МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
КАФЕДРА №51

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

по курсу: МЕТОДЫ МОДЕЛИРОВАНИЯ И ОПТИМИЗАЦИИ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР.	5711М		Пятаков В.С.
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2017

Цель работы

Разработать математическую модель и решить задачу линейного программирования симплекс-методом и графическим методом.

Постановка задачи однопараметрической оптимизации

Задачей линейного программирования называется оптимизационная задача, в которой целевая функция линейна на множестве линейных ограничений:

$$F = \sum_{j=1}^{n} c_j x_j \to \min$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_j, i = \overline{1,m}, x_j \ge 0.$$

где F – целевая функция;

$a_{ij} b_i$ – постоянные коэффициенты для уравнений;

c_j – постоянный коэффициент для целевой функции;

x_j – неизвестные переменные.

Ограничения, накладываемые на координаты x_j могут быть равенствами и неравенствами.

Порядок выполнения работы

1. Реализовать алгоритм симплекс-метода решения задачи линейного программирования для функции оптимизации, приведенной в таблице.
2. Зафиксировать количественные характеристики симплекс-метода, полученные в результате решения задачи, последнюю симплекс-таблицу и значение целевой функции.
3. Решить эту же задачу линейного программирования графическим методом.
4. Сравнить результаты целевой функции, полученной симплекс-методом и графическим методом, оценить интервальную ошибку.
5. Оформить отчет о проделанной работе.

Описание методов

Симплекс-метод

Движение к точке оптимума осуществляется путем перехода от одной угловой точки к соседней, которая ближе и быстрее приближает к $X_{опт}$. Такую схему перебора точек, называемую симплекс-метод, предложил Р. Данцигом.

Угловые точки характеризуются *m* базисными переменными, поэтому переход от одной угловой точки к соседней возможно осуществить сменой в базисе только одной базисной переменной на переменную из небазиса.

Алгоритм

1. Составление первого опорного плана. Переход к канонической форме задачи линейного программирования путем введения неотрицательных дополнительных балансовых переменных.
2. Проверка плана на оптимальность. Если найдется хотя бы один коэффициент индексной строки меньше нуля, то план не оптимальный, и его необходимо улучшить.
3. Определение ведущих столбца и строки. Из отрицательных коэффициентов индексной строки выбирается наибольший по абсолютной величине. Затем элементы столбца свободных членов симплексной таблицы делит на элементы того же знака ведущего столбца.
4. Построение нового опорного плана. Переход к новому плану осуществляется в результате пересчета симплексной таблицы методом Жордана—Гаусса.

Блок схема

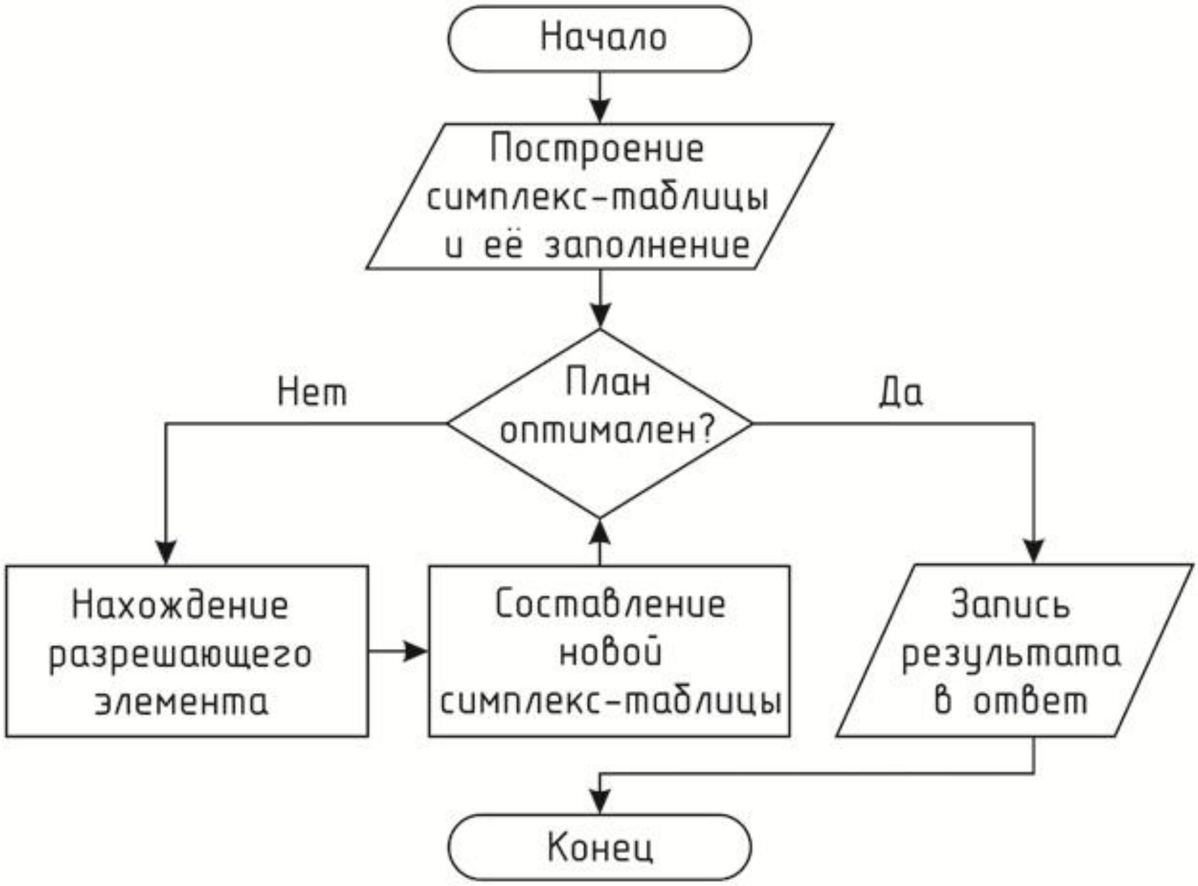

рисунок 1 - блок схема алгоритма симпекс-метода

Графический метод

В линейном программировании используется графический метод, с помощью которого определяют выпуклые множества (многогранник решений). Если основная задача линейного программирования имеет оптимальный план, то целевая функция принимает значение в одной из вершин многогранника решений

Алгоритм

1. Построить многоугольник решений.
2. Построить вектор c=(c 1 , c 2) и перпендикулярно ему провести линию уровня линейной функции F, например, F=0.
3. Параллельным перемещением прямой F=0 в направлении вектора c(-c) найти точку A max (B min), в которой F достигает максимума (минимума).
4. Решая совместно уравнения прямых, пересекающихся в точке оптимума, найти ее координаты.
5. Вычислить F max (F min).

Полученные результаты

Вариант 14

$$f(x) = x_1 + 3x_2 \rightarrow \min$$

$$\begin{cases} x_1 - x_2 \le 3 \\ 2x_1 + x_2 \ge 3 \\ x_1 - 3x_2 \le 1 \\ x_1, x_2 \ge 0 \end{cases}$$

Симплекс-метод.

В 1-м неравенстве вводим базисную переменную x_3. В 2-м неравенстве вводим базисную переменную x_4 со знаком минус. В 3-м неравенстве вводим базисную переменную x_5.

Приведем систему к единичной матрице методом жордановских преобразований.

1. В качестве базовой переменной можно выбрать x_3.
2. В качестве базовой переменной можно выбрать x_4.
3. В качестве базовой переменной можно выбрать x_5.

Базисное решение называется допустимым, если оно неотрицательно.

Таблица 1 - Канонический вид симплекс таблицы

Базис	**B**	**x_1**	**x_2**	**x_3**	**x_4**	**x_5**
x_3	6	3	0	1	-1	0
x_2	3	2	1	0	-1	0
x_5	10	7	0	0	-3	1
F(X0)	0	5	0	0	-3	0

Основной алгоритм

1. Проверка критерия оптимальности.

Текущий опорный план не оптимален, так как в индексной строке находятся положительные коэффициенты.

2. Определение новой базисной переменной.

В качестве ведущего выберем столбец, соответствующий переменной x_1, так как это наибольший коэффициент.

3. Определение новой свободной переменной.

Вычислим значения D_i по строкам как частное от деления: b_i / a_{i1} и из них выберем наименьшее.

Таблица 2 - опорный план симплекс таблицы

Базис	**B**	**x_1**	**x_2**	**x_3**	**x_4**	**x_5**	**min**
x_3	6	3	0	1	-1	0	2
x_2	3	2	1	0	-1	0	1.5
x_5	10	7	0	0	-3	1	1.43
F(X1)	0	5	0	0	-3	0	0

4. Пересчет симплекс-таблицы.

Формируем следующую часть симплексной таблицы. Вместо переменной x_5 в план 1 войдет переменная x_1.

Строка, соответствующая переменной x_1 в плане 1, получена в результате деления всех элементов строки x_5 плана 0 на разрешающий элемент РЭ=7. На месте разрешающего элемента получаем 1. В остальных клетках столбца x_1 записываем нули.

Таким образом, в новом плане 1 заполнены строка x_1 и столбец x_1. Все остальные элементы нового плана 1, включая элементы индексной строки, определяются по правилу прямоугольника.

Для этого выбираем из старого плана четыре числа, которые расположены в вершинах прямоугольника и всегда включают разрешающий элемент РЭ. НЭ = СЭ - (А*В)/РЭ СТЭ - элемент старого плана, РЭ - разрешающий элемент (7), А и В - элементы старого плана, образующие прямоугольник с элементами СТЭ и РЭ.

Таблица 3- новая симплекс таблица

Базис	**B**	x_1	x_2	x_3	x_4	x_5
x_3	1.7	0	0	1	0.3	-0.43
x_2	0.14	0	1	0	-0.14	-0.3
x_1	1.43	1	0	0	-0.43	0.14
F(X1)	-7.12	0	0	0	-0.9	-0.71

Проверка критерия оптимальности

Среди значений индексной строки нет положительных. Поэтому эта таблица определяет оптимальный план задачи.

Таблица 4 - итоговая симплекс таблица

Базис	**B**	x_1	x_2	x_3	x_4	x_5
x_3	1.7	0	0	1	0.3	-0.43
x_2	0.14	0	1	0	-0.14	-0.3
x_1	1.43	1	0	0	-0.43	0.14
F(X2)	-7.12	0	0	0	-0.9	-0.71

Оптимальный план можно записать так:

$x_1 = 1.43$, $x_2 = 0.14$;

$F(X) = 1 \bullet 1.43 + 3 \bullet 0.14 = 1.85$.

Графический метод

Шаг №1. Построим область допустимых решений, т.е. решим графически систему неравенств. Для этого построим каждую прямую и определим полуплоскости, заданные неравенствами (полуплоскости обозначены штрихом).

Построим уравнение x_1-x_2 = 3 по двум точкам. Для нахождения первой точки приравниваем x_1 = 0. Находим x_2 = -3. Для нахождения второй точки приравниваем x_2 = 0. Находим x_1 = 3. Соединяем точку (0;-3) с (3;0) прямой линией. Определим полуплоскость, задаваемую неравенством. Выбрав точку (0; 0), определим знак неравенства в полуплоскости:1 • 0 - 1 • 0 - 3 ≤ 0, т.е. x_1-x_2 - 3≤ 0 в полуплоскости *ниже*прямой.

Построим уравнение 2x_1+x_2 = 3 по двум точкам. Для нахождения первой точки приравниваем x_1 = 0. Находим x_2 = 3. Для нахождения второй точки приравниваем x_2 = 0. Находим x_1 = 1.5. Соединяем точку (0;3) с (1.5;0) прямой линией. Определим полуплоскость, задаваемую неравенством. Выбрав точку (0; 0), определим знак неравенства в полуплоскости:2 • 0 + 1 • 0 - 3 ≤ 0, т.е. 2x_1+x_2 - 3≥ 0 в полуплоскости *выше* прямой.

Построим уравнение x_1-3x_2 = 1 по двум точкам. Для нахождения первой точки приравниваем x_1 = 0. Находим x_2 = -0.33. Для нахождения второй точки приравниваем x_2 = 0. Находим x_1 = 1. Соединяем точку (0;-0.33) с (1;0) прямой линией. Определим полуплоскость, задаваемую неравенством. Выбрав точку (0; 0), определим знак неравенства в полуплоскости:1 • 0 - 3 • 0 - 1 ≤ 0, т.е. x_1-3x_2 - 1≤ 0 в полуплоскости *ниже* прямой.

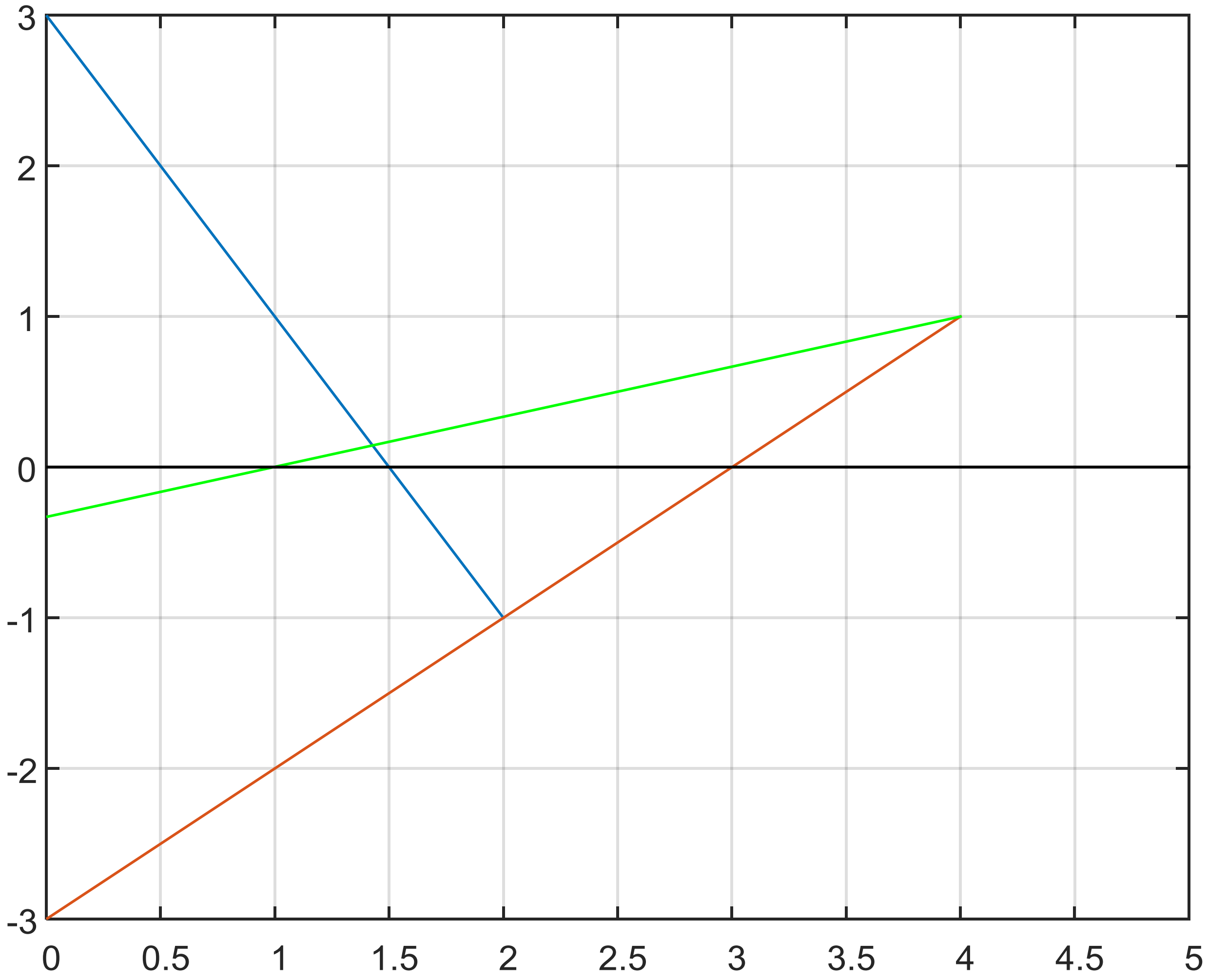

рисунок 2 - графический метод решения ЗЛП этап 1.

Шаг №2. Границы области допустимых решений. Пересечением полуплоскостей будет являться область, координаты точек которого удовлетворяют условию неравенствам системы ограничений задачи.

Шаг №3. Рассмотрим целевую функцию задачи $F = x_1+3x_2 \rightarrow \min$. Построим прямую, отвечающую значению функции $F = x_1+3x_2 = 0$. Вектор-градиент, составленный из коэффициентов целевой функции, указывает направление максимизации F(X). Начало вектора – точка (0; 0), конец – точка (1;3). Будем двигать эту прямую параллельным образом. Поскольку нас интересует минимальное решение, поэтому двигаем прямую до первого касания обозначенной области.

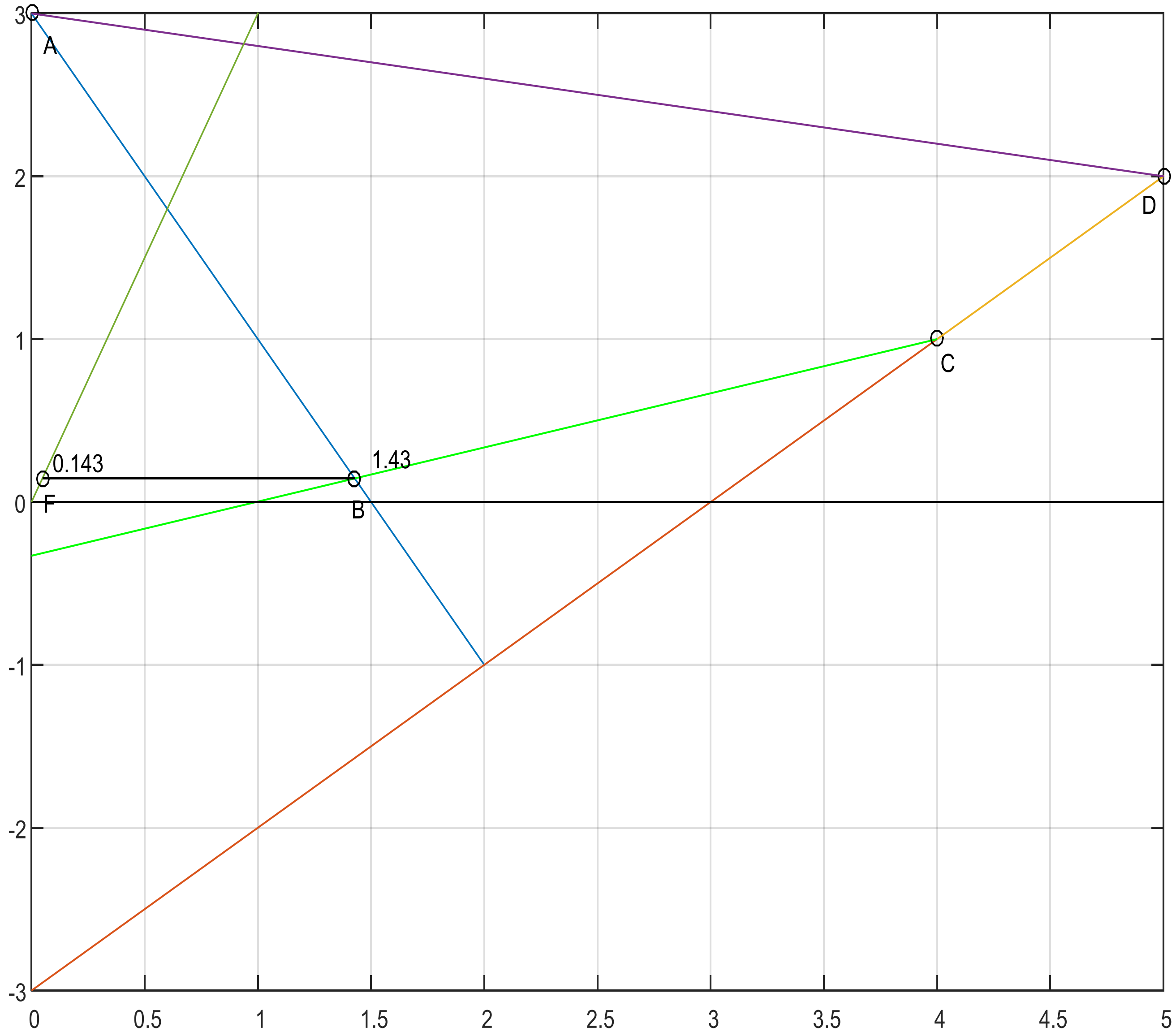

рисунок 3 - графический метод решения ЗЛП .

Прямая **F(x) = const** пересекает область в точке B. Так как точка B получена в результате пересечения прямых **(2)** и **(3)**, то ее координаты удовлетворяют уравнениям этих прямых:

$$2x_1+x_2=3$$

$$x_1-3x_2=1$$

Решив систему уравнений, получим: $x_1 = 1.4286$, $x_2 = 0.1429$

Откуда найдем минимальное значение целевой функции:

$$F(X) = 1*1.4286 + 3*0.1429 = 1.8571$$

Выводы

В ходе исследования были изучены и реализованы два метода решения задач линейного программирования.

Реализовав оба метода, можно сделать вывод о том, что симплекс-метод более трудоемок, в сравнении с графическим методом.

Трудозатраты на реализацию данного метода не оправданы, так как значение целевой функции совпадают, с графическим методом.

Но если увеличивать точность вычисления целевой функции, решаю эту задачу графическим методом, рано или поздно потерпим неудачу.

Список использованной литературы

1. Татарникова Т.М. Методы моделирования и оптимизации: Методические указания к выполнению лабораторных работ. СПб.: ГУАП, 2017
2. https://math.semestr.ru/lp/index.php
3. https://www.youtube.com/watch?v=A7z4hnduzLI
4. http://uchimatchast.ru/teory/tabl_simplex.php

ПРИЛОЖЕНИЕ

Листинг программ реализующих методы оптимизации реализованных в среде matlab и python

Графический метод

```
close all;
clear all;
clc;

plot([0 2],[3 -1]);
hold on
plot([0 4],[-3 1]);
plot([0 4],[-0.33 1],'green');
 plot([0 5],[0 0],'black');
 plot(4,1,'blacko');
 plot(0,3,'blacko');
 plot(1.4276,0.1447,'blacko');
 plot([4 5],[1 2]);
  plot(5,2,'blacko');
  plot([0 5],[3 2]);
  plot([0 1],[0 3]);
  plot([0.05 1.4276],[0.1447 0.1447],'*-');
 grid on
```

Симплекс метод

```
def transf(A,k,l,N,n):
    B=[[0 for x in range(n)] for y in range(N)]
    for i in range(N):
        for j in range(n):
            if (i==k[1]) and (j==l[1]):
                B[i][j]=1/A[i][j]
            elif (i==k[1]) and (j!=l[1]):
                B[i][j]=A[i][j]/A[k[1]][l[1]]
            elif (j==l[1]) and (i!=k[1]):
                B[i][j]=(-1)*A[i][j]/A[k[1]][l[1]]
            else:
                if (i<k[1]) and (j<l[1]):
                    B[i][j]=A[i][j]-(A[i][l[1]]*A[k[1]][j])/A[k[1]][l[1]]
                elif (i<k[1]) and (j>l[1]):
                    B[i][j]=A[i][j]-(A[k[1]][j]*A[i][l[1]])/A[k[1]][l[1]]
                elif (i>k[1]) and (j<l[1]):
                    B[i][j]=A[i][j]-(A[k[1]][j]*A[i][l[1]])/A[k[1]][l[1]]
                else:
                    B[i][j]=A[i][j]-(A[i][l[1]]*A[k[1]][j])/A[k[1]][l[1]]
    return B


def F_transf(A,N,n,x,y):
    while True:
        b=[0,0]
        l=[0,0]
        for i in range(n-1):
            if A[0][i]<l[0]:
                l[0]=A[0][i]
                l[1]=i
            if l[0]!=0:
                j=1
                while A[j][l[1]]<0:
                    j+=1
                if j>N:
                    print('func neogranichena')
                    break
                b[0]=A[j][n-1]/A[j][l[1]]
                b[1]=j
                for i in range(N-1):
                    if A[i+1][l[1]]>0:
                        if (A[i+1][n-1]/A[i+1][l[1]])<b[0]:
                            b[0]=(A[i+1][n-1]/A[i+1][l[1]])
                            b[1]=i+1
                A=transf(A,b,l,N,n)
                v=x[l[1]]
                x[l[1]]=y[b[1]-1]
                y[b[1]-1]=v
                b=[0,0]
                l=[0,0]
            else:
              for i in range(N-1):
                  print('x',y[i],'=',A[i+1][n-1],'\n')
              print('F =',A[0][n-1])
              return 0

def simpl(A):
```

```
    N=len(A)
    n=len(A[0])
    x=[(i+1) for i in range(n-1)]
    y=[(j+n) for j in range(N-1)]
    while True:
        k=[0,0]
        l=[0,0]
        b=[0,0]
        for i in range(N-1):
            if A[i+1][n-1]<k[0]:
                k[0]=A[i+1][n-1]
                k[1]=i+1
        if k[0]!=0:
            for i in range(n-1):
                if A[k[1]][i]<l[0]:
                    l[0]=A[k[1]][i]
                    l[1]=i
            if l[0]==0:
                print('Net resheniy')
                break;
            else:
                A=transf(A,k,l,N,n)
                v=x[l[1]]
                x[l[1]]=y[b[1]-1]
                y[b[1]-1]=v
        else:   ##k[0]==0 FFFFFF
            F_transf(A,N,n,x,y)
            break


A=[[-5,-4,0],
    [6,4,24],
    [1,2,6],
    [-1,1,1],
    [0,1,2],
    [-1,0,0],
    [0,-1,0]]
simpl(A)
a=0
```