Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Факультет информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"
профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

Курсовая работа по дисциплине
Алгоритмы и вычислительные методы оптимизации

Метод искусственного базиса

Вариант 8

Выполнил:

студент гр.ИП-814 Краснов И.В.

«14» апреля 2021 г. Подпись студента________________

Проверил:
ассистент кафедры ПМиК Новожилов Д. И.

«14» апреля 2021 г. Подпись преподавателя___________

Оценка__________________________

Новосибирск 2021 г.

Содержание

Задание

1. Перейти к канонической форме записи задачи линейного программирования.

$$Z(x_1,x_2)=p_1x_1+p_2x_2\to\min$$
$$\begin{cases} a_1x_1+a_2x_2\geq a \\ b_1x_1+b_2x_2\geq b \\ c_1x_1+c_2x_2\geq c \\ x_1;x_2\geq 0 \end{cases}$$

2. Написать программу, решающую задачу линейного программирования в канонической форме (с выводом всех промежуточных таблиц) методом искусственного базиса.

3. Решить исходную задачу графически и отметить на чертеже точки, соответствующие симплексным таблицам, полученным при выполнении программы из п.2.

4. Составить двойственную задачу к исходной и найти ее решение на основании теоремы равновесия.

Номер варианта	***a***	***b***	***c***	a_1	b_1	c_1	a_2	b_2	c_2	p_1	p_2	**Метод решения задачи**
8.	45	8	30	10	1	3	3	1	5	2	10	2

Рис. 1 – Исходные данные для варианта №8.

Переход к канонической форме записи

Для того, чтобы перейти к канонической форме записи исходной ЗЛП (задачи линейного программирования), необходимо добавить балансовые переменные соответствующего неравенству знака, а также сменить условие минимизации целевой функции на условие максимизации. Так как все знаки в неравенствах – это «≥», то необходимо в каждое неравенство добавить по одной уникальной балансовой переменной со знаком минус, тогда уравнение можно будет записать со знаком «=».

1) $Z = 2x_1 + 10x_2 \to min$

$$\begin{cases} 10x_1 + 3x_2 \geq 45 \\ x_1 + x_2 \geq 8 \\ 3x_1 + 5x_2 \geq 30 \\ x_1, x_2 \geq 0 \end{cases}$$

$Z' = -2x_1 - 10x_2 \to max$

$$\begin{cases} 10x_1 + 3x_2 - x_3 = 45 \\ x_1 + x_2 - x_4 = 8 \\ 3x_1 + 5x_2 - x_5 = 30 \\ x_i \geq 0, x = 1..5 \end{cases}$$

Рис. 2 – Каноническая форма записи исходной ЗЛП.

Решение задачи линейного программирования методом искусственного базиса

Далее будет описан алгоритм работы программы. Сначала пользователь вводит входные данные, т.е. саму ЗЛП в канонической форме записи.

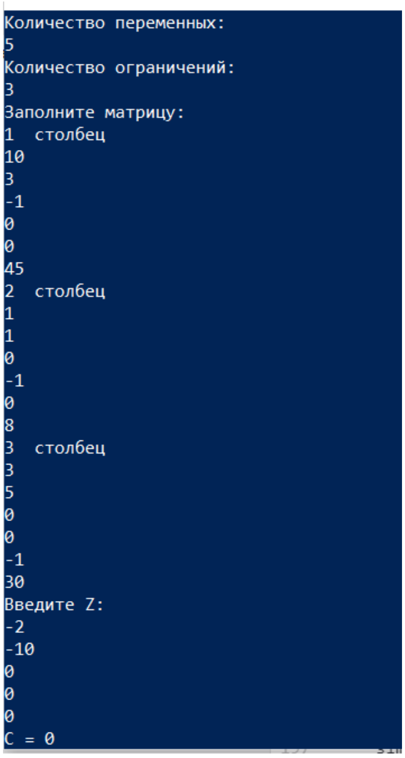


Рис. 3 – Входные данные.

После того, как данные введены, вызывается функция artificial_basis(), в котором выполняется сам метод искусственного базиса. Сначала происходит добавление нового базиса (в каждую строку добавляется новая переменная). После этого добавляются последовательно строки Z и M. Таким образом сформировалась симплекс-таблица, над которой будут производится дальнейшие вычисления.

После того, как сформировалась симплекс-таблица, начинается первый цикл, обрабатывающий строку M. В этом цикле из таблицы последовательно выводятся базисные переменные. В функции *check_m()* проверяется строка M, если в переменных есть отрицательное значение, цикл продолжается, если вся строка обнулилась, цикл завершается, иначе программа завершается со словами «Система не совместна». После завершения цикла M-строка удаляется.

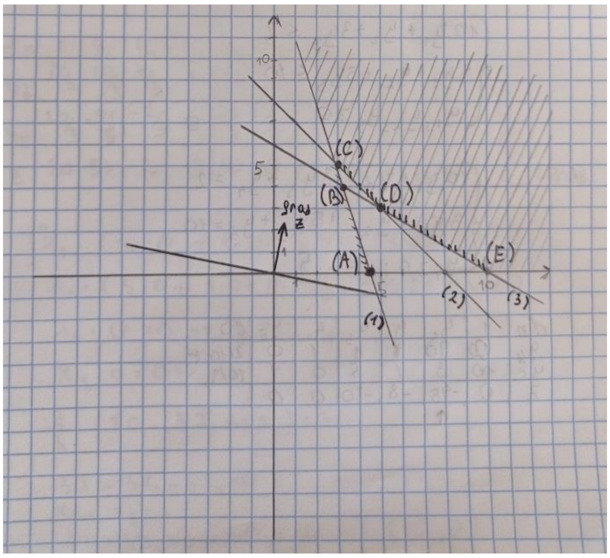
Затем следует цикл симплекс-метода, выход из цикла происходит, когда все отрицательные значения в строке Z перестали быть отрицательными (проверка происходит в функции check_z()).

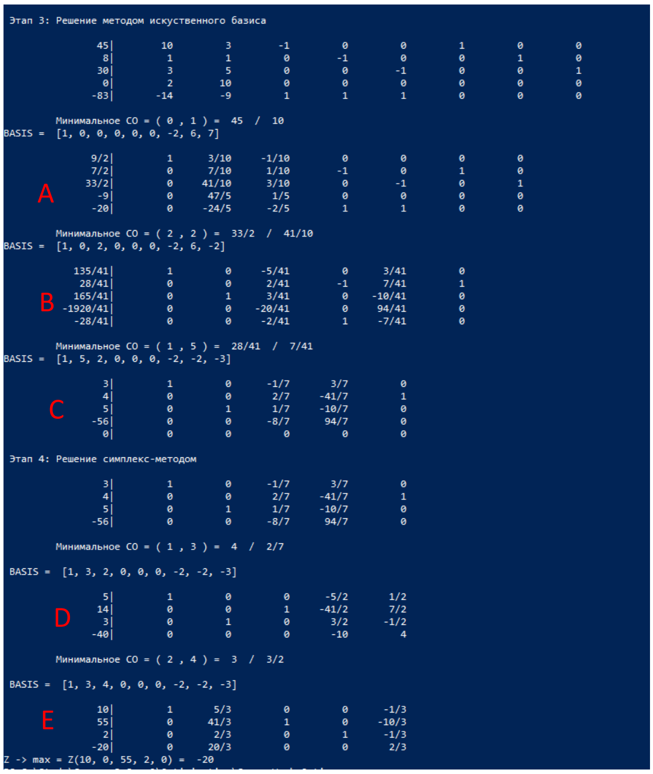
```
 Этап 3: Решение методом искуственного базиса

             45|       10        3       -1        0        0        1        0        0
              8|        1        1        0       -1        0        0        1        0
             30|        3        5        0        0       -1        0        0        1
              0|        2       10        0        0        0        0        0        0
            -83|      -14       -9        1        1        1        0        0        0

        Минимальное CO = ( 0 , 1 ) =  45  /  10
BASIS =  [1, 0, 0, 0, 0, 0, -2, 6, 7]

            9/2|        1     3/10    -1/10        0        0        0        0
            7/2|        0     7/10     1/10       -1        0        1        0
           33/2|        0    41/10     3/10        0       -1        0        1
             -9|        0     47/5      1/5        0        0        0        0
            -20|        0    -24/5     -2/5        1        1        0        0

        Минимальное CO = ( 2 , 2 ) =  33/2  /  41/10
BASIS =  [1, 0, 2, 0, 0, 0, -2, 6, -2]

         135/41|        1        0    -5/41        0     3/41        0
          28/41|        0        0     2/41       -1     7/41        1
         165/41|        0        1     3/41        0   -10/41        0
       -1920/41|        0        0   -20/41        0    94/41        0
         -28/41|        0        0    -2/41        1    -7/41        0

        Минимальное CO = ( 1 , 5 ) =  28/41  /  7/41
BASIS =  [1, 5, 2, 0, 0, 0, -2, -2, -3]

              3|        1        0     -1/7      3/7        0
              4|        0        0      2/7    -41/7        1
              5|        0        1      1/7    -10/7        0
            -56|        0        0     -8/7     94/7        0
              0|        0        0        0        0        0

 Этап 4: Решение симплекс-методом

              3|        1        0     -1/7      3/7        0
              4|        0        0      2/7    -41/7        1
              5|        0        1      1/7    -10/7        0
            -56|        0        0     -8/7     94/7        0

        Минимальное CO = ( 1 , 3 ) =  4  /  2/7

 BASIS =  [1, 3, 2, 0, 0, 0, -2, -2, -3]

              5|        1        0        0     -5/2      1/2
             14|        0        0        1    -41/2      7/2
              3|        0        1        0      3/2     -1/2
            -40|        0        0        0      -10        4

        Минимальное CO = ( 2 , 4 ) =  3  /  3/2

 BASIS =  [1, 3, 4, 0, 0, 0, -2, -2, -3]

             10|        1      5/3        0        0     -1/3
             55|        0     41/3        1        0    -10/3
              2|        0      2/3        0        1     -1/3
            -20|        0     20/3        0        0      2/3
Z -> max = Z(10, 0, 55, 2, 0) =  -20
```

Рис. 4 – Промежуточные таблицы работы алгоритма

Графическое решение исходной задачи

Для графического решения исходной ЗЛП мы построили линии, соответствующие уравнениям из исходной системы ограничений, определили, где находится область, которую мы ищем, и построили градиент, по которому, с помощью линий уровня, нашли точку минимума нашей функции. Это точка с координатами (8;0).





```
 Этап 3: Решение методом искуственного базиса

          45|       10        3       -1        0        0        1        0        0
           8|        1        1        0       -1        0        0        1        0
          30|        3        5        0        0       -1        0        0        1
           0|        2       10        0        0        0        0        0        0
         -83|      -14       -9        1        1        1        0        0        0

        Минимальное CO = ( 0 , 1 ) =  45  /  10
BASIS =  [1, 0, 0, 0, 0, 0, -2, 6, 7]

         9/2|        1     3/10    -1/10        0        0        0        0
         7/2|        0     7/10     1/10       -1        0        1        0
  A     33/2|        0    41/10     3/10        0       -1        0        1
          -9|        0     47/5      1/5        0        0        0        0
         -20|        0    -24/5     -2/5        1        1        0        0

        Минимальное CO = ( 2 , 2 ) =  33/2  /  41/10
BASIS =  [1, 0, 2, 0, 0, 0, -2, 6, -2]

      135/41|        1        0    -5/41        0     3/41        0
       28/41|        0        0     2/41       -1     7/41        1
  B   165/41|        0        1     3/41        0   -10/41        0
    -1920/41|        0        0   -20/41        0    94/41        0
      -28/41|        0        0    -2/41        1    -7/41        0

        Минимальное CO = ( 1 , 5 ) =  28/41  /  7/41
BASIS =  [1, 5, 2, 0, 0, 0, -2, -2, -3]

           3|        1        0     -1/7      3/7        0
           4|        0        0      2/7    -41/7        1
  C        5|        0        1      1/7    -10/7        0
         -56|        0        0     -8/7     94/7        0
           0|        0        0        0        0        0

 Этап 4: Решение симплекс-методом

           3|        1        0     -1/7      3/7        0
           4|        0        0      2/7    -41/7        1
           5|        0        1      1/7    -10/7        0
         -56|        0        0     -8/7     94/7        0

        Минимальное CO = ( 1 , 3 ) =  4  /  2/7

 BASIS =  [1, 3, 2, 0, 0, 0, -2, -2, -3]

           5|        1        0        0     -5/2      1/2
          14|        0        0        1    -41/2      7/2
  D        3|        0        1        0      3/2     -1/2
         -40|        0        0        0      -10        4

        Минимальное CO = ( 2 , 4 ) =  3  /  3/2

 BASIS =  [1, 3, 4, 0, 0, 0, -2, -2, -3]

          10|        1      5/3        0        0     -1/3
  E       55|        0     41/3        1        0    -10/3
           2|        0      2/3        0        1     -1/3
         -20|        0     20/3        0        0      2/3
Z -> max = Z(10, 0, 55, 2, 0) =  -20
```

Рис. 5 – Графическое решение исходной ЗЛП.

Решение двойственной задачи

4) $Z = 2x_1 + 10x_2 \to min$

$$\begin{matrix} y_1 \\ y_2 \\ y_3 \\ \\ \end{matrix} \begin{cases} 10x_1 + 3x_2 \geq 45 \\ x_1 + x_2 \geq 8 \\ 3x_1 + 5x_2 \geq 30 \\ x_1, x_2 \geq 0 \end{cases}$$

$$W = 45y_1 + 8y_2 + 30y_3 \to max$$

$$\begin{cases} 10y_1 + y_2 + 3y_3 \leq 2 \\ 3y_1 + y_2 + 5y_3 \leq 10 \\ y_1, y_2, y_3 \geq 0 \end{cases}$$

$$\begin{cases} 10y_1 + y_2 + 3y_3 + y_4 = 2 \\ 3y_1 + y_2 + 5y_3 + y_5 = 10 \\ y_i \geq 0; \ i = 1..5 \end{cases}$$

бп	1	y_1	y_2	y_3	y_4	y_5	CO
y_4	2	10	1	3	1	0	2/10 ←
y_5	10	3	1	5	0	1	10/3
Z	0	-45 ↑	-8	-30	0	0	

Рис. 6 – Составление двойственной задачи и симплексной таблицы.

б.п	1	y_1	y_2	y_3	y_4	y_5	CO
y_1	2/10	1	1/10	3/10	1/10	0	$\frac{2}{3}$
y_5	$\frac{94}{10}$	0	$\frac{7}{10}$	$\frac{41}{10}$	$-\frac{3}{10}$	1	$\frac{94}{41}$
z	9	0	$-\frac{35}{10}$	$-\frac{265}{10}$	$\frac{45}{10}$	0	

б.п	y_1	y_1	y_2	y_3	y_4	y_5
y_1		1		0		
y_3	$\frac{94}{41}$	0	$\frac{7}{41}$	1	$-\frac{3}{41}$	$\frac{10}{41}$
z						

б.п	1	y_1	y_2	y_3	y_4	y_5	CO
y_3	2/3	10/3	1/3	1	1/3	0	
y_5	20/3	$-\frac{41}{3}$	$-2/3$	0	$-\frac{5}{3}$	1	
z	20	55	2	0	10	0	

$(0; 0; 2/3; 0; 20/3)$

$$\begin{cases} x_1 \cdot (10 \cdot 0 + 1 \cdot 0 + 3 \cdot 2/3 + 1 \cdot 0 - 2) = 0 \\ x_2 \cdot (3 \cdot 0 + 1 \cdot 0 + 5 \cdot 2/3 + 20/3 - 10) = 0 \end{cases}$$

$x_1 \cdot 0 = 0 \Rightarrow \cancel{x \neq 0}$ $\underline{x_1}$

$x_2 \cdot 0 = 0 \Rightarrow \underline{x_2}$

$\frac{2}{3} \cdot (30 - (3x_1 + 5x_2)) = 0$

$30 - 3x_1 - 5x_2 = 0$

при $x_2 = 0$

$30 - 3x_1 = 0 \Rightarrow x_1 = 10$ $z(10; 0) = z \to \min = 20$

Рис. 7 –Решение двойственной задачи на основании теоремы равновесия.

Код программы

SimpleFrac.py – файл с описанием класса дроби

```
class SimpleFrac:
    a = 0
    b = 1

    def __init__(self):
        self.a = 0
        self.b = 1

    def write(self, x: int, y: int):
        self.a = x
        self.b = y

    def __str__(self):
        if self.b != 1:
            return '{}/{}'.format(int(self.a), int(self.b))
        else:
            return '{}'.format(int(self.a))

    def sf_reduce(self):
        temp_a = abs(self.a)
        temp_b = abs(self.b)

        if self.a == 0:
            return self

        while temp_a != temp_b:
            if temp_a > temp_b:
                tmp = temp_a
                temp_a = temp_b
                temp_b = tmp
            temp_b = temp_b - temp_a

        self.a /= temp_a
        self.b /= temp_a
```

```
        return self

    def __add__(self, other):
        result = SimpleFrac()
        result.a = self.a * other.b + other.a * self.b
        result.b = self.b * other.b

        if result.b < 0:
            result.a *= -1
            result.b *= -1

        if result.a == 0:
            result.b = 1

        result.sf_reduce()

        return result

    def __sub__(self, other):
        result = SimpleFrac()
        result.a = self.a * other.b - other.a * self.b
        result.b = self.b * other.b
        if result.b < 0:
            result.a *= -1
            result.b *= -1

        if result.a == 0:
            result.b = 1

        result.sf_reduce()

        return result

    def __mul__(self, other):
        result = SimpleFrac()
        temp1 = SimpleFrac()
```

```
        temp2 = SimpleFrac()

        temp1.a = self.a
        temp1.b = other.b
        temp2.a = other.a
        temp2.b = self.b
        temp1.sf_reduce()
        temp2.sf_reduce()
        result.a = temp1.a * temp2.a
        result.b = temp1.b * temp2.b

        if result.b < 0:
            result.a *= -1
            result.b *= -1

        if result.a == 0:
            result.b = 1

        result.sf_reduce()

        return result

    def __truediv__(self, other):
        result = SimpleFrac()
        temp1 = SimpleFrac()
        temp2 = SimpleFrac()

        temp1.a = self.a
        temp1.b = other.a
        temp2.a = other.b
        temp2.b = self.b
        temp1.sf_reduce()
        temp2.sf_reduce()
        result.a = temp1.a * temp2.a
        result.b = temp1.b * temp2.b

        if result.b < 0:
```

```
            result.a *= -1
            result.b *= -1

        if result.a == 0:
            result.b = 1

        result.sf_reduce()

        return result

    def __gt__(self, other):
        result = self - other
        if result.a > 0:
            return True
        else:
            return False

    def __ge__(self, other):
        result = self - other
        if result.a >= 0:
            return True
        else:
            return False

    def __lt__(self, other):
        result = self - other
        if result.a < 0:
            return True
        else:
            return False

    def __le__(self, other):
        result = self - other
        if result.a <= 0:
            return True
        else:
            return False
```

```
    def __abs__(self):
        if self.a < 0:
            self.a *= -1
        return self
```

main.py – главный исполнительный файл

```
from SimpleFrac import SimpleFrac
import math


def print_matrix(matrix, n, m, k):

    print()
    if k == "matrix":
        for i in range(m):
            for j in range(n + 1):
                if j == n:
                    print("|%10s" % matrix[i][j], end='')
                else:
                    print("%10s" % matrix[i][j], end='')
            print()
    elif k == "simplex":
        for i in range(m + 2):
            print("\t", end='')
            for j in range(n + 1):
                if j == 0:
                    print("%10s|" % matrix[i][j], end='')
                else:
                    print("%10s" % matrix[i][j], end='')
            print()
    else:
        print("\nERROR: UNKNOWN SITUATION\n")



def check_m(simplex_list, n):
```

```
    k = 0

    for i in range(1, n + 1):
        if simplex_list[i].a > 0:
            k = 1

        elif simplex_list[i].a < 0:
            return 0

    if k == 0 and simplex_list[0].a == 0:
        return 1
    else:
        return 2


def check_z(simplex_list, n):

    k = 0

    for i in range(1, n + 1):
        if simplex_list[i].a < 0:
            return 0

    return 1


def artificial_basis(matrix, n, m, z):

    N = n
    simplex = []
    basis = []

    for i in range(n + m + 1):
        basis.append(i)
    for i in range(n + 1):
        basis[i] = 0
```

```
print("\n", "Этап 1: Расширение матрицы")

for i in range(m):
    k = 0
    for j in range(m):
        matrix[i].insert(N + k, SimpleFrac())
        if j == i:
            matrix[i][N + k].write(1, 1)
        k += 1

N += m
print_matrix(matrix, N, m, "matrix")

print("\n", "Этап 2: Построение Симплекс-таблицы")

for i in range(m):
    simplex.append([])
    for j in range(N + 1):
        if j == 0:
            simplex[i].append(matrix[i][N])
        else:
            simplex[i].append(matrix[i][j - 1])

simplex.append([])
for j in range(n + 1):
    if j == 0:
        simplex[m].append(z[n])
    else:
        temp = z[j - 1]
        temp.a *= -1
        simplex[m].append(temp)

for j in range(n + 1, N + 1):
    simplex[m].append(SimpleFrac())

simplex.append([])
for j in range(n + 1):
```

```
        if j == 0:
            temp = SimpleFrac()
            for i in range(m):
                temp = temp + matrix[i][N]
            temp.a *= -1
            simplex[m + 1].append(temp)
        else:
            temp = SimpleFrac()
            for i in range(m):
                temp = temp + matrix[i][j - 1]
            temp.a *= -1
            simplex[m + 1].append(temp)

    for j in range(n + 1, N + 1):
        simplex[m + 1].append(SimpleFrac())

    print_matrix(simplex, N, m, "simplex")

    print("\n", "Этап 3: Решение методом искуственного базиса")

    print_matrix(simplex, N, m, "simplex")
    # print("\n\t", "BASIS = ", basis)
    while True:

        stop_kof = check_m(simplex[m + 1], n)           # проверка строки M

        if stop_kof == 0:       # если есть отрицательный элемент в строке M
            maxi = SimpleFrac()
            maxi_i = -1
            for j in range(1, N + 1):               # поиск столбца с 
максимальным по модулю значением в M
                if int(simplex[m + 1][j].a) < 0 and simplex[m + 1][j] < maxi:
                    maxi = simplex[m + 1][j]
                    maxi_i = j

            mini = SimpleFrac()
            mini_i = -1
```

```
            k = -1

            for i in range(m):              # поиск опорного элемента, чьё
симплексное отношение будет взято как минимальное

                if simplex[i][maxi_i].a > 0 and simplex[i][0].a > 0:

                    mini = simplex[i][0] / simplex[i][maxi_i]

                    k = i

                    break


            if k == -1:

                print("Нет решений")

                return 1


            for i in range(k, m):           # поиск строки с минимальным
сиплексным отношением

                if simplex[i][maxi_i].a > 0 and simplex[i][0].a > 0 and
simplex[i][0] / simplex[i][maxi_i] <= mini:

                    mini = simplex[i][0] / simplex[i][maxi_i]

                    mini_i = i


            print("\n\t", "Минимальное СО = (", mini_i, ",", maxi_i, ") = ",
simplex[mini_i][0], " / ", simplex[mini_i][maxi_i])


            mini = simplex[mini_i][maxi_i]

            for i in range(N + 1):          # делим строку с минимальным
сиплексным отношением на опорный элемент

                simplex[mini_i][i] = simplex[mini_i][i] / mini


            for i in range(m + 2):          # шаг Гаусса

                for j in range(N + 1):

                    if i != mini_i and j != maxi_i:

                        temp_1 = simplex[i][j] * simplex[mini_i][maxi_i]

                        temp_2 = simplex[i][maxi_i] * simplex[mini_i][j]

                        simplex[i][j] = temp_1 - temp_2


            for i in range(m + 2):          # обнуляем столбец с опорным
элементом

                if i != mini_i:

                    simplex[i][maxi_i].a = 0

                    simplex[i][maxi_i].b = 1
```

```
            if basis[n + 1 + mini_i] >= 0:      # удаление столбца с искусственной переменной и обновление базиса
                # print("\n\t", "DEL = ", basis[n + 1 + mini_i])
                temp_k = basis[n + 1 + mini_i]
                for i in range(m + 2):
                    simplex[i].pop(temp_k)
                N -= 1
                basis[n + 1 + mini_i] = -1
                for i in range(n + 1 + mini_i, n + m + 1):
                    basis[i] -= 1
            basis[mini_i] = maxi_i
            print("BASIS = ", basis)

            print_matrix(simplex, N, m, "simplex")

        elif stop_kof == 1:         # если строка M обнулилась
            break

        elif stop_kof == 2:                 # если матрица не разрешима
            print("\n", "Система ограничений не совместна")
            return 1

    print("\n", "Этап 4: Решение симплекс-методом")

    simplex.pop(m + 1)
    print_matrix(simplex, N, m - 1, "simplex")
    while True:
        kof = check_z(simplex[m], N)        # проверяем строку z

        if kof == 0:        # если в строке есть отрицательный элемент
            maxi = SimpleFrac()
            maxi_i = -1
            for j in range(1, N + 1):  # поиск столбца с максимальным по модулю значением в Z
                if int(simplex[m][j].a) < 0 and simplex[m][j] < maxi:
                    maxi = simplex[m][j]
                    maxi_i = j
```

```
            mini = SimpleFrac()
            mini_i = -1
            k = -1
            for i in range(m):  # поиск опорного элемента, чьё симплексное
отношение будет взято как минимальное
                if simplex[i][maxi_i].a > 0 and simplex[i][0].a > 0:
                    mini = simplex[i][0] / simplex[i][maxi_i]
                    k = i
                    break

            if k == -1:
                print("Начинайте паниковать")
                print("Но сперва проверьте правильно ли записана строка z")
                return 1
            # print("Cur mini = ", mini)
            for i in range(k, m):  # поиск строки с минимальным сиплексным
отношением
                if simplex[i][maxi_i].a > 0 and simplex[i][0].a > 0 and
simplex[i][0] / simplex[i][maxi_i] <= mini:
                    mini = simplex[i][0] / simplex[i][maxi_i]
                    mini_i = i

            print("\n\t", "Минимальное СО = (", mini_i, ",", maxi_i, ") = ",
simplex[mini_i][0], " / ",
                  simplex[mini_i][maxi_i])

            mini = simplex[mini_i][maxi_i]
            for i in range(N + 1):  # делим строку с минимальным сиплексным
отношением на опорный элемент
                simplex[mini_i][i] = simplex[mini_i][i] / mini

            for i in range(m + 1):  # шаг Гаусса
                for j in range(N + 1):
                    if i != mini_i and j != maxi_i:
                        temp_1 = simplex[i][j] * simplex[mini_i][maxi_i]
                        temp_2 = simplex[i][maxi_i] * simplex[mini_i][j]
                        simplex[i][j] = temp_1 - temp_2

            for i in range(m + 1):  # обнуляем столбец с опорным элементом
```

```
                if i != mini_i:
                    simplex[i][maxi_i].a = 0
                    simplex[i][maxi_i].b = 1

            basis[mini_i] = maxi_i              # записываем новый элемент в базис
            print("\n", "BASIS = ", basis)

            print_matrix(simplex, N, m - 1, "simplex")

        if kof == 1:
            print("Z -> max = Z(", end='')

            for i in range(n):
                k = -1
                for j in range(n):
                    if i + 1 == basis[j] and j <= m:
                        print(simplex[j][0], end='');
                        k = 0
                        break;
                if k != 0:
                    print(0, end='')
                if i != n - 1:
                    print(", ", end='')
            print(") = ", simplex[m][0])
            return 0

    return 1


if __name__ == '__main__':

    matrix = []
    z = []

    print("Количество переменных: ")
    n = int(input())
```

```
print("Количество ограничений: ")
m = int(input())

for i in range(m):
    matrix.append([])
    for j in range(n + 1):
        matrix[i].append(SimpleFrac())

print("Заполните матрицу: ")
for i in range(m):
    print(i + 1, " столбец")
    for j in range(n + 1):
        matrix[i][j].write(int(input()), 1)

print('Введите Z: ')
for i in range(n + 1):
    if i == n:
        print("C = ", end='')
    z.append(SimpleFrac())
    z[i].write(int(input()), 1)

print("\n", "Ваша матрица:")
print_matrix(matrix, n, m, "matrix")

artificial_basis(matrix, n, m, z)
```

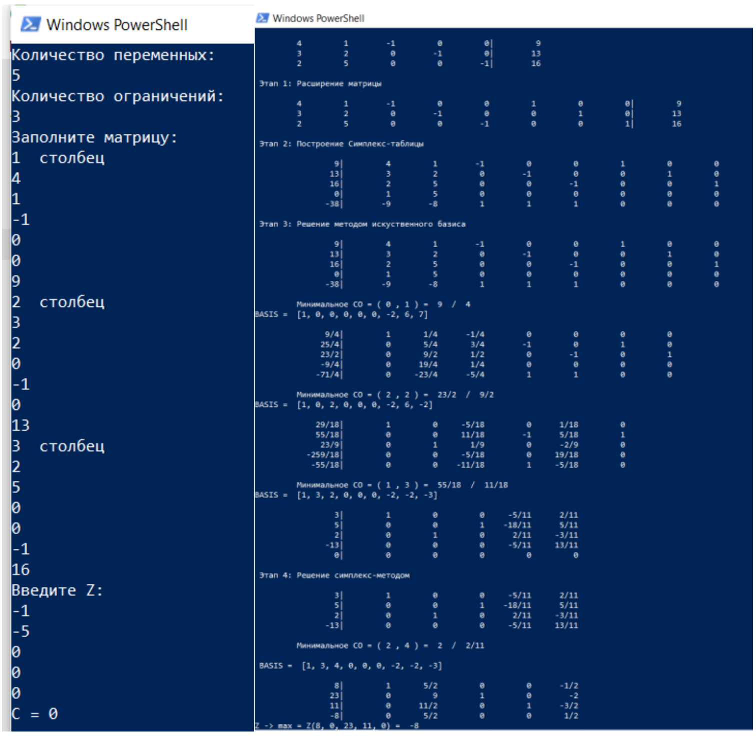


Рис. 8,9 Решение задачи из варианта 2

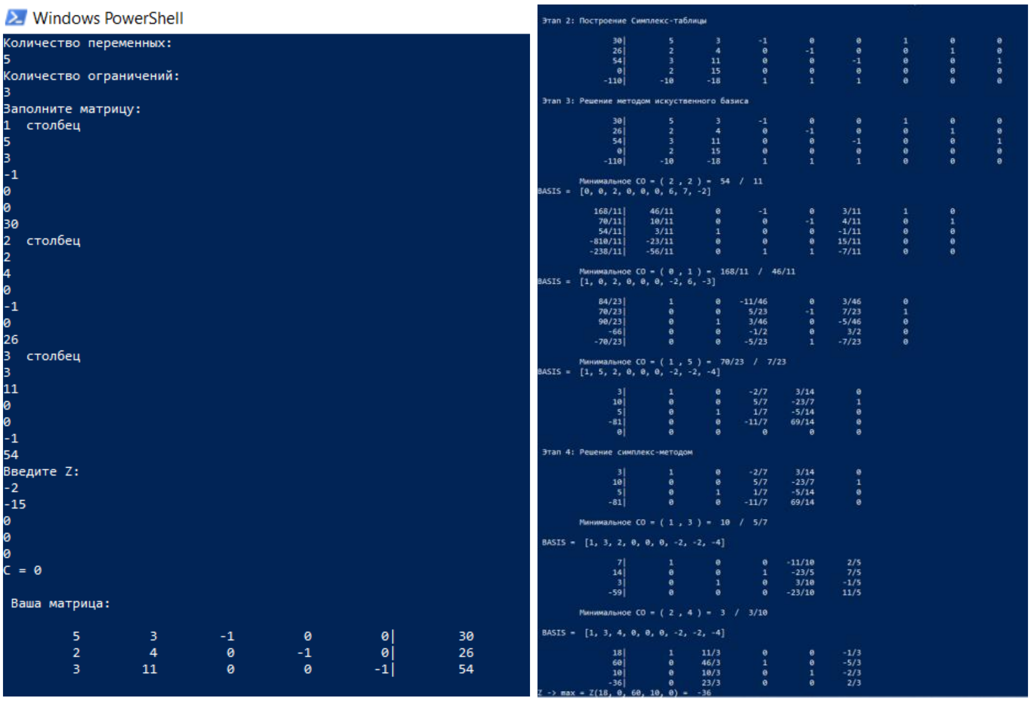
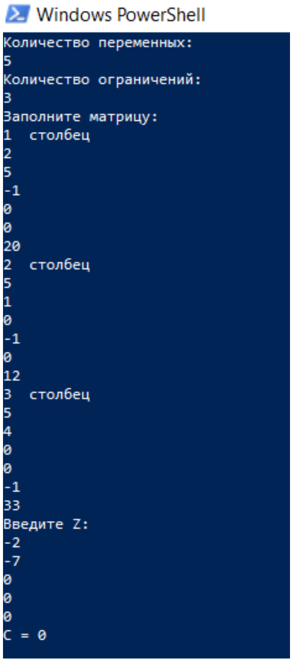


Рис. 10,11 Решение задачи из варианта 5



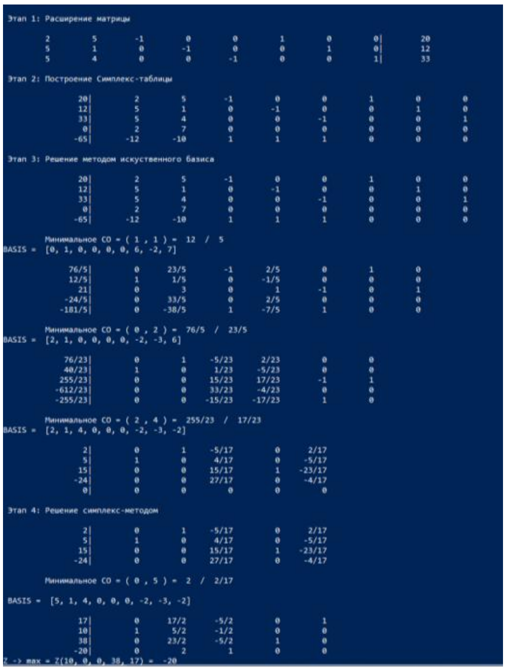


Рис. 12, 13 Решение задачи из варианта 11