МIНIСТЕРСТВО ОСВIТИ I НАУКИ УКРАЇНИ

НАЦIОНАЛЬНИЙ ТЕХНIЧНИЙ УНIВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛIТЕХНIЧНИЙ IНСТИТУТ»

Кафедра прикладної математики

Звіт

із лабораторної роботи №5

з дисципліни «Алгоритми і системи комп’ютерної математики.

Математичні алгоритми»

на тему:

«Лінійне програмування (симплекс-метод)»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-63 | *Старший викладач Бай Ю.П.* |
| *Артеменко Я.К.* |  |

Київ — 2019

# **ЗМІСТ**

[**ВСТУП** 3](#_Toc27145806)

[**2 ОСНОВНА ЧАСТИНА** 4](#_Toc27145807)

[**2.1 Завдання** 4](#_Toc27145808)

[**2.2 Описання методу** 4](#_Toc27145809)

[**2.3 Використані тест кейси** 5](#_Toc27145810)

[**3 ВИСНОВКИ** 7](#_Toc27145811)

[**ПЕРЕЛІК ПОСИЛАНЬ** 8](#_Toc27145812)

[**ДОДАТКИ** 9](#_Toc27145813)

[**Додаток А (код програми для алгоритму)** 9](#_Toc27145814)

[**Додаток Б (код програми для тестів)** 11](#_Toc27145815)

[**Додаток В (результати виконання програми)** 12](#_Toc27145816)

[**Додаток Г (підтвердження правильності роботи алгоритму)** 12](#_Toc27145817)

# **ВСТУП**

Мета лабораторної роботи:

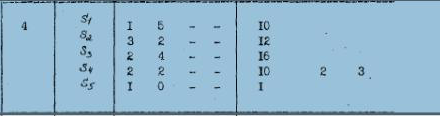
Розробити програмне забезпечення для лінійного програмування, а саме для задачі про суміші та провести аналіз задачі, що розв’язується, та методу її розв’язання на предмет виключних ситуацій, які можуть виникати під час застосування заданого методу до розв’язання поставленої задачі.

Завдання складання оптимальної суміші є однією з найбільш поширених технологічних задач оптимізації. Вона застосовується в різних областях: нафтопереробці, виробництві мінеральних добрив і кормів для тварин, складанні оптимальних раціонів і дієт. Тому дана тема є актуальною.

# **2 ОСНОВНА ЧАСТИНА**

# **2.1 Завдання**

Розв’язати задачу про суміші для поставленого варіанту №4:



**Математична постановка даної задачі має наступний вигляд:**

# **2.2 Описання методу**

Постановка задачі:

N інгредієнтів - y1, y2, y3, y4. В результаті змішування цих інгредієнтів в пропорціях g11: g12: g13: g14, g21: g22: g23: g24, g31: g32: g33: g34 і g41: g42: g43: g44 отримують суміш n сортів x1, x2, x3, x4. Ціна його реалізації відповідно s1, s2, s3, s4.

Економіко-математична модель задачі:



Σg1i = g11 + g12 + g13 + g14

Цільова функція:  
F(x) = s1x1 + s2x2 + s3x3 + s4x4 → max

Обмеження:  
x1g11/Σg1i + x2g21/Σg2i + x3g31/Σg3i + x4g41/Σg4i ≤ y1  
x1g12/Σg1i + x2g22/Σg2i + x3g32/Σg3i + x4g42/Σg4i ≤ y2  
x1g13/Σg1i + x2g23/Σg2i + x3g33/Σg3i + x4g43/Σg4i ≤ y3  
x1g14/Σg1i + x2g24/Σg2i + x3g34/Σg3i + x4g44/Σg4i ≤ y4

# **2.3 Використані тест кейси**

Для заданого алгоритму було розроблено ряд тест кейсів. Задача вирішується для трьох варіантів вхідних даних, серед яких знаходиться варіант, який потрібно вирішити. Також на перевірку неправильного вводу даних поступають масиви з числами, які згенеровано за допомогою генератора випадкових чисел.

Додаткові варіанти чисел, для перевірки правильності виконання програми:

1. A1 = [[2, 5], [8, 5], [5, 6]]  
   B1 = [20, 40, 30]  
   C1 = [-50, -40]
2. A2 = [[4, 2, 1], [6, 0, 2], [0, 2, 4], [8, 7 , 0]]  
   B2 = [150000, 170000, 100000, 200000]  
   C2 = [-100, -150, -200]

При виконанні тест кейсів, якщо дані введено правильно, то програма виводить результат, де варіанти вхідних даних підписано (чи це дані для прикладу, чи дані для поставленої задачі) та виводиться результат підрахунку.

Якщо під час виконання тесту була виявлена помилка, то біля вхідних даних виводиться повідомлення про помилку.

# **3 ВИСНОВКИ**

В даній лабораторній роботі було розроблено програмне забезпечення для розв’язання задачі про суміші. Також було розроблено ряд тестів, які перевіряють наявність помилок і виконують певні дії з помилками, які можуть виникати при вводі даних, а також, якщо дані введено правильно. То виводиться результат обчислення.

# **ПЕРЕЛІК ПОСИЛАНЬ**

1. <https://math.semestr.ru/lp/korm.php>
2. <https://studfile.net/preview/2437366/page:3/>
3. <https://studfile.net/preview/5768610/page:13/>
4. <http://www.math-pr.com/zlp_3.php?Q1=100&Q2=150&Q3=200&Q4=&MiniMax=1&A11=4&A12=2&A13=1&LimDir1=-1&A14=150000&A21=6&A22=0&A23=2&LimDir2=-1&A24=170000&A31=0&A32=2&A33=4&LimDir3=-1&A34=100000&A41=8&A42=7&A43=0&LimDir4=-1&A44=200000&max_line_a=4&max_coln_a=3&Number_form=1&Table_type=0>

# **ДОДАТКИ**

# **Додаток А (код програми для алгоритму)**

from math import exp, log, sin, cos, tan  
from copy import deepcopy  
  
  
def main(A, B, C):  
  
 matrix = []  
  
 for i in range(len(A) + 1):  
 matrix.append([])  
 leg = len(A[0])  
  
 for j in range(leg + 2):  
 if i == 0 and (j == 0 or j == 1):  
 matrix[i].append('')  
 elif i == 0:  
 matrix[i].append(j-1)  
 else:  
 if j == 0:  
 matrix[i].append(i+len(C))  
 elif j == 1:  
 matrix[i].append(B[i-1])  
 else:  
 matrix[i].append(A[i-1][j-2])  
  
 matrix.append([])  
 ddd = len(A) + 2  
  
 for j in range(len(C) + 2):  
 if j == 0:  
 matrix[ddd-1].append("Z")  
 elif j == 1:  
 matrix[ddd-1].append(0)  
 else:  
 matrix[ddd-1].append(C[j - 2])  
  
 dop = True  
 flag = True  
  
 while(dop or flag):  
 dop = True  
 flag = False  
 for i in range(1, len(matrix)-1):  
 if matrix[i][1] < 0:  
 flag = True  
 break  
  
 if flag == True:  
 min\_val = matrix[1][1]  
 min\_val\_index = 1  
 for i in range(1, len(matrix) - 1):  
 try:  
 if matrix[i][1] < min\_val:  
 min\_val\_index = i  
 min\_val = matrix[i][1]  
 except:  
 continue  
  
 min\_elem = matrix[min\_val\_index][2]  
 min\_elem\_index = 2  
 for i in range(2, len(matrix[0])):  
 try:  
 if matrix[min\_val\_index][i] < min\_elem:  
 min\_elem\_index = i  
 except:  
 continue  
  
 else:  
 min\_elem = matrix[-1][2]  
 min\_elem\_index = 2  
 for i in range(2, len(matrix[-1])):  
 if matrix[-1][i] < min\_elem:  
 min\_elem\_index = i  
 min\_elem = matrix[-1][i]  
  
 min\_val = matrix[1][1] / matrix[1][min\_elem\_index]  
 min\_val\_index = 1  
 for i in range(1, len(matrix)-1):  
 try:  
 if matrix[i][1] / matrix[i][min\_elem\_index] < min\_val:  
 min\_val\_index = i  
 except:  
 continue  
  
 matrix[0][min\_elem\_index], matrix[min\_val\_index][0] = matrix[min\_val\_index][0], matrix[0][min\_elem\_index]  
 old\_matrix = deepcopy(matrix)  
 matrix[min\_val\_index][min\_elem\_index] = 1 / matrix[min\_val\_index][min\_elem\_index]  
  
 for i in range(1, len(matrix)):  
 if i != min\_val\_index:  
 matrix[i][min\_elem\_index] = -matrix[i][min\_elem\_index] \* matrix[min\_val\_index][min\_elem\_index]  
  
 for i in range(1, len(matrix[min\_elem\_index])):  
 if i != min\_elem\_index:  
 matrix[min\_val\_index][i] = matrix[min\_val\_index][i] \* matrix[min\_val\_index][min\_elem\_index]  
  
 for i in range(1, len(matrix)):  
 for j in range(1, len(matrix[i])):  
 if i != min\_val\_index and j != min\_elem\_index:  
 matrix[i][j] = matrix[i][j] - old\_matrix[min\_val\_index][j]\*old\_matrix[i][min\_elem\_index]\*matrix[min\_val\_index][min\_elem\_index]  
  
 dop = False  
 for i in range(2, len(matrix[-1])):  
 if matrix[-1][i] < 0:  
 dop = True  
 break  
  
 flag = False  
 for i in range(1, len(matrix) - 1):  
 if matrix[i][1] < 0:  
 flag = True  
 break  
  
 print("Результат: F(x) = ", matrix[-1][1], "\tx" + str(matrix[0][2]) + " = ", matrix[-1][2], "\tx" + str(matrix[0][3]) + " = ", matrix[-1][3])

# **Додаток Б (код програми для тестів)**

from linear import main  
import unittest  
from math import exp, sqrt  
import random  
  
  
a = [random.randint(-10, 10) for i in range(20)]  
a.append("A")  
A\_rand = [[[random.choice(a) for i in range(3)] for j in range(3)] for k in range(2)]  
B\_rand = [[random.choice(a) for j in range(3)] for k in range(2)]  
C\_rand = [[random.choice(a) for q in range(3)] for k in range(2)]  
  
# УМОВА  
A = [[1, 5], [3, 2], [2, 4], [2, 2], [1, 0]]  
B = [10, 12, 16, 10, 1]  
C = [-2, -3]  
  
# # ЛІТЕРА "А"  
# A0 = [[4, 2, 1], [6, 0, 2], [0, 2, 4], ['A', 7 , 0]]  
# B0 = [150000, 170000, 100000, 200000]  
# C0 = [-100, -150, -200]  
  
#ПРИКЛАДИ  
A1 = [[2, 5], [8, 5], [5, 6]]  
B1 = [20, 40, 30]  
C1 = [-50, -40]  
  
A2 = [[4, 2, 1], [6, 0, 2], [0, 2, 4], [8, 7 , 0]]  
B2 = [150000, 170000, 100000, 200000]  
C2 = [-100, -150, -200]  
  
class main\_test(unittest.TestCase):  
 def test\_main(self):  
 # try: x = main(A0, B0, C0)  
 # except TypeError:  
 # print("\n\n[{}, {}, {}] : НЕ ПОДХОДЯЩИЕ ДАННЫЕ".format(A0, B0, C0))  
  
 try:  
 print("\n[{},{}, {}] : УСЛОВИЕ".format(A, B, C))  
 x = main(A, B, C)  
 except:  
 pass  
  
 try:  
 print("\n[{},{}, {}] : ПРИМЕР №1".format(A1, B1, C1))  
 x = main(A1, B1, C1)  
 except:  
 pass  
  
 try:  
 print("\n[{},{}, {}] : ПРИМЕР №2".format(A2, B2, C2))  
 x = main(A2, B2, C2)  
 except:  
 pass  
  
 for i in A\_rand:  
 for j in B\_rand:  
 for q in C\_rand:  
 try:  
 x = main(i, j, q)  
 except TypeError:  
 print("\n\n[{},{}, {}]: Не подходящие данные".format(i, j, q))  
  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()

# **Додаток В (результати виконання програми)**

C:\Users\Yaros\AppData\Local\Programs\Python\Python36\python.exe "E:/учеба/4 курс/Автоматизация/5лаб/linear\_test.py"

.

----------------------------------------------------------------------

Ran 1 test in 0.001s

OK

[[[1, 5], [3, 2], [2, 4], [2, 2], [1, 0]],[10, 12, 16, 10, 1], [-2, -3]] : УСЛОВИЕ

Результат: F(x) = 7.4 x7 = 1.4 x3 = 0.6000000000000001

[[[2, 5], [8, 5], [5, 6]],[20, 40, 30], [-50, -40]] : ПРИМЕР №1

Результат: F(x) = 265.2173913043478 x5 = 3.043478260869567 x4 = 4.347826086956521

[[[4, 2, 1], [6, 0, 2], [0, 2, 4], [8, 7, 0]],[150000, 170000, 100000, 200000], [-100, -150, -200]] : ПРИМЕР №2

Результат: F(x) = 7320000.0 x7 = 8.0 x5 = 6.0

[[[1, 8, 9], [6, 3, 2], [1, 3, 6]],[4, 'A', 6], ['A', 6, 8]]: Не подходящие данные

[[[1, 8, 9], [6, 3, 2], [1, 3, 6]],[4, 'A', 6], [8, 1, -4]]: Не подходящие данные

[[[1, 8, 9], [6, 3, 2], [1, 3, 6]],[4, 6, 4], ['A', 6, 8]]: Не подходящие данные

Результат: F(x) = 1.7777777777777777 x1 = 8.444444444444445 x2 = 4.555555555555555

[[[4, 6, 8], [4, 1, 4], [7, 'A', 2]],[4, 'A', 6], ['A', 6, 8]]: Не подходящие данные

[[[4, 6, 8], [4, 1, 4], [7, 'A', 2]],[4, 'A', 6], [8, 1, -4]]: Не подходящие данные

[[[4, 6, 8], [4, 1, 4], [7, 'A', 2]],[4, 6, 4], ['A', 6, 8]]: Не подходящие данные

[[[4, 6, 8], [4, 1, 4], [7, 'A', 2]],[4, 6, 4], [8, 1, -4]]: Не подходящие данные

Process finished with exit code 0

# **Додаток Г (підтвердження правильності роботи алгоритму)**

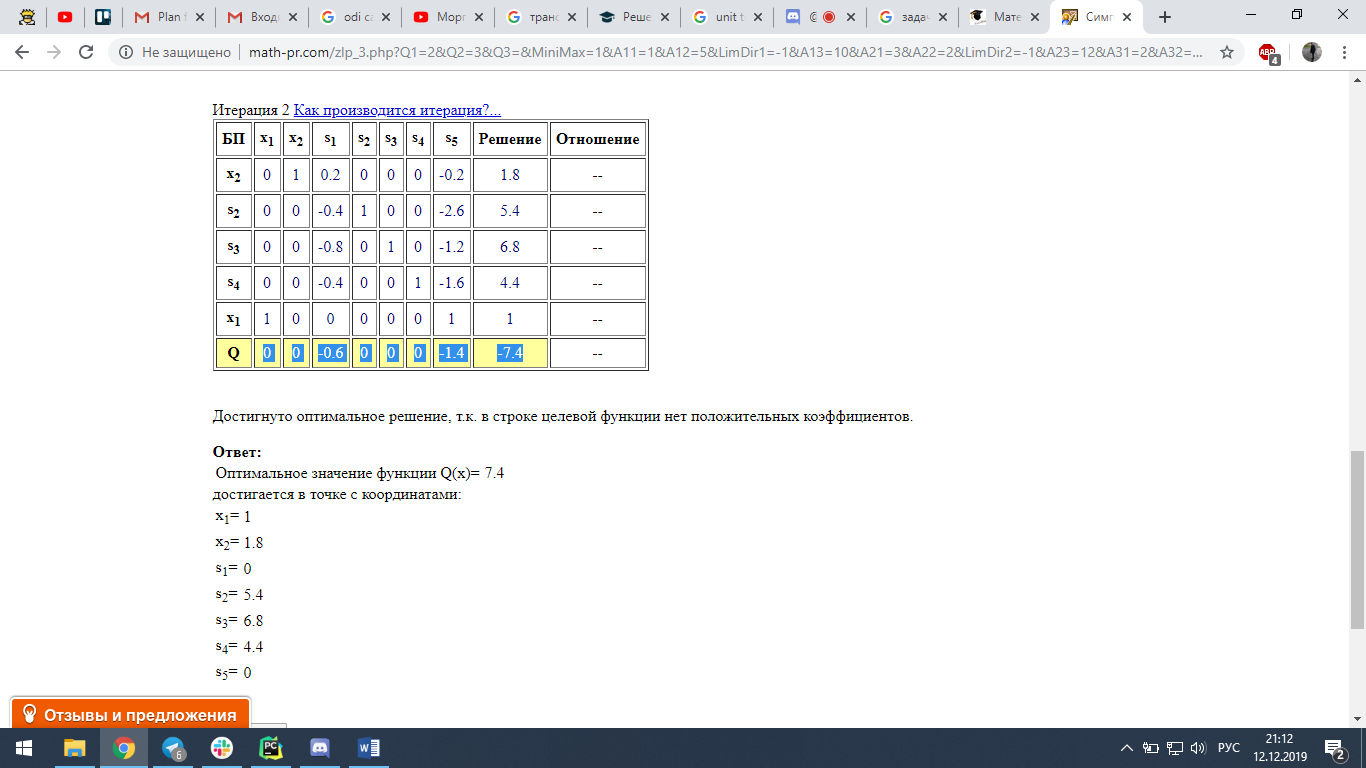
1. Для поставленої задачі:

Результат виконання програми:

[[[1, 5], [3, 2], [2, 4], [2, 2], [1, 0]],[10, 12, 16, 10, 1], [-2, -3]] : УСЛОВИЕ

Результат: F(x) = 7.4 x7 = 1.4 x3 = 0.6000000000000001

Результат з онлайн джерела:



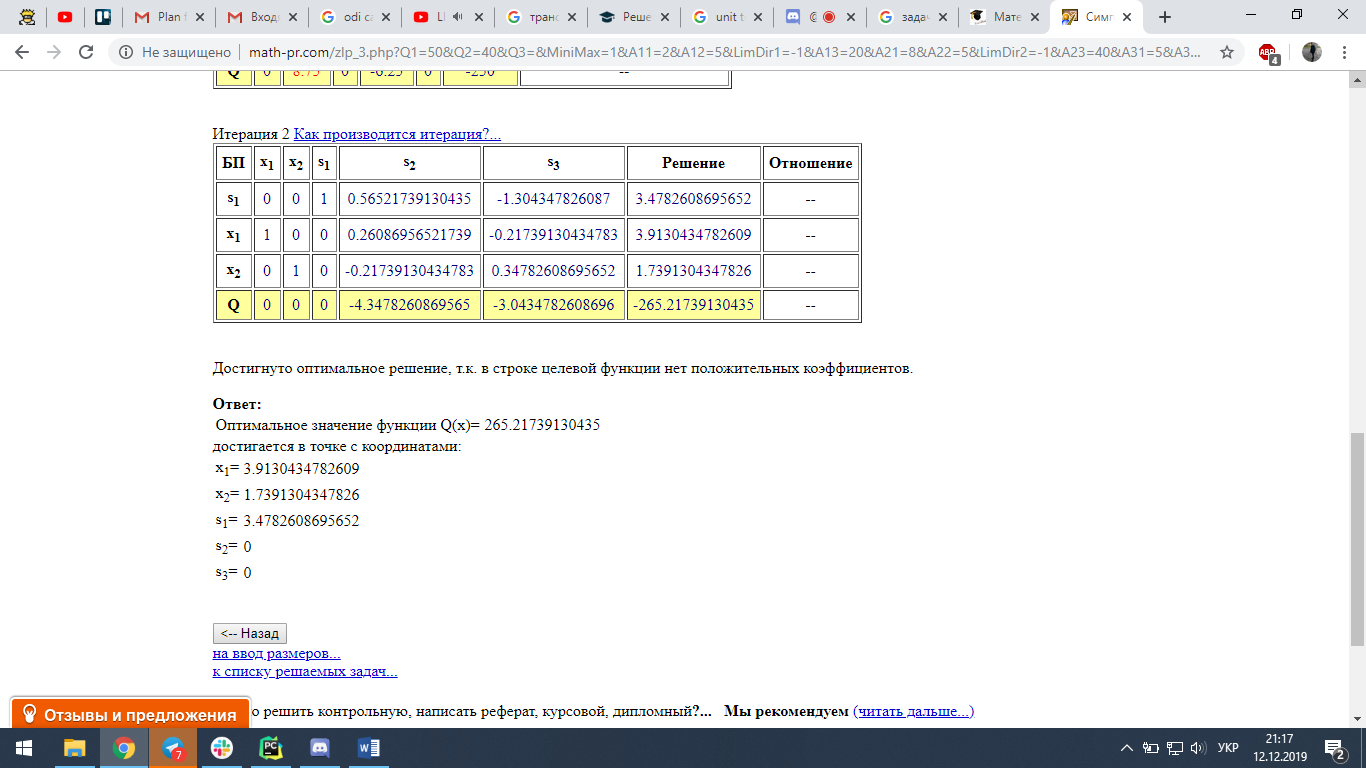
1. Для першого прикладу:

Результат виконання програми:

[[[2, 5], [8, 5], [5, 6]],[20, 40, 30], [-50, -40]] : ПРИМЕР №1

Результат: F(x) = 265.2173913043478 x5 = 3.043478260869567 x4 = 4.347826086956521

Результат з онлайн джерела:



1. Для першого прикладу:

Результат виконання програми:

[[[4, 2, 1], [6, 0, 2], [0, 2, 4], [8, 7, 0]],[150000, 170000, 100000, 200000], [-100, -150, -200]] : ПРИМЕР №2

Результат: F(x) = 7320000.0 x7 = 8.0 x5 = 6.0

Результат з онлайн джерела:

