]

self.X = X1

self.swap(g\_row, g\_col)

self.isAdmissible = all((np.round(self.X.T[0][:-1], 5)) >= 0)

if not self.unable\_g\_row:

self.simplex\_method\_for\_admissible\_solution()

def simplex\_method\_fictitious\_basis(X, ficts, initial\_shape):

free = {}

basis = {}

for i in range(X.shape[1]-1):

free['x'+str(i+1)] = 0.

for i in range(X.shape[1], X.shape[1]+X.shape[0]-2):

basis['x'+str(i)] = X[i-X.shape[1]][0]

unable\_g\_col = False

unable\_g\_row = False

while round(X[-1][0], 5) != 0:

if all(np.round(X[-1][1:], 5) <= 0):

unable\_g\_col = True

break

X1 = np.zeros((X.shape[0], X.shape[1]))

g\_col = np.argmax(X[-1][1:])+1

g\_row = np.argmin(np.array([X[i][0]/X[i][g\_col] \

if X[i][g\_col] > 0 else np.inf \

for i in range(X.shape[0]-2)]))

for j in range(X.shape[1]):

if j == g\_col:

continue

X1[g\_row][j] = X[g\_row][j] / X[g\_row][g\_col]

for i in range(X.shape[0]):

if i == g\_row:

continue

X1[i][g\_col] = (-1) \* X[i][g\_col] / X[g\_row][g\_col]

for i in range(X.shape[0]):

if i == g\_row:

continue

for j in range(X.shape[1]):

if j == g\_col:

continue

X1[i][j] = X[i][j] + X1[i][g\_col] \* X[g\_row][j]

X = X1

free\_keys = []

for i, j in zip(free.keys(), range(1, X.shape[1])):

if j != g\_col:

free\_keys.append(i)

else:

toDelete\_free = i

free.pop(toDelete\_free)

basis\_keys = []

for i, j in zip(basis.keys(), range(X.shape[0]-2)):

if j != g\_row:

basis\_keys.append(i)

else:

toDelete\_basis = i

basis.pop(toDelete\_basis)

if g\_row not in ficts:

free\_keys = np.insert(free\_keys, g\_col-1, toDelete\_basis)

basis\_keys = np.insert(basis\_keys, g\_row, toDelete\_free)

free.clear()

basis.clear()

for i in free\_keys:

free[i] = 0.

for i, j in zip(basis\_keys, range(X.shape[0]-1)):

basis[i] = X[j][0]

if g\_row in ficts:

X = np.delete(X, g\_col, 1)

ficts = np.delete(ficts, np.where(ficts == g\_row)[0], 0)

if unable\_g\_col:

return "Система обмежень несумісна", None

X = np.delete(X, -1, 0)

isOptimal = all((np.round(X[-1][1:], 5)) <= 0)

while isOptimal == False:

g\_col = np.argmax(X[-1][1:]) + 1

unable\_g\_row = all((np.round(X.T[g\_col][:-1], 5)) <= 0)

if unable\_g\_row:

break

g\_row = np.argmin(np.array(

[X[i][0]/X[i][g\_col] \

if X[i][g\_col] > 0 else np.inf \

for i in range(X.shape[0]-1)]

))

X1 = np.zeros((X.shape[0], X.shape[1]))

X1[g\_row][g\_col] = 1 / X[g\_row][g\_col]

for j in range(X.shape[1]):

if j == g\_col:

continue

X1[g\_row][j] = X[g\_row][j] / X[g\_row][g\_col]

for i in range(X.shape[0]):

if i == g\_row:

continue

X1[i][g\_col] = (-1) \* X[i][g\_col] / X[g\_row][g\_col]

for i in range(X.shape[0]):

if i == g\_row:

continue

for j in range(X.shape[1]):

if j == g\_col:

continue

X1[i][j] = X[i][j] + X1[i][g\_col] \* X[g\_row][j]

X = X1

free\_keys = []

for i, j in zip(free.keys(), range(1, X.shape[1])):

if j != g\_col:

free\_keys.append(i)

else:

toDelete\_free = i

free.pop(toDelete\_free)

basis\_keys = []

for i, j in zip(basis.keys(), range(X.shape[0]-1)):

if j != g\_row:

basis\_keys.append(i)

else:

toDelete\_basis = i

basis.pop(toDelete\_basis)

free\_keys = np.insert(free\_keys, g\_col-1, toDelete\_basis)

basis\_keys = np.insert(basis\_keys, g\_row, toDelete\_free)

free.clear()

basis.clear()

for i in free\_keys:

free[i] = 0.

for i, j in zip(basis\_keys, range(X.shape[0]-1)):

basis[i] = X[j][0]

isOptimal = all((np.round(X[-1][1:], 5)) <= 0)

if unable\_g\_row:

return "Необмежений мінімум", None

d = dict(free.items() | basis.items())

return round(X[-1][0], 5), [round(d['x'+str(i)], 5) \

for i in range(1, initial\_shape)]