Національний авіаційний університет

Факультет кібербезпеки, комп’ютерної та програмної інженерії

**ЗВІТ**

**по лабораторній роботі No 1**

**Симплексний метод розв’язання задач лінійного програмування.**

Дисципліна: «Методи оптимізацій та дослідження операцій»

Кафедра: прикладної математики

ОС: бакалавр

Спеціальність: 113 «Прикладна математика»

ОПП: «Прикладне програмне забезпечення»

Виконав: здобувач вищої освіти 4 курсу. 451 групи

Архіпов Олексій Тімурович

Перевірив: Хребет Валерій Григорович

Київ 2023

**Тема:** Симплексний метод розв’язання задач лінійного програмування.

**Мета:** Ознайомитися з методами умовної оптимізації лінійної цільової функції, засвоїти алгоритм симплекс-методу.

**Завдання:**

1. Розв’яжіть задачу лінійного програмування симплекс методом;

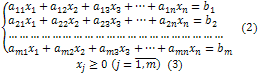
2. Скласти програму знаходження оптимального значення цільової функції (згідно варіанту).

**Порядок обчислень**

Якщо серед векторів https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy1.gif системи обмежень є рівно https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy2.gif одиничних, то **розв’язок задачі лінійного програмування** можна отримати за допомогою **симплекс методу**. Однак, не завжди дана умова виконується. Відмітимо, що саме в такому випадку використовується **метод штучного базису**. Отже, нехай потрібно знайти максимальне значення цільової функції:

https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy3.gif

при обмеженнях :

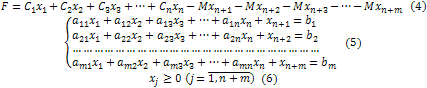


І серед векторів:

https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy18.gif

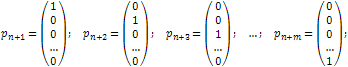
немає https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy2.gif одиничних.

Тоді, задачу (1) — (3) приводимо до еквівалентної задачі наступним чином:



де https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy8.gif — достатньо велике число, значення якого, практично, не задається, а змінні https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy9.gif називаються штучними. Тоді задачу (4) — (6) називають розширеною задачею по відношенню до задачі (1) — (3).

Вектори, що відповідають коефіцієнтам при штучних змінних, а саме:



називаються штучними векторами.

Тепер розширена задача матиме опорний план:

https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy11.gif

Маючи початковий опорний план, за допомогою **симплекс методу** ми отримуємо розв’язок розширеної задачі (4) — (6).

**Теорема:** якщо в **оптимальному розв’язку** https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy12.gif розширеної задачі (4) — (6), штучні змінні https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy13.gif, то пан https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy14.gif буде служити **оптимальним розв’язком задачі лінійного програмування** (1) — (3).

Характерною ознакою **симплекс таблиць** застосованих до **методу штучного базису** є наявніст https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy15.gif-го та https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy16.gif-го рядків. Де https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy15.gif-й рядок містить доданок без коефіцієнта https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy8.gif, а https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy16.gif-й — містить коефіцієнти при https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy8.gif.

Початок розвя’зку в **симплекс таблиці** здійснюється по https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy16.gif-му рядку (аналогічно, як і у **симплекс методі** по https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy15.gif-му). І ітераційний процес продовжується до тих пір, поки:

1. Усі штучні вектори не будуть виключені із базису.
2. Елементи https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy16.gif-го рядка не міститимуть від’ємних значень, і не всі штучні вектори виключені з базису.

В першому випадку знайдено деякий опорний план розширеної задачі, а знаходження **оптимального плану** ведеться за **симплекс методом** по https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy15.gif-му рядку.

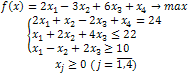
В другому випадку, якщо в стовпці https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy17.gif знаходиться від’ємне значення, то **задача лінійного програмування** немає розв’язку.

Якщо в стовпці https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy17.gif стоїть значення *0*, то дана задача є виродженою і її розв’язок містить принаймі один чи декілька штучних векторів.

**Результати тестування**

Розглянемо результати тестування на прикладах взятих із джерел інтернету.

Приклад 1:



https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy43.gif

https://www.mathros.net.ua/wp-content/uploads/2012/09/metod_shtychnoho_bazusy44.gif

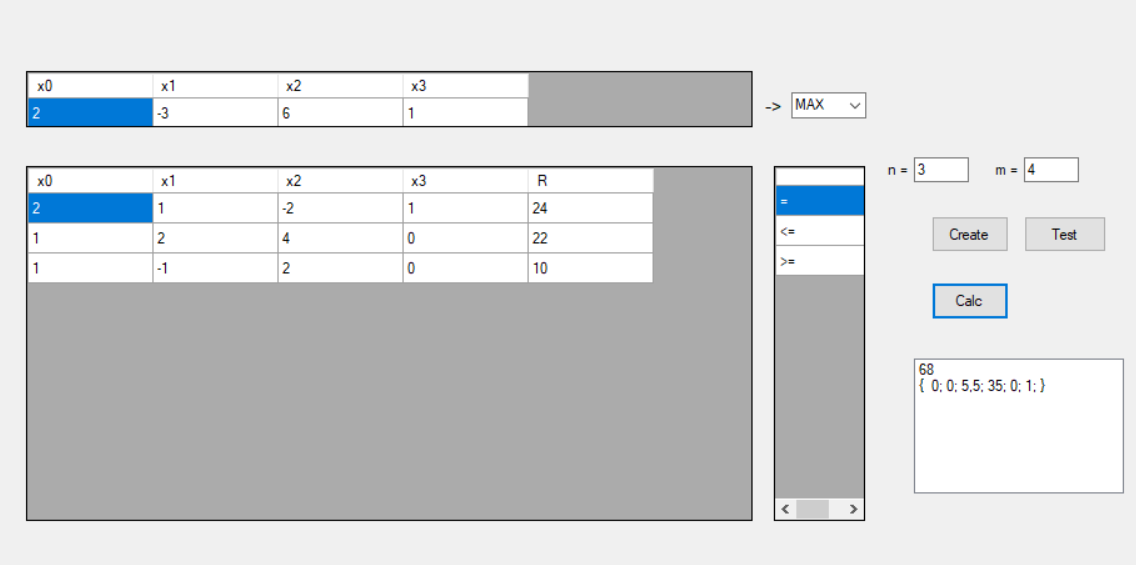


Рисунок 1. Приклад 1

Як видно з рисунку 1 значення співпадають.

Приклад 2:

Знайти максимум цільової функції: при умовахhttps://stud.com.ua/imag/econom/fed_emmpm/image296.jpg https://stud.com.ua/imag/econom/fed_emmpm/image297.jpg

https://stud.com.ua/imag/econom/fed_emmpm/image298.jpg

отриманий оптимальний план вихідної задачі (4, 0, 2) і максимальне значення цільової функції https://stud.com.ua/imag/econom/fed_emmpm/image302.jpgmax

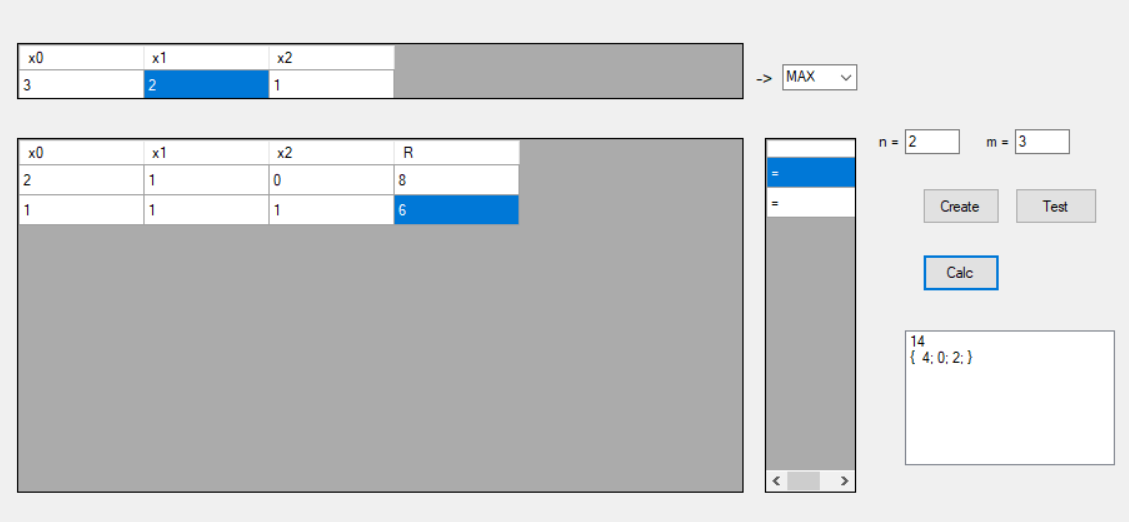
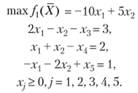


Рисунок 2. Приклад 2

Як видно з рисунку 2 значення співпадають.

Приклад 3:



https://stud.com.ua/imag/econom/fed_emmpm/image307.jpg(вихідні змінні), https://stud.com.ua/imag/econom/fed_emmpm/image308.jpg(додаткові змінні), при цьомуhttps://stud.com.ua/imag/econom/fed_emmpm/image309.jpg

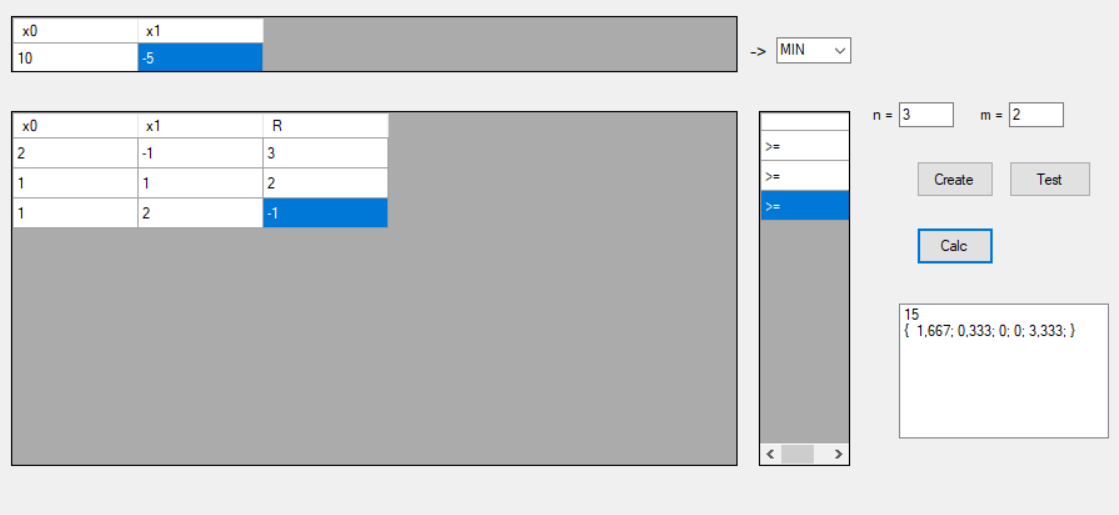


Рисунок 3. Приклад 3

Як видно з рисунку 3 значення співпадають.

**Висновоки**

1. Розв’язав задачу лінійного програмування симплекс методом;
2. Склав програму знаходження оптимального значення цільової функції (згідно варіанту).

**Код програми**

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace lab1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

dgMatrix.RowHeadersVisible = false;

dg2.RowHeadersVisible = false;

dgF.RowHeadersVisible = false;

dgMatrix.AllowUserToAddRows = false;

dg2.AllowUserToAddRows = false;

dgF.AllowUserToAddRows = false;

}

int n = 0; int m = 0;

List<List<double>> P;

List<List<double>> X\_i;

List<double> P0, F;

List<string> Znak;

string max\_or\_min = "";

void CreateMatrix2()

{

n = Convert.ToInt32(tbn.Text);

m = Convert.ToInt32(tbm.Text);

dgMatrix.Rows.Clear();

dgMatrix.Columns.Clear();

dg2.Rows.Clear();

dg2.Columns.Clear();

dgF.Rows.Clear();

dgF.Columns.Clear();

for (int i = 0; i < m; i++)

{

dgMatrix.Columns.Add("x" + i.ToString(), "x" + i.ToString());

dgF.Columns.Add("x" + i.ToString(), "x" + i.ToString());

}

dgMatrix.Columns.Add("R", "R");

dg2.Columns.Add("", "");

dgMatrix.Rows.Add(n);

dg2.Rows.Add(n);

dgF.Rows.Add(1);

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

dgMatrix.Rows[i].Cells[j].Value = "1"; // записываем значеня в соответсвующие места

if(i == 0) dgF.Rows[0].Cells[j].Value = "1";

}

dg2.Rows[i].Cells[0].Value = "=";

}

}

void CreateMatrixTest()

{

n = 3;

m = 4;

dgMatrix.Rows.Clear();

dgMatrix.Columns.Clear();

dg2.Rows.Clear();

dg2.Columns.Clear();

dgF.Rows.Clear();

dgF.Columns.Clear();

for (int i = 0; i < m; i++)

{

dgMatrix.Columns.Add("x" + i.ToString(), "x" + i.ToString());

dgF.Columns.Add("x" + i.ToString(), "x" + i.ToString());

}

dgMatrix.Columns.Add("R", "R");

dg2.Columns.Add("", "");

dgMatrix.Rows.Add(n);

dg2.Rows.Add(n);

dgF.Rows.Add(1);

int j = 0;

dgMatrix.Rows[j].Cells[0].Value = "2";

dgMatrix.Rows[j].Cells[1].Value = "1";

dgMatrix.Rows[j].Cells[2].Value = "-2";

dgMatrix.Rows[j].Cells[3].Value = "1";

dgMatrix.Rows[j].Cells[4].Value = "24";

dg2.Rows[j].Cells[0].Value = "=";

dgF.Rows[j].Cells[0].Value = "2";

dgF.Rows[j].Cells[1].Value = "-3";

dgF.Rows[j].Cells[2].Value = "6";

dgF.Rows[j].Cells[3].Value = "1";

j = 1;

dgMatrix.Rows[j].Cells[0].Value = "1";

dgMatrix.Rows[j].Cells[1].Value = "2";

dgMatrix.Rows[j].Cells[2].Value = "4";

dgMatrix.Rows[j].Cells[3].Value = "0";

dgMatrix.Rows[j].Cells[4].Value = "22";

dg2.Rows[j].Cells[0].Value = "<=";

j = 2;

dgMatrix.Rows[j].Cells[0].Value = "1";

dgMatrix.Rows[j].Cells[1].Value = "-1";

dgMatrix.Rows[j].Cells[2].Value = "2";

dgMatrix.Rows[j].Cells[3].Value = "0";

dgMatrix.Rows[j].Cells[4].Value = "10";

dg2.Rows[j].Cells[0].Value = ">=";

cb.SelectedIndex = 1;

}

void ReadAll()

{

n = Convert.ToInt32(tbn.Text);

m = Convert.ToInt32(tbm.Text);

P = new List<List<double>>();

X\_i = new List<List<double>>();

P0 = new List<double>();

F = new List<double>();

Znak = new List<string>();

for (int i = 0; i < n; i++)

{

List<double> temp = new List<double>();

for (int j = 0; j < m; j++)

{

temp.Add(Convert.ToDouble(dgMatrix.Rows[i].Cells[j].Value));

//P[i, j] = Convert.ToDouble(dgMatrix.Rows[i].Cells[j].Value);

if (i == 0) F.Add(Convert.ToDouble(dgF.Rows[i].Cells[j].Value));

}

P.Add(temp);

P0.Add(Convert.ToDouble(dgMatrix.Rows[i].Cells[m].Value));

Znak.Add(Convert.ToString(dg2.Rows[i].Cells[0].Value));

}

max\_or\_min = (string)cb.SelectedItem;

for(int i = 0; i < m; i++)

{

List<double> temp = new List<double>();

for (int j = 0; j < n; j++)

{

temp.Add(Convert.ToDouble(dgMatrix.Rows[j].Cells[i].Value));

}

X\_i.Add(temp);

}

}

void Calc()

{

ReadAll();

listBox1.Items.Clear();

//List<List<double>> new\_p = new List<List<double>>();

List<int> index = new List<int>();

for(int i = 0; i < P0.Count(); i++)//Змінюємо всі вільні члени на додатні

{

if (P0[i] < 0)

{

index.Add(i);

P0[i] \*= -1;

}

}

if(index.Count > 0)

{

for (int i = 0; i < index.Count; i++)

{

for(int j = 0; j < X\_i.Count; j++)

{

X\_i[j][index[i]] \*= -1;

}

if (Znak[index[i]] == "<=") Znak[index[i]] = ">=";

else if (Znak[index[i]] == ">=") Znak[index[i]] = "<=";

}

}

for(int i = 0; i < Znak.Count; i++)//Змінюємо нерівність на рівність

{

List<double> temp = new List<double>();

for(int j = 0; j < Znak.Count; j++)

{

if (Znak[i] == ">=")

{

if (i == j) temp.Add(-1);

else temp.Add(0);

}

else if(Znak[i] == "<=")

{

if (i == j) temp.Add(1);

else temp.Add(0);

}

}

if (temp.Count > 0)

{

X\_i.Add(temp);

F.Add(0);

}

}

int baz\_count = Znak.Count;

index.Clear();

List<int> index2 = new List<int>();

List<int> index3 = new List<int>();

List<int> indexM = new List<int>();

for (int i = 0; i < X\_i.Count; i++) //Починаємо рахувати кількість базисних зміних

{

int c\_0 = 0, c = 0;

for (int j = 0; j < X\_i[i].Count; j++)

{

if (X\_i[i][j] < 0)

{

if(j < X\_i[i].Count - 1) break;

else

{

c\_0++;

break;

}

}

else

{

if (X\_i[i][j] == 0) c\_0++;

else c = j;

}

}

if (c\_0 == X\_i[i].Count - 1)

{

index.Add(i);

index2.Add(c);

}

}

if(index.Count < baz\_count)

{

for(int i = 0; i < Znak.Count(); i++)//Визначаємо строку де немає bазисної зміної

{

if (!index2.Contains(i)) index3.Add(i);

}

for(int i = 0; i < index3.Count; i++)//Додаємо зміні М

{

List<double> temp = new List<double>();

for(int j = 0; j < X\_i[i].Count; j++)

{

if (j == index3[i]) temp.Add(1);

else temp.Add(0);

}

X\_i.Add(temp);

index.Add(X\_i.Count-1);

indexM.Add(X\_i.Count - 1);

index2.Add(index3[i]);

if (max\_or\_min == "MAX") F.Add(-1);

else if (max\_or\_min == "MIN") F.Add(1);

}

if(max\_or\_min == "MIN")

{

for(int i = 0; i < F.Count; i++)

{

F[i] \*= -1;

}

}

double sign = 0;

List<Tuple<int, int>> tupleM = new List<Tuple<int, int>>();//1 індекс - номер столбца; 2 - номер стрічки

for(int i = 0; i < index3.Count; i++)

{

Tuple<int, int> tuple = new Tuple<int, int>(index3[i], indexM[i]);

tupleM.Add(tuple);

}

List<double> deltaM = new List<double>();

List<double> delta = new List<double>();

bool t = false;

List<Tuple<double, int, int>> Cb;

int iter = 0;

do

{

iter++;

Cb = new List<Tuple<double, int, int>>();// Створюю Cb где первое значение єто значения в рядке Cb, второе это рядок где стоит базис, третье - строка где стоит еденица

Tuple<double, int, int> cb = new Tuple<double, int, int>(0, 0, 0); ;

for (int i = 0; i < index.Count; i++)

{

if(index2.Contains(i)) cb = new Tuple<double, int, int>(F[index[i]], index[i], index2[i]);

else if (index3.Contains(i)) cb = new Tuple<double, int, int>(F[index[i]], index[i], index3[i]);

Cb.Add(cb);

}

Cb = Cb.OrderBy(cb1 => cb1.Item3).ToList();

List<int> num\_N = new List<int>();

List<int> num\_not\_N = new List<int>();

List<int> num\_not\_N\_sort = new List<int>();

for (int i = 0; i < index2.Count; i++)// Дивлюся на якій стрічці стоїть М на якій ні

{

if (!index3.Contains(index2[i]))

{

num\_not\_N.Add(index2[i]);

num\_not\_N\_sort.Add(index2[i]);

}

}

for (int i = 0; i < tupleM.Count; i++)

{

int c = 0;

for(int j = 0; j < Cb.Count; j++)

{

if (Cb[j].Item2 == tupleM[i].Item2)

{

c = 1;

break;

}

}

if(c == 1) num\_N.Add(tupleM[i].Item1);

}

deltaM = new List<double>();

delta = new List<double>();

num\_not\_N\_sort.Sort();

if (num\_N.Count > 0)//якщо М у базисі

{

sign = F[F.Count - 1];

for (int i = 0; i < F.Count + 1; i++)// рахуємо дельта(значення біля М)

{

double c = 0;

for(int j = 0; j < num\_N.Count; j++)

{

if(i == 0)

{

c += (P0[num\_N[j]] \* sign);

}

else

{

c += (X\_i[i - 1][num\_N[j]] \* sign);

}

}

deltaM.Add(c);

}

for (int i = 0; i < index.Count; i++)

{

deltaM[index[i] + 1] = 0;

}

}

for (int i = 0; i < F.Count + 1; i++)//рахуємо просто дельта

{

double c = 0;

for (int j = 0; j < num\_not\_N.Count; j++)

{

if (i == 0)

{

c += (P0[num\_not\_N\_sort[j]] \* Cb[num\_not\_N\_sort[j]].Item1);

}

else

{

c += (X\_i[i - 1][num\_not\_N[j]] \* Cb[num\_not\_N[j]].Item1);

}

}

if (i == 0) delta.Add(c);

else delta.Add(c - F[i - 1]);

}

for (int i = 0; i < index.Count; i++)

{

delta[index[i] + 1] = 0;

}

double max = 0;

int ind = 0;

if (deltaM.Count > 0)//якщо дельта М не порожня(тобто у базисі ще залишилося М)

{

max = deltaM[1];//шукаємо найбільше від'єме число у дельта М

for (int i = 2; i < deltaM.Count; i++)

{

if (deltaM[i] < max)

{

max = deltaM[i];

ind = i - 1;

}

}

}

double max2 = delta[1];

if (max >= 0)//якщо зміні М всі додатні у дельта М, то шукаємо від'ємні у дельта

{

for (int i = 2; i < delta.Count; i++)

{

if (delta[i] < max2)

{

max2 = delta[i];

ind = i - 1;

}

}

if (max2 >= 0)

{

if (max\_or\_min == "MIN")

listBox1.Items.Add(delta[0] \* (-1));

else listBox1.Items.Add(delta[0]);

string str = "{ ";

int count = 0;

Cb = Cb.OrderBy(cb1 => cb1.Item2).ToList();

for (int i = 0; i < Cb.Count; i++)

{

for (int j = count; j < F.Count; j++)

{

if (j == Cb[i].Item2)

{

str += " " + P0[Cb[i].Item3].ToString() + ";";

count++;

break;

}

else

{

str += " 0;";

count++;

}

}

}

str += " }";

listBox1.Items.Add(str);

break;

}

}

if(deltaM.Count > 0 && max > 0 && max2 < 0)

{

if (max\_or\_min == "MIN")

listBox1.Items.Add(deltaM[0] \* (-1) + " \* M +(" + delta[0] \* (-1) + ")");

else listBox1.Items.Add(deltaM[0] + " \* M +(" + delta[0] + ")");

break;

}

if (max < 0 || max2 < 0)//якщо у одному з дельта є від'ємне то починаемо симплекс метод

{

List<double> Q = new List<double>();

for (int i = 0; i < X\_i[i].Count; i++)//знаходимо КЬЮ (мінімальне відношення Cb/aii)

{

if (X\_i[ind][i] <= 0) Q.Add(1000000000000);

else Q.Add(P0[i] / X\_i[ind][i]);

}

int ind\_q = Q.IndexOf(Q.Min());

double coef = X\_i[ind][ind\_q];//знаходимо позицію для головного числа

List<List<double>> X\_i\_new = new List<List<double>>();

List<double> P0\_new = new List<double>();

for (int i = 0; i < X\_i.Count; i++)

{

List<double> temp = new List<double>();

for (int j = 0; j < X\_i[i].Count; j++)

{

temp.Add(X\_i[i][j]);

if(i == 0)

{

P0\_new.Add(P0[j]);

}

}

X\_i\_new.Add(temp);

}

for (int i = 0; i < X\_i.Count; i++)//робимо симплекс метод

{

for (int j = 0; j < X\_i[i].Count; j++)

{

if (i == ind && j != ind\_q) X\_i\_new[i][j] = 0;

else if (i != ind && j == ind\_q) X\_i\_new[i][j] /= coef;

else if (i == ind && j == ind\_q) X\_i\_new[i][j] = 1;

else X\_i\_new[i][j] = ((X\_i[i][j] \* coef) - (X\_i[ind][j] \* X\_i[i][ind\_q])) / coef;

if (i == ind)

{

if (j == ind\_q) P0\_new[j] /= coef;

else P0\_new[j] = ((P0[j] \* coef) - (P0[ind\_q] \* X\_i[i][j])) / coef;

}

}

}

X\_i.Clear();

P0.Clear();

for (int i = 0; i < X\_i\_new.Count; i++)

{

List<double> temp = new List<double>();

for (int j = 0; j < X\_i\_new[i].Count; j++)

{

temp.Add(X\_i\_new[i][j]);

if (i == 0)

{

P0.Add(P0\_new[j]);

}

}

X\_i.Add(temp);

}

index.Clear();

index2.Clear();

for (int i = 0; i < X\_i.Count; i++) //Починаємо рахувати кількість базисних зміних

{

int c\_0 = 0, c = 0;

for (int j = 0; j < X\_i[i].Count; j++)

{

if (X\_i[i][j] < 0)

{

if (j < X\_i[i].Count - 1) break;

else

{

c\_0++;

break;

}

}

else

{

if (X\_i[i][j] == 0) c\_0++;

else if (X\_i[i][j] == 1) c = j;

else break;

}

}

if (c\_0 == X\_i[i].Count - 1)

{

index.Add(i);

index2.Add(c);

}

}

if(tupleM.Count > 0)

{

for(int i = 0; i < tupleM.Count; i++)

{

int c;

if (!index.Contains(tupleM[i].Item2))

{

c = tupleM[i].Item2;

X\_i.RemoveAt(c);

F.RemoveAt(c);

index3.Remove(tupleM[i].Item1);

tupleM.RemoveAt(i);

}

}

for(int i = 0; i < index.Count; i++)

{

if (index[i] > X\_i.Count - 1) index[i] -= 1;

}

for (int i = 0; i < tupleM.Count; i++)

{

if (tupleM[i].Item2 > X\_i.Count - 1)

{

Tuple<int, int> tuple = new Tuple<int, int>(tupleM[i].Item1, tupleM[i].Item2 - 1);

tupleM[i] = tuple;

}

}

}

if (index3.Count == 0 && delta.Skip(1).Min() >= 0 && deltaM.Count == 0)

{

if (max\_or\_min == "MIN")

listBox1.Items.Add(delta[0] \* (-1));

else listBox1.Items.Add(delta[0]);

string str = "{ ";

int count = 0;

Cb = Cb.OrderBy(cb1 => cb1.Item2).ToList();

for (int i = 0; i < Cb.Count; i++)

{

for (int j = count; j < F.Count; j++)

{

if (j == Cb[i].Item2)

{

str += " " + P0[Cb[i].Item3].ToString() + ";";

count++;

break;

}

else

{

str += " 0;";

count++;

}

}

}

str += " }";

listBox1.Items.Add(str);

break;

}

}

else

{

if (max\_or\_min == "MIN")

listBox1.Items.Add(delta[0] \* (-1));

else listBox1.Items.Add(delta[0]);

string str = "{ ";

int count = 0;

Cb = Cb.OrderBy(cb1 => cb1.Item2).ToList();

for (int i = 0; i < Cb.Count; i++)

{

for (int j = count; j < F.Count; j++)

{

if (j == Cb[i].Item2)

{

str += " " + P0[Cb[i].Item3].ToString() + ";";

count++;

break;

}

else

{

str += " 0;";

count++;

}

}

}

str += " }";

listBox1.Items.Add(str);

break;

}

if(iter > 20)

{

listBox1.Items.Add("Розв'язків немає або їх безліч");

break;

}

} while (t == false);

}

else if(index.Count == baz\_count)

{

listBox1.Items.Add("Error");

}

}

private void btnCalc\_Click(object sender, EventArgs e)

{

Calc();

}

private void btnCreate\_Click(object sender, EventArgs e)

{

CreateMatrix2();

}

private void btnTest\_Click(object sender, EventArgs e)

{

CreateMatrixTest();

}

}

}