МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Решение системы линейных уравнений итерационным методом и методом Гаусса-Зейделя

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к лабораторной работе №2

по дисциплине

Вычислительная Математика

РУКОВОДИТЕЛЬ:

Суркова Анна Сергеевна

_____

(подпись)

СТУДЕНТ:

Цветков Николай Максимович

_____

(подпись)

19-ИВТ-3

Работа защищена «___» _____

С оценкой _____

Нижний Новгород 2020

Оглавление

Цель

Закрепление знаний и умений по нахождению решений систем линейных уравнений различными способами.

Постановка задачи

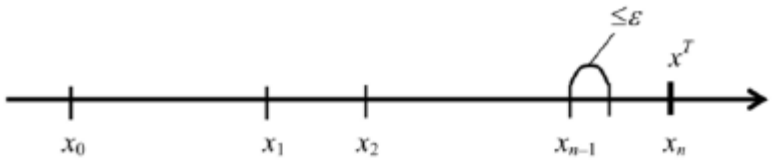
Решить систему линейных уравнений методом Гаусса, итерационным методом и методом Гаусса-Зейделя. При необходимости преобразовать систему к диагонально преобладающему виду. Сделать оценку количества итераций для итерационных методов, сравнить. Задание по вариантам. Номер варианта – номер студента в списке группы. ε=0.001

Вариант №7:

$$\begin{cases} 3.11x_1 - 1.66x_2 - 0.60x_3 = -0.92 \\ -1.65x_1 + 3.51x_2 - 0.78x_3 = 2.57 \\ 0.60x_1 + 0.78x_2 - 1.87x_3 = 1.65 \end{cases}$$

Теоретические сведения

Процесс нахождения оптимального решения чаще всего имеет итерационный характер, т.е. последовательность $\{x0,\ x1,\ \ldots,xn \rightarrow x\}$ стремится к точному решению при увеличении кол-ва итераций n.



Весьма важным элементом всех итерационных методов является критерий (правило) остановки итерационного процесса. Именно критерий определяет точность достижения решения, а соответственно и эффективность метода.

Наиболее распространённые критерии остановки при решении системы линейных уравнения:

1) $\left|\left|x^{(k)} - x^{(k-1)}\right|\right| = \sqrt{\sum_{i=1}^{n}\left(x_i^{(k)} - x_i^{(k-1)}\right)^2} < \varepsilon$ - расстояние между последовательными приближениями меньше ε

2) $\max\limits_{1 \le j \le n} \left|x_i^{(k)} - x_i^{(k-1)}\right| < \varepsilon$ - максимум из поэлементных разностей двух последовательный приближений меньше ε

Метод Гаусса

Это метод последовательного исключения переменных, когда с помощью элементарных преобразований система уравнений приводится к равносильной системе треугольного вида, из которой последовательно, начиная с последних (по номеру), находятся все переменные системы.

1. На первом этапе осуществляется так называемый прямой ход, когда путём элементарных преобразований над строками систему приводят к ступенчатой или треугольной форме, либо устанавливают, что система несовместна. Для этого среди элементов первого столбца матрицы выбирают ненулевой, перемещают содержащую его строку в крайнее верхнее положение, делая эту строку первой. Далее ненулевые элементы первого столбца всех нижележащих строк обнуляются путём вычитания из каждой строки первой строки, домноженной на отношение первого элемента этих строк к первому элементу первой строки. После того, как указанные преобразования были совершены, первую строку и первый столбец мысленно вычёркивают и продолжают, пока не останется матрица нулевого размера. Если на какой-то из итераций среди элементов первого столбца не нашёлся ненулевой, то переходят к следующему столбцу и проделывают аналогичную операцию.

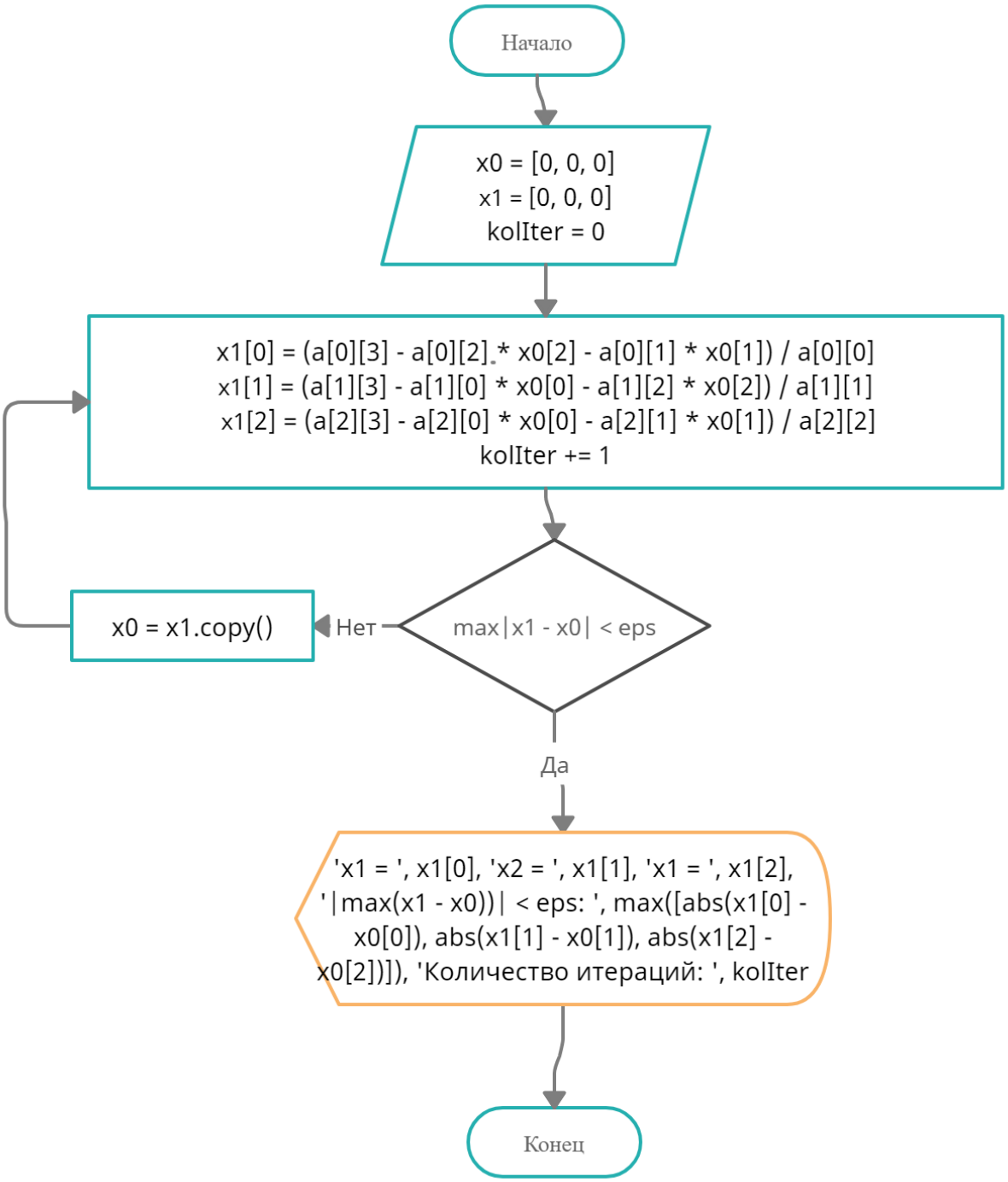
2. На втором этапе осуществляется так называемый обратный ход, суть которого заключается в том, чтобы выразить все получившиеся базисные переменные через небазисные и построить фундаментальную систему решений, либо, если все переменные являются базисными, то выразить в численном виде единственное решение системы линейных уравнений. Эта процедура начинается с последнего уравнения, из которого выражают соответствующую базисную переменную (а она там всего одна) и подставляют в предыдущие уравнения, и так далее, поднимаясь по «ступенькам» наверх. Каждой строчке соответствует ровно одна базисная переменная, поэтому на каждом шаге, кроме последнего (самого верхнего), ситуация в точности повторяет случай последней строки.

Метод простой итерации (Метод Якоби)

Метод Якоби — разновидность метода простой итерации для решения системы линейных алгебраических уравнений, которые обладают свойством строгого диагонального преобладания.

Для того, чтобы построить итеративную процедуру метода Якоби, необходимо провести предварительное преобразование системы уравнений $A\vec{x} = \vec{b}$ к итерационному виду $\vec{x} = B\vec{x} + \vec{g}$. Оно может быть осуществлено по следующему правилу:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{i \neq j} a_{ij} x_j^{(k)}\right), \qquad i = 1, 2, \dots, n.$$

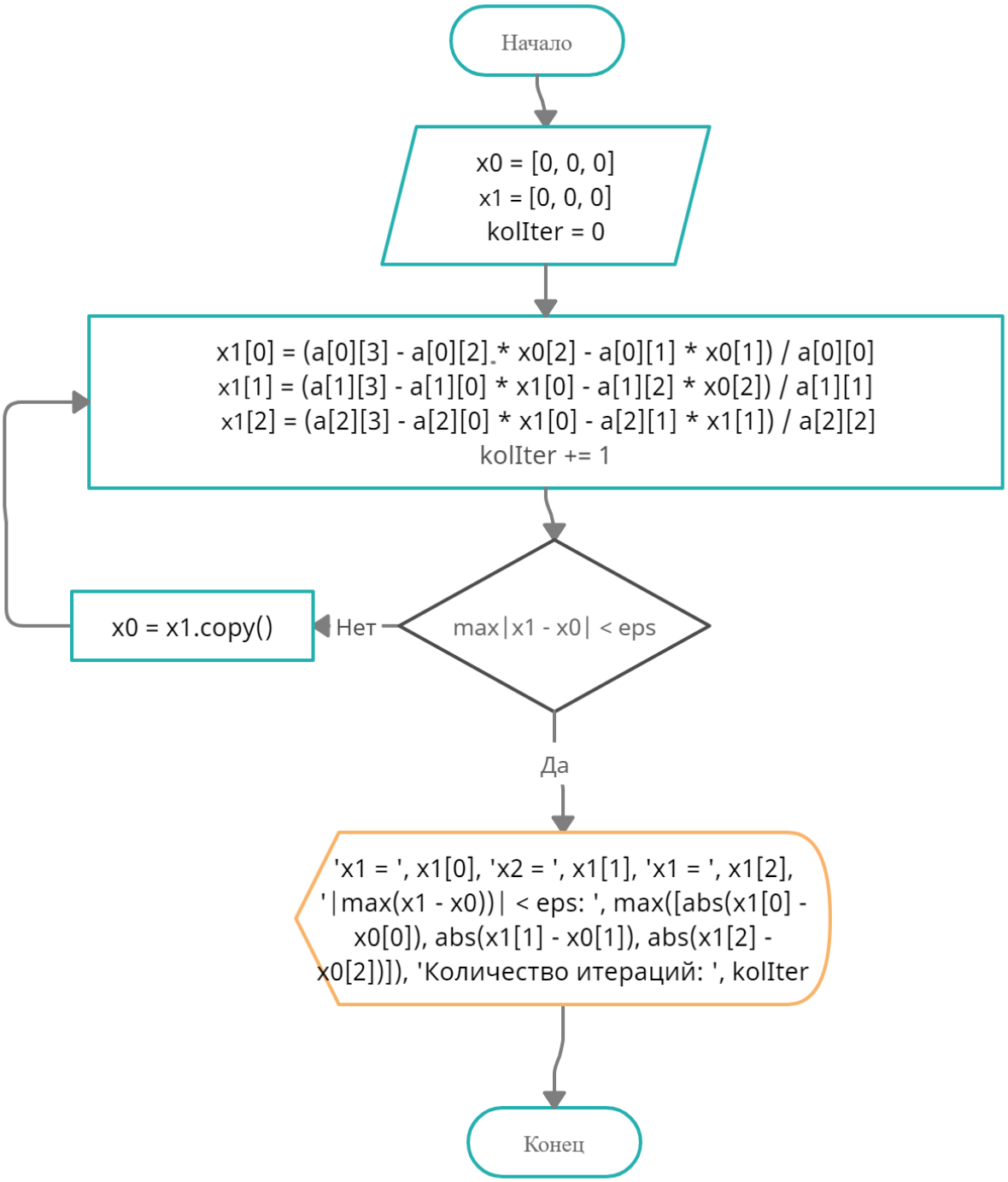

Начало
x0 = [0, 0, 0]
x1 = [0, 0, 0]
kolIter = 0
x1[0] = (a[0][3] - a[0][2] * x0[2] - a[0][1] * x0[1]) / a[0][0]
x1[1] = (a[1][3] - a[1][0] * x0[0] - a[1][2] * x0[2]) / a[1][1]
x1[2] = (a[2][3] - a[2][0] * x0[0] - a[2][1] * x0[1]) / a[2][2]
kolIter += 1
max|x1 - x0| < eps
Нет
x0 = x1.copy()
Да
'x1 = ', x1[0], 'x2 = ', x1[1], 'x1 = ', x1[2], '|max(x1 - x0))| < eps: ', max([abs(x1[0] - x0[0]), abs(x1[1] - x0[1]), abs(x1[2] - x0[2])]), 'Количество итераций: ', kolIter
Конец

Метод Гаусса - Зейделя

Метод Гаусса - Зейделя можно рассматривать как модификацию метода Якоби. Основная идея модификации состоит в том, что новые значения $\vec{xi}$ используются здесь сразу же по мере получения, в то время как в методе Якоби они не используются до следующей итерации.

Для того, чтобы построить итеративную процедуру метода Гаусса - Зейделя, необходимо провести предварительное преобразование системы уравнений $\vec{Ax}= \vec{b}$ к итерационному виду $\vec{x}=\vec{Bx}+ \vec{g}$. Оно может быть осуществлено по следующему правилу:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} \right), \qquad i = 1, 2, \dots, n.$$


Начало
x0 = [0, 0, 0]
x1 = [0, 0, 0]
kolIter = 0
x1[0] = (a[0][3] - a[0][2] * x0[2] - a[0][1] * x0[1]) / a[0][0]
x1[1] = (a[1][3] - a[1][0] * x1[0] - a[1][2] * x0[2]) / a[1][1]
x1[2] = (a[2][3] - a[2][0] * x1[0] - a[2][1] * x1[1]) / a[2][2]
kolIter += 1
max|x1 - x0| < eps
Нет
x0 = x1.copy()
Да
'x1 = ', x1[0], 'x2 = ', x1[1], 'x1 = ', x1[2], '|max(x1 - x0))| < eps: ', max([abs(x1[0] - x0[0]), abs(x1[1] - x0[1]), abs(x1[2] - x0[2])]), 'Количество итераций: ', kolIter
Конец

Расчетные данные

Исходная система:

$$\begin{cases} 3.11x_1 - 1.66x_2 - 0.60x_3 = -0.92 \\ -1.65x_1 + 3.51x_2 - 0.78x_3 = 2.57 \\ 0.60x_1 + 0.78x_2 - 1.87x_3 = 1.65 \end{cases}$$

Метод простой итерации:

N	x_1	x_2	x_3	e_1	e_2	e_3
0	0	0	0			
1	-0.296	0.732	-0.882	0.296	0.732	0.882
2	-0.0752	0.397	-0.672	-0.221	-0.335	-0.21
3	-0.214	0.548	-0.741	0.138	0.15	0.069
4	-0.147	0.467	-0.722	-0.067	-0.0803	-0.0184
5	-0.186	0.503	-0.734	0.0393	0.0356	0.012
6	-0.169	0.482	-0.732	-0.0167	-0.0212	-0.00222
7	-0.18	0.49	-0.736	0.0109	0.00833	0.00347
8	-0.176	0.484	-0.736	-0.00378	-0.00588	1.0E-5
9	-0.179	0.486	-0.737	0.00314	0.00177	0.00124
10	-0.179	0.484	-0.737	-0.000707	-0.00175	0.000268
11	-0.18	0.484	-0.738	0.000987	0.000273	0.000504

Метод Гаусса – Зейделя:

N	x_1	x_2	x_3	e_1	e_2	e_3
0	0	0	0			
1	-0.296	0.593	-0.73	0.296	0.593	0.73
2	-0.12	0.514	-0.707	-0.176	-0.0796	-0.0232
3	-0.158	0.501	-0.724	0.038	-0.0127	0.0175
4	-0.168	0.492	-0.731	0.0102	-0.00866	0.00687

5	-0.174	0.488	-0.735	0.00595	-0.00432	0.00371
6	-0.177	0.486	-0.737	0.00302	-0.00225	0.00191
7	-0.179	0.484	-0.738	0.00157	-0.00116	0.000986
8	-0.18	0.484	-0.738	0.000809	-0.0006	0.00051

Метод	**Полученный вектор X**
Метод Гаусса	{-0.18, 0.48, -0.74}
Метод простой итераций	{-0.179667956, 0.4843626, -0.737764}
Метод Гаусса - Зейделя	{-0.1795205, 0.483891, -0.7381161}

Листинг разработанной программы

Main.py:

```
import SolvingLinearEquations as sle

sle.GaussMethod()
sle.SimpleIteration()
sle.GaussSeidel()
```

SolvingLinearEquations.py:

```
import numpy as np



eps = 0.001
def GaussMethod():
    print('Метод Гаусса:')
    a = np.array([
        [3.11, -1.66, -0.60, -0.92],
        [-1.65, 3.51, -0.78, 2.57],
        [0.60, 0.78, -1.87, 1.65]
    ])
    a[1] = a[1] - np.round(a[0] * (-
0.53), 2) # из второй строчки вычитаем первую, умноженную на -0.53
    print(a)
    a[2] = a[2] - np.round(a[0] * 0.1929, 2) # из третей строчки вычитаем первую,
 умноженную на 0.1929
    print(a)
    a[2] = a[2] - np.round(a[1] * 0.4183, 2) # из третей строчки вычитаем вторую,
 умноженную на 0.4183
    print(a)
    x3 = round(a[2][3] / a[2][2], 2) # выводим x3
    x2 = round(((a[1][3] - x3 * a[1][2]) / a[1][1]), 2) # выводим x2
    x1 = round(((a[0][3] - x3 * a[0][2] - x2 * a[0][1]) / a[0][0]), 2) # выводим
x1
    print('x1 = ', x1, 'x2 = ', x2, 'x3 = ', x3)



def SimpleIteration():
    print('Метод простой итерации:')
    a = np.array([
        [3.11, -1.66, -0.60, -0.92],
        [-1.65, 3.51, -0.78, 2.57],
        [0.60, 0.78, -1.87, 1.65]
    ])
    x0 = [0, 0, 0] # вектор x(k-1)
    x1 = [0, 0, 0] # вектор x(k)
    kolIter = 0
    while True:
        # по формуле находим x1(k), x2(k), x3(k)
```

```
        x1[0] = (a[0][3] - a[0][2] * x0[2] - a[0][1] * x0[1]) / a[0][0]

        x1[1] = (a[1][3] - a[1][0] * x0[0] - a[1][2] * x0[2]) / a[1][1]

        x1[2] = (a[2][3] - a[2][0] * x0[0] - a[2][1] * x0[1]) / a[2][2]

        kolIter += 1 # увеличиваем количество итераций

        if max([abs(x1[0] - x0[0]), abs(x1[1] - x0[1]), abs(x1[2] - x0[2])]) < ep
s: # берем масимальное значение |x(k) - x(k-1)| и сравниваем его с eps
            break
        x0 = x1.copy() # обновляем x(k-1) для следующей итерации
    print('x1 = ', x1[0], 'x2 = ', x1[1], 'x1 = ', x1[2], '|max(x1 - x0))| < eps:
 ', max([abs(x1[0] - x0[0]), abs(x1[1] - x0[1]), abs(x1[2] - x0[2])]), 'Количеств
o итераций: ', kolIter)

def GaussSeidel():
    print('Метод Гаусса-Зейделя:')
    a = np.array([
        [3.11, -1.66, -0.60, -0.92],
        [-1.65, 3.51, -0.78, 2.57],
        [0.60, 0.78, -1.87, 1.65]
    ])
    x0 = [0, 0, 0] # вектор x(k-1)
    x1 = [0, 0, 0] # вектор x(k)
    kolIter = 0
    while True:
        # по формуле находим x1(k), x2(k), x3(k)
        x1[0] = (a[0][3] - a[0][2] * x0[2] - a[0][1] * x0[1]) / a[0][0]

        x1[1] = (a[1][3] - a[1][0] * x1[0] - a[1][2] * x0[2]) / a[1][1]

        x1[2] = (a[2][3] - a[2][0] * x1[0] - a[2][1] * x1[1]) / a[2][2]

        kolIter += 1 # увеличиваем количество итераций

        if max([abs(x1[0] - x0[0]), abs(x1[1] - x0[1]), abs(x1[2] - x0[2])]) < ep
s: # берем масимальное значение |x(k) - x(k-1)| и сравниваем его с eps
            break
        x0 = x1.copy() # обновляем x(k-1) для следующей итерации
    print('x1 = ', x1[0], 'x2 = ', x1[1], 'x1 = ', x1[2], '|max(x1 - x0))| < eps:
 ', max([abs(x1[0] - x0[0]), abs(x1[1] - x0[1]), abs(x1[2] - x0[2])]), 'Количеств
o итераций: ', kolIter)
```

Результат разработанной программы

```
Метод Гаусса:
[[ 3.11 -1.66 -0.6  -0.92]
 [ 0.    2.63 -1.1   2.08]
 [ 0.6   0.78 -1.87  1.65]]
[[ 3.11 -1.66 -0.6  -0.92]
 [ 0.    2.63 -1.1   2.08]
 [ 0.    1.1  -1.75  1.83]]
[[ 3.11 -1.66 -0.6  -0.92]
 [ 0.    2.63 -1.1   2.08]
 [ 0.    0.   -1.29  0.96]]
x1 =  -0.18 x2 =  0.48 x3 =  -0.74
Метод простой итерации:
x1 =  -0.17966795612913164 x2 =  0.4843626603216028 x1 =  -0.7377640635705753 |max(x1 - x0))| < eps:  0.0009867297845005951 Количество итераций:  11
Метод Гаусса-Зейделя:
x1 =  -0.179520542951465 x2 =  0.4838912219964912 x1 =  -0.7381161350874951 |max(x1 - x0))| < eps:  0.0008094023965607988 Количество итераций:  8
```

Вывод

Итерационные методы решения системы линейных уравнений удобно использовать при большом кол-ве неизвестных, т.к. сложность прямых вычислений вырастает при каждом новом неизвестном.

Метод Гаусса имеет высокую сложность вычислений и сложную реализацию для решения уравнений с любыми коэффициентами.

Методы Якоби и Гаусса-Зейделя имеют простую программную реализацию для решений системы линейных уравнений общего вида при любом кол-ве неизвестных (в ходе лабораторной реализованы частные случаи для 3ех неизвестных).

Метод Гаусса-Зейделя является более быстродейственным, чем метод Якоби, что было подтверждено практически в ходе выполнения работы.