Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Методы численного анализа»

ОТЧЕТ

к лабораторной работе №2

на тему:

«ЧИСЛЕННОЕ РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ МЕТОДОМ ПРОСТЫХ ИТЕРАЦИЙ И МЕТОДОМ ЗЕЙДЕЛЯ»

БГУИР 1-40-04-01

Выполнил студент группы 253504
Дмитрук Богдан Ярославович

____
(дата, подпись студента)

Проверил
Анисимов Владимир Яковлевич

____
(дата, подпись преподавателя)

Минск 2023

СОДЕРЖАНИЕ

1. ЦЕЛИ ВЫПОЛНЕНИЯ ЗАДАНИЯ

- Изучить итерационные методы решения СЛАУ (метод простых итераций, метод Зейделя).
- Составить алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ.
- Составить программу решения СЛАУ по разработанному алгоритму.
- Численно решить тестовые примеры и проверить правильность работы программы. Сравнить трудоемкость решения методом простых итерацией и методом Зейделя.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Итерационные методы основаны на построении сходящейся к точном решению $\boldsymbol{x}$ рекуррентной последовательности.

Для решения СЛАУ **методом простых итераций** преобразуем систему от первоначальной формы $\boldsymbol{Ax} = \boldsymbol{b}$ или

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1,$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2,$$

$$\ldots\ldots$$

(2.1)

$$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n$$

к виду

$$\boldsymbol{x} = \boldsymbol{Bx} + \boldsymbol{c}. \qquad (2.2)$$

Здесь $\boldsymbol{B}$ – квадратная матрица с элементами b_{ij} ($i, j = 1, 2, \ldots, n$), $\boldsymbol{c}$ – вектор-столбец с элементами c_i ($i = 1, 2, \ldots, n$).

В развернутой форме записи система (2.2) имеет следующий вид:

$$x_1 = b_{11}x_1 + b_{12}x_2 + b_{13}x_3 + \ldots + b_{1n}x_n + c_1$$

$$x_2 = b_{21}x_1 + b_{22}x_2 + b_{23}x_3 + \ldots + b_{2n}x_n + c_2$$

. . . .

$$x_n = b_{n1}x_1 + b_{n2}x_2 + b_{n3}x_3 + \ldots + b_{nn}x_n + c_n$$

Вообще говоря, операция *приведения системы к виду, удобному для итераций*, не является простой и требует специальных знаний, а также существенного использования специфики системы.

Можно, например, преобразовать систему (2.1) следующим образом

$$x_1 = (b_1 - a_{11}x_1 - a_{12}x_2 - \ldots - a_{1n}x_n) / a_{11} + x_1,$$

$$x_2 = (b_2 - a_{21}x_1 - a_{22}x_2 - \ldots - a_{2n}x_n) / a_{22} + x_2,$$

…..

$$x_n = (b_n - a_{n1}x_1 - a_{n2}x_2 - \ldots - a_{nn}x_n) / a_{nn} + x_n$$

если диагональные элементы матрицы $\boldsymbol{A}$ отличны от нуля.

Можно преобразовать систему (2.1) в эквивалентную ей систему

$\boldsymbol{x = (E\text{-}A)x+b}$.

Задав произвольным образом столбец начальных приближений $\boldsymbol{x}^0 = (x_1^0, x_2^0, \ldots, x_n^0)^T$, подставим их в правые части системы (2.2) и вычислим новые приближения $\boldsymbol{x}^1 = (x_1^1, x_2^1, \ldots, x_n^1)^T$, которые опять подставим в систему (2.2) и т.д. Таким образом, организуется итерационный процесс

$\boldsymbol{x}^k = \boldsymbol{B}\boldsymbol{x}^{k-1} + \boldsymbol{c}$, k = 1,2, …. ,. Известно, что система (2.1) имеет единственное решение $\boldsymbol{x}^*$ и последовательность $\{\boldsymbol{x}^k\}$ сходится к этому решению со скоростью геометрической прогрессии, если $\| \boldsymbol{B} \| < 1$ в любой матричной норме. Т.е.

Метод Зейделя. Метод Зейделя является модификацией метода простых итераций. Суть его состоит в том, что при вычислении следующего x_i^k : 2 $\le 1 \le n$ в формуле $\boldsymbol{x^k = Bx^{k-1} + c}$, k = 1,2, …. используются вместо $x_1^{k-1},\dots,x_{i-1}^{k-1}$ уже вычисленные ранее $x_1^k,\dots,x_{i-1}^k$, т.е.

$$x_i^k = \sum_{j=1}^{i-1} g_{ij} x_j^k + \sum_{j=i+1}^{n} g_{ij} x_j^{k-1} + c_i \quad . \tag{2.3}$$

Схема алгоритма аналогична схеме метода простых итераций.

Самый простой способ приведения системы к виду, удобному для итераций, состоит в следующем. Из первого уравнения системы выразим неизвестное x_1:

$x_1 = a_{11}{}^{-1} (b_1 - a_{12}x_2 - a_{13}x_3 - \dots - a_{1n}x_n)$,

из второго уравнения – неизвестное x_2:

$x_2 = a_{21}{}^{-1} (b_2 - a_{22}x_2 - a_{23}x_3 - \dots - a_{2n}x_n)$,

и т. д. В результате получим систему

$x_1 = b_{12}x_2 + \quad b_{13}x_3 + \dots + \quad b_{1,n-1}x_{n-1} + \quad b_{1n}x_n + c_1$,

$x_2 = b_{21}x_1 + \quad b_{23}x_3 + \ldots + \; b_{2,n-1}x_{n-1} + \quad b_{2n}x_n + c_2$,

$x_3 = b_{31}x_1 + \quad b_{32}x_2 + \ldots + \; b_{3,n-1}x_{n-1} + \quad b_{3n}x_n + c_3$,

. . . .

$x_n = b_{n1}x_1 + \quad b_{n2}x_2 + b_{n3}x_3 + \quad \ldots + \quad b_{n,n-1}x_{n-1} + \quad c_n$,

в которой на главной диагонали матрицы ***B*** находятся нулевые элементы. Остальные элементы выражаются по формулам

$$b_{ij} = -a_{ij} \,/\, a_{ii},\; c_i = b_i \,/\, a_{ii} \; (i, j = 1, 2, \ldots, n, j \neq i) \qquad (2.4)$$

Конечно, для возможности выполнения указанного преобразования необходимо, чтобы диагональные элементы матрицы ***A*** были ненулевыми.

Введем нижнюю $\boldsymbol{B}_1$ (получается из ***B*** заменой нулями элементов стоявших на главной диагонали и выше ее) и верхнюю ***B***2 (получается из ***B*** заменой нулями элементов стоявших на главной диагонали и ниже ее) треугольные матрицы.

Заметим, что $\boldsymbol{B} = \boldsymbol{B}_1 + \boldsymbol{B}_2$ и поэтому решение $\boldsymbol{x}$ исходной системы удовлетворяет равенству

$$\boldsymbol{x} = \boldsymbol{B}_1\boldsymbol{x} + \boldsymbol{B}_2\,\boldsymbol{x} + \boldsymbol{c} \qquad (2.5)$$

Выберем начальное приближение $\boldsymbol{x}^{(0)} = [x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}]^{\mathrm{T}}$. Подставляя его в правую часть равенства при верхней треугольной матрице $\boldsymbol{B}_2$ и вычисляя полученное выражение, находим первое приближение

$$\boldsymbol{x}^{(1)} = \boldsymbol{B}_1\boldsymbol{x}^{(0)} + \boldsymbol{B}_2\boldsymbol{x}^{(1)} \qquad (2.6)$$

Подставляя приближение $\boldsymbol{x}^{(1)}$, получим

$$\boldsymbol{x}^{(2)} = \boldsymbol{B}_1\boldsymbol{x}^{(1)} + \boldsymbol{B}_2\boldsymbol{x}^{(2)} \qquad (2.7)$$

Продолжая этот процесс далее, получим последовательность $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$, … приближений к вычисляемых по формуле

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{B}_1^{(k+1)} + \boldsymbol{B}_2^{(k)} + \boldsymbol{c} \qquad (2.8)$$

или в развернутой форме записи

$\mathrm{x}_1^{(k+1)} = \quad b_{12}x_2^{(k)} \; + \; b_{13}x_2^{(k)} \; + \quad \ldots \quad + \quad b_{1n}x_n^{(k)} + c_1$,

$$x_2^{(k+1)} = b_{21}x_1^{(k+1)} + b_{23}x_3^{(k)} + \ldots + b_{2n}x_n^{(k)} + c_2,$$

$$x_3^{(k+1)} = b_{31}x_1^{(k+1)} + b_{32}x_2^{(k+1)} + \ldots + b_{3n}x_n^{(k)} + c_3,$$

. . . .

$$x_n^{(k+1)} = b_{n1}x_1^{(k+1)} + b_{n2}x_2^{(k+1)} + b_{n3}x_3^{(k+1)} + \ldots + c_n.$$

Объединив приведение системы к виду, удобному для итераций и метод Зейделя в одну формулу, получим

$$x_i^{(k+1)} = x_i^{(k)} - a_{ii}^{-1}\left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} + \sum_{j=1}^{n} a_{ij}x_i^{(k)} - b_i\right). \qquad (2.9)$$

Тогда достаточным условием сходимоти метода Зейделя будет условие *доминированния диагональных элементов в строках или столбцах матрицы* ***A****, т.е.*

$a_{ii} > a_{i1} + \ldots\ldots + a_{in}$ *для всех* $i=1,\ldots n$,

или $a_{jj} > a_{1j} + \ldots\ldots + a_{nj}$ *для всех* $j=1,\ldots.,n$.

3. ЗАДАНИЕ

ЗАДАНИЕ. Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение системы ***Ax=b***,

где A = kC + D, A – исходная матрица для расчёта, k – номер варианта (0-*15*), ***матрицы*** C, D ***и вектор свободных членов*** b ***задаются ниже.***

$$C = \begin{bmatrix} 0{,}01 & 0 & -0{,}02 & 0 & 0 \\ 0{,}01 & 0{,}01 & -0{,}02 & 0 & 0 \\ 0 & 0{,}01 & 0{,}01 & 0 & -0{,}02 \\ 0 & 0 & 0{,}01 & 0{,}01 & 0 \\ 0 & 0 & 0 & 0{,}01 & 0{,}01 \end{bmatrix}, \quad D = \begin{bmatrix} 1{,}33 & 0{,}21 & 0{,}17 & 0{,}12 & -0{,}13 \\ -0{,}13 & -1{,}33 & 0{,}11 & 0{,}17 & 0{,}12 \\ 0{,}12 & -0{,}13 & -1{,}33 & 0{,}11 & 0{,}17 \\ 0{,}17 & 0{,}12 & -0{,}13 & -1{,}33 & 0{,}11 \\ 0{,}11 & 0{,}67 & 0{,}12 & -0{,}13 & -1{,}33 \end{bmatrix}.$$

Вектор ***b***= (1,2; 2,2; 4,0; 0,0; -1,2)^{T}.

Вариант 5 (k = 5)

4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

```
#-------------главный исполняемый файл------------

from simple_iteration import simple_iteration as simp
from seidel import seidel as sd
import secondary_functions as sf
import numpy as np
import initial_data as init

np.seterr(over="ignore")

print("\nВходные данные варианта 5\n")
sf.test(simp, init.get_initial_matrix)
sf.test(sd, init.get_initial_matrix)

print("\nВходные данные теста 1\n")
sf.test(simp, init.test1)
sf.test(sd, init.test1)

print("\nВходные данные теста 2\n")
sf.test(simp, init.test2)
sf.test(sd, init.test2)

print("\nВходные данные теста 3\n")
sf.test(simp, init.test3)
sf.test(sd, init.test3)

#------------файл с исходными данными----------------

import numpy as np

Accuracy = 1e-5

Limit_number_of_iterations = 50

C = np.array([
        [0.01, 0, -0.02, 0, 0],
        [0.01, 0.01, -0.02, 0, 0],
        [0, 0.01, 0.01, 0, -0.02],
        [0, 0, 0.01, 0.01, 0],
        [0, 0, 0, 0.01, 0.01]
    ])

D = np.array([

        [1.33, 0.21, 0.17, 0.12, -0.13],
```

```
        [-0.13, -1.33, 0.11, 0.17, 0.12],
        [0.12, -0.13, -1.33, 0.11, 0.17],
        [0.17, 0.12, -0.13, -1.33, 0.11],
        [0.11, 0.67, 0.12, -0.13, -1.33]
    ])

b = np.array([
        [1.2],
        [2.2],
        [4.0],
        [0.0],
        [-1.2]
    ])

k = 5


def test1(): #Ошибка из-за нулевого диагонального элемента
    M = np.array([
        [0, 1, 2],
        [3, 4, 5],
        [6, 7, 8]
    ])

    B = np.array([
        [1],
        [2],
        [3]
    ])

    return (M, B)

def test2(): #Потенциальная расходимость итерационного процесса
    M = np.array([
        [2, 1, 1],
        [1, 2, 1],
        [1, 1, 2]
    ])

    B = np.array([
        [1],
        [2],
        [3]
    ])

    return (M, B)

def test3(): #Ошибка из-за расходимости последовательности
```

```
    M = np.array([
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]
    ])

    B = np.array([
        [1],
        [2],
        [3]
    ])

    return (M, B)

def get_initial_matrix():
    M = k*C + D

    col = M.shape[0]
    M = np.transpose(M)
    for c in range(col):
        if all(i == 0 for i in M[c]):
            M = np.delete(M, c, axis = 0)
    M = np.transpose(M)

    M_ex = np.hstack((M,b))

    if np.linalg.matrix_rank(M) != np.linalg.matrix_rank(M_ex):
        print("the matrix is inconsistent")
        exit(0)
    if np.linalg.matrix_rank(M) != len(M[0]):
        print("the matrix has an infinite number of solutions")
        exit(0)
    return (M, b)

#------------файл реализации метода простых итераций----------

import numpy
from initial_data import Accuracy as ac
import secondary_functions as sf
from initial_data import Limit_number_of_iterations as il


def simple_iteration(A, b):

    print("\n---------------------------Метод простых итераций------
---------------------\n")

    n = A.shape[0]
```

```
    for i in range(n):
        if A[i, i] == 0.0:
            print("Возникла ошибка:\n"
                  + "Обнаружен нулевой диагональный элемент!")
            return sf.err(n)

    transition_matrix = sf.get_transition_matrix(A)

    if not (min(sf.Norms(transition_matrix)) < 1):
        print("Возникла потенциальная расходимость итерационного
процесса:\n||B|| >= 1.")

    initial_solution  = numpy.zeros((n, 1))
    for i in range(n):
        initial_solution [i] = b[i] / A[i, i]

    x = numpy.zeros((n, 1))
    count = 0
    deltax = ac
    deltaf = ac

    print("Промежуточные результаты метода простого
итерирования:\n")

    while deltax + deltaf > ac:

        oldx = x
        x = initial_solution  + transition_matrix.dot(x)
        deltax = numpy.absolute((x - oldx)).max()
        deltaf = numpy.absolute((A.dot(x) - b)).max()

        if not numpy.isfinite(deltax + deltaf) or count > il:
            print("Возникла ошибка:\n"
                  + f"Последовательность {x} расходится")
            return sf.err(n)

        for cur in range(len(x)):
            print("%.4f" % x[cur, 0], end=", ")
        print()

        count += 1
    print("Количество итераций:", count)

    return x

#----------- файл реализации метода Зейделя ------------------
```

```
from fractions import Fraction as fr
import numpy as np
import copy

def full_mod(M, cur_col, arr = None):
    row, col = M.shape
    mr, mc = cur_col, cur_col

    max = np.abs(M[cur_col][cur_col])
    for r in range(cur_col, row):
        for c in range(cur_col, col - 1):
            if np.abs(M[r][c]) > max:
                max = np.abs(M[r][c])
                mr, mc = r,c

    arr[cur_col], arr[mc] = arr[mc], arr[cur_col]

    M[mr], M[cur_col] = M[cur_col].copy(), M[mr].copy()

    M[:, [cur_col, mc]] = M[:, [mc, cur_col]]

#----------- файл реализации вспомогательных функций ---------


import numpy as np
import initial_data as init


def err(n):
    v = np.zeros((n, 1))
    v[:] = np.NaN
    return v

def Norms(A):
    n = A.shape[0]
    f = max(np.absolute(A[i]).sum() for i in range(n))
    s = max(np.absolute(A.T[j]).sum() for j in range(n))
    t = ((A**2).sum()) ** (1 / 2)
    return (f, s, t)

def get_transition_matrix(A):
    n = A.shape[0]
    alpha = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            alpha[i, j] = -A[i, j] / A[i, i]
        alpha[i, i] = 0
    return alpha
```

```
def output(A, x, b):
    np.set_printoptions(suppress=True, precision=4,
floatmode="fixed")
    if not np.isnan(x).any():
        print(f"A = \n{A}\n"
            +f"x = \n{x.T}\n"
            +f"Проверка: b = \n{b.T}\n{A.dot(x).T}")




def test(method, initial):
    (A, b) = initial()
    x = method(A.copy(), b.copy())
    output(A, x, b)
```

5. ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

Вывод программного продукта

```
---------------------------Метод простых итераций-------------------
--------


Промежуточные результаты метода простого итерирования:

0.8696, -1.7188, -3.1250, 0.0000, 0.9375,
1.3779, -1.7096, -2.8848, 0.2302, -0.1804,
1.2390, -1.8137, -2.8790, 0.1875, -0.1238,
1.2636, -1.8054, -2.8861, 0.1638, -0.1870,
1.2588, -1.8161, -2.8898, 0.1629, -0.1797,
1.2614, -1.8152, -2.8893, 0.1621, -0.1860,
1.2607, -1.8161, -2.8895, 0.1620, -0.1853,
1.2610, -1.8160, -2.8895, 0.1619, -0.1858,
1.2609, -1.8161, -2.8895, 0.1619, -0.1857,
1.2609, -1.8160, -2.8895, 0.1618, -0.1857,
Количество итераций: 10
A =
[[ 1.3800  0.2100  0.0700  0.1200 -0.1300]
 [-0.0800 -1.2800  0.0100  0.1700  0.1200]
 [ 0.1200 -0.0800 -1.2800  0.1100  0.0700]
 [ 0.1700  0.1200 -0.0800 -1.2800  0.1100]
 [ 0.1100  0.6700  0.1200 -0.0800 -1.2800]]
b =
[[ 1.2000  2.2000  4.0000  0.0000 -1.2000]]
x =
[[ 1.2609 -1.8160 -2.8895  0.1618 -0.1857]]

Проверка: b =
[[ 1.2000  2.2000  4.0000  0.0000 -1.2000]]
[[ 1.2000  2.2000  4.0000 -0.0000 -1.2000]]

---------------------------Метод Зейделя---------------------------


Промежуточные результаты метода Зейделя:

0.8696, -1.7731, -2.9327, 0.1326, -0.1991,
1.2579, -1.8213, -2.8927, 0.1600, -0.1890,
1.2617, -1.8167, -2.8898, 0.1616, -0.1860,
1.2610, -1.8161, -2.8896, 0.1618, -0.1858,
1.2609, -1.8161, -2.8895, 0.1618, -0.1857,
1.2609, -1.8161, -2.8895, 0.1618, -0.1857,
Количество итераций: 6
```

```
A =
[[ 1.3800  0.2100  0.0700  0.1200 -0.1300]
 [-0.0800 -1.2800  0.0100  0.1700  0.1200]
 [ 0.1200 -0.0800 -1.2800  0.1100  0.0700]
 [ 0.1700  0.1200 -0.0800 -1.2800  0.1100]
 [ 0.1100  0.6700  0.1200 -0.0800 -1.2800]]
b =
[[ 1.2000  2.2000  4.0000  0.0000 -1.2000]]
x =
[[ 1.2609 -1.8161 -2.8895  0.1618 -0.1857]]

Проверка: b =
[[ 1.2000  2.2000  4.0000  0.0000 -1.2000]]
[[ 1.2000  2.2000  4.0000  0.0000 -1.2000]]

Входные данные теста 1


---------------------------Метод простых итераций-------------------
--------


Возникла ошибка:
Обнаружен нулевой диагональный элемент!

A =
[[0 1 2]
 [3 4 5]
 [6 7 8]]
b =
[[1 2 3]]
x =
[[nan nan nan]]


---------------------------Метод Зейделя---------------------------


Возникла ошибка:
Обнаружен нулевой диагональный элемент!

A =
[[0 1 2]
 [3 4 5]
 [6 7 8]]
b =
[[1 2 3]]
x =
[[nan nan nan]]


Входные данные теста 2
```

```
---------------------------Метод простых итераций-------------------
--------

Возникла потенциальная расходимость итерационного процесса:
||B|| >= 1.

Промежуточные результаты метода простого итерирования:

0.5000, 1.0000, 1.5000,
-0.7500, 0.0000, 0.7500,
0.1250, 1.0000, 1.8750,
-0.9375, 0.0000, 0.9375,
0.0312, 1.0000, 1.9688,
-0.9844, 0.0000, 0.9844,
0.0078, 1.0000, 1.9922,
-0.9961, 0.0000, 0.9961,
0.0020, 1.0000, 1.9980,
-0.9990, 0.0000, 0.9990,
0.0005, 1.0000, 1.9995,
-0.9998, 0.0000, 0.9998,
0.0001, 1.0000, 1.9999,
-0.9999, 0.0000, 0.9999,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
```

```
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,
-1.0000, 0.0000, 1.0000,
0.0000, 1.0000, 2.0000,

Возникла ошибка:
Последовательность расходится

A =
[[2 1 1]
 [1 2 1]
 [1 1 2]]
b =
[[1 2 3]]
x =
[[nan nan nan]]


---------------------------Метод Зейделя---------------------------

Внимание: Метод Зейделя может расходится

Промежуточные результаты метода Зейделя:

0.5000, 0.7500, 0.8750,
-0.3125, 0.7188, 1.2969,
-0.5078, 0.6055, 1.4512,
-0.5283, 0.5386, 1.4949,
-0.5167, 0.5109, 1.5029,
-0.5069, 0.5020, 1.5025,
-0.5022, 0.4999, 1.5012,
-0.5005, 0.4997, 1.5004,
-0.5001, 0.4998, 1.5001,
-0.5000, 0.4999, 1.5000,
-0.5000, 0.5000, 1.5000,
Количество итераций: 11
A =
[[2 1 1]
 [1 2 1]
 [1 1 2]]
b =
[[1 2 3]]
x =
[[-0.5000  0.5000  1.5000]]

Проверка: b =
[[1 2 3]]
[[1.0000 2.0000 3.0000]]

Входные данные теста 3


---------------------------Метод простых итераций-------------------
--------
```

```
Возникла потенциальная расходимость итерационного процесса:
||B|| >= 1.

Промежуточные результаты метода простого итерирования:

1.0000, 0.4000, 0.3333,
-0.8000, -0.8000, -0.8000,
5.0000, 2.0000, 1.6667,
-8.0000, -5.6000, -5.3333,
28.2000, 13.2000, 11.5333,
-60.0000, -36.0000, -33.3333,
173.0000, 88.4000, 79.0000,
-412.8000, -232.8000, -212.8000,
1105.0000, 586.0000, 528.3333,
-2756.0000, -1517.6000, -1380.0000,
7176.2000, 3861.2000, 3492.8667,
-18200.0000, -9932.0000, -9013.3333,
46905.0000, 25376.4000, 22984.3333,
-119704.8000, -65104.8000, -59038.1333,
307325.0000, 166610.0000, 150975.0000,
-786144.0000, -427029.6000, -387128.0000,
2015444.2000, 1093469.2000, 991027.5333,
-5160020.0000, -2801588.0000, -2539540.0000,
13221797.0000, 7175464.4000, 6503649.6667,
-33861876.8000, -18381816.8000, -16661810.1333,
86749065.0000, 47083674.0000, 42676408.3333,
-222196572.0000, -120610941.6000, -109323649.3333,
569192832.2000, 308945637.2000, 280029282.2000,
-1457979120.0000, -791389404.0000, -717323880.0000,
3734750449.0000, 2027171952.4000, 1837441008.3333,
-9566666928.8000, -5192729568.8000, -4706736528.8000,
24505668725.0000, 13301417378.0000, 12056500561.6667,
-62772336440.0000, -34072335653.6000, -30883446677.3333,
160795011340.2000, 87278005165.2000, 79109448923.5333,
-411884357100.0000, -223567347780.0000, -202643235633.3333,
1055064402460.9999, 572679368440.4000, 519081031326.9999,
-2702601830860.7998, -1466948759560.8000, -1329653973860.8000,
6922859440705.0000, 3757666233322.0000, 3405978099168.3335,
-17733266764148.0000, -9625461271565.5996, -8724593994612.0000,
45424704526968.2031, 24656126204853.1992, 22348506391284.8633,
-116357771583559.9844, -63157971291116.0000, -57246882369733.3281,
298056589691433.0000, 161782476110528.4062, 146640907934872.3125,
-763487676025672.7500, -414414361274992.8125, -375628437413806.1875,
1955714034791405.2500, 1061544265717106.0000, 962192069153295.0000,
-5009664738894096.0000, -2719201710817077.5000, -
2464705818808520.0000,
12832520878059716.0000, 6965378773685501.0000,
6313474095421699.0000,
-32871179833636100.0000, -17842185616953812.0000, -
16172297370655780.0000,
84201263345874960.0000, 45703700711695816.0000,
41426193752342576.0000,
-215685982680419360.0000, -117072443179511056.0000, -
106115383234965696.0000,
```

```
552491036063919232.0000, 299887246026294336.0000,
271820158244336000.0000,
-1415234966785596672.0000, -768177018744338560.0000, -
696281691184198784.0000,
3625199111041273344.0000, 1967726002849516032.0000,
1783562324161542656.0000,
-9286138978183659520.0000, -5040434077826870272.0000, -
4568689088898337792.0000,
23786935422348754944.0000, 12911338089224933376.0000,
11702938385544509440.0000,
-60931491335083393024.0000, -33073074400532414464.0000, -
29977694741137862656.0000,
156079233024478412800.0000, 84718426757432147968.0000,
76789448283315896320.0000,

Возникла ошибка:
Последовательность расходится

A =
[[1 2 3]
 [4 5 6]
 [7 8 9]]
b =
[[1 2 3]]
x =
[[nan nan nan]]


---------------------------Метод Зейделя---------------------------

Внимание: Метод Зейделя может расходится

Промежуточные результаты метода Зейделя:

1.0000, -0.4000, -0.0889,
2.0667, -1.1467, -0.2548,
4.0578, -2.5404, -0.5645,
7.7745, -5.1422, -1.1427,
14.7124, -9.9987, -2.2219,
27.6632, -19.0642, -4.2365,
51.8380, -35.9866, -7.9970,
96.9643, -67.5750, -15.0167,
181.1999, -126.5400, -28.1200,
338.4399, -236.6079, -52.5795,
631.9545, -442.0681, -98.2374,
1179.8483, -825.5938, -183.4653,
2202.5836, -1541.5085, -342.5574,
4111.6894, -2877.8826, -639.5295,
7675.3535, -5372.4474, -1193.8772,
14327.5265, -10028.9685, -2228.6597,
26744.9161, -18721.1413, -4160.2536,
49924.0434, -34946.5304, -7765.8956,
93191.7476, -65233.9233, -14496.4274,
173958.1289, -121770.3902, -27060.0867,
324722.0406, -227305.1284, -50512.2508,
```

```
606148.0091, -424303.3063, -94289.6236,
1131476.4836, -792033.2385, -176007.3863,
2112089.6360, -1478462.4452, -328547.2100,
3942567.5205, -2759796.9644, -613288.2143,
7359459.5717, -5151621.4002, -1144804.7556,
13737658.0671, -9616360.3470, -2136968.9660,
25643628.5919, -17950539.7144, -3989008.8254,
47868106.9049, -33507674.5335, -7446149.8963,
89353799.7559, -62547659.5291, -13899479.8954,
166793759.7444, -116755631.5210, -25945695.8936,
311348351.7228, -217943845.9060, -48431965.7569,
581183590.0826, -406828512.7578, -90406336.1684,
1084876035.0208, -759413224.2145, -168758494.2699,
2025101932.2388, -1417571352.2671, -315015856.0594,
3780190273.7123, -2646133191.2986, -588029598.0664,
7056355177.7964, -4939448624.1575, -1097655249.8128,
13171862998.7532, -9220304098.8273, -2048956466.4061,
24587477597.8727, -17211234318.2109, -3824718737.3802,
45896624849.5624, -32127637394.3937, -7139474976.5319,
85673699719.3831, -59971589803.2682, -13327019956.2818,
159924239476.3818, -111946967633.1673, -24877103918.4816,
298525247022.7794, -208967672915.6456, -46437260647.9212,
557247127776.0548, -390072989442.9384, -86682886542.8752,
1040194638515.5023, -728136246960.5515, -161808054880.1227,
1941696658562.4712, -1359187660993.4297, -302041702442.9846,
3624500429316.8135, -2537150300521.4692, -563811177893.6602,
6765734134724.9189, -4736013894307.1426, -1052447532068.2544,
12629370384820.0469, -8840559269373.7324, -1964568726527.4966,
23574824718330.9531, -16502377302831.3672, -3667194956184.7500,
44006339474217.9844, -30804437631952.2852, -6845430584878.2881,

Возникла ошибка:
Последовательность расходится

A =
[[1 2 3]
 [4 5 6]
 [7 8 9]]
b =
[[1 2 3]]
x =
[[nan nan nan]]
```

6. ВЫВОДЫ

В ходе лабораторной работы был применены итерационные методы решения СЛАУ в двух вариантах: метод простых итераций и метод Зейделя. Были разработаны алгоритмы решения СЛАУ указанными методами, составлена программа по разработанным алгоритмам, решены тестовые примеры.

На основании тестовых примеров можно сделать следующие выводы:

1. Метод простых итераций более ресурсозатратный (исходя из примеров, приведенных в моих результатах) из-за большего количества проводимых итераций.
2. Метод Зейделя является более быстрой модификацией метода простых итерация.
3. Оба метода позволяют получить решение с заданной точностью, причем корни в обоих методах могут отличаться в пределах заданной погрешности.
4. Если не выполняется необходимое и достаточное условие сходимости, то следует прибегнуть к преобразованиям, чтобы попытаться решить заданое СЛАУ.