Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Методы численного анализа»

**Отчет по лабораторной работе № 2**

«Численное решение систем линейных уравнений методом простых итераций и методом Зейделя»

Выполнил: студент гр. 053501

Кононович Станислав Владимирович

Проверил: Анисимов Владимир Яковлевич

Минск 2022

**Вариант 6**

**Цели работы:**

* Изучить итерационные методы решения СЛАУ (метод простых итераций, метод Зейделя);
* Составить алгоритмы решения СЛАУ указанными методами, применимыми для организаций вычислений на ЭВМ;
* Составить программу решения СЛАУ по разработанным алгоритмам;
* Численно решить тестовые примеры и проверить правильность работы программы. Сравнить трудоемкость решения методом простых итераций и методом Зейделя.

**ВВЕДЕНИЕ**

Прямые методы применяют главным образом для решения задач малой размерности, когда нет ограничений в доступной оперативной памяти ЭВМ или необходимости выполнения чрезмерно большого числа арифметических операций. Большие системы уравнений, возникающие в основном в приложениях, как правило, являются разреженными. Методы исключения для систем с разреженным и матрицами неудобны, например, тем, что при их использовании большое число нулевых элементов превращается в ненулевые, и матрица теряет свойство разреженности. В противоположность им при использовании итерационных методов в ходе итерационного процесса матрица не меняется, и она, естественно, остается разреженной. Большая эффективность итерационных методов по сравнению с прямыми методами тесно связанна с возможностью существенного использования разреженности матриц.

Итерационные методыоснованы на построении сходящейся к точном решению ***x*** рекуррентной последовательности.

Популярными среди итерационных методов являются **метод простых итераций** и **метод Зейделя**.

**ДЕТАЛЬНОЕ ОПИСАНИЕ И СХЕМА АЛГОРИТМА МЕТОДОВ**

Для решения СЛАУ **методом простых итераций** преобразуем систему от первоначальной формы ***Ax*** = ***b***  или

*a11 x1 + a12 x2 + … + a1n xn = b1* ,

*a21 x1 + a22 x2 + … + a2n xn = b2* , (1.1)

…..

*an1 x1 + an2 x2 + … + ann xn = bn*

к виду

***x*** = ***Bx*** + ***c***. (1.2)

Здесь ***B*** – квадратная матрица с элементами *bij* (*i, j* = 1, 2, …, *n*), ***c*** – вектор-столбец с элементами *ci* (*i* = 1, 2, …, *n*).

### В развернутой форме записи система (1.2) имеет следующий вид:

*x*1 = *b*11*x*1 + *b*12*x*2 + *b*13*x*3 + … + *b*1*nxn* + *c*1

*x*2 = *b*21*x*1 + *b*22*x*2 + *b*23*x*3 + … + *b*2*nxn* + *c*2

. . . . . . . . . . . . . . . . .

*xn* = *bn*1*x*1 + *bn*2*x*2 + *bn*3*x*3 + … + *bnnxn* + *cn*

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 (1.1) следующим образом

*x1 = (b1 - a11 x1 – a12 x2 - … - a1n xn) / a11 + x1,*

*x2 = (b2 - a21 x1 – a22 x2 - … - a2n xn) / a22 + x2,*

…..

*xn = (bn - an1 x1 – an2 x2 - … - ann xn) / ann + xn*

если диагональные элементы матрицы ***А*** отличны от нуля.

Можно преобразовать систему (1.1) в эквивалентную ей систему

***x = (E-A)x+b.***

Задав произвольным образом столбец начальных приближений ***х0*** = (x10 , x20 , … , xn0 )T , подставим их в правые части системы (1.2) и вычислим новые приближения ***x1 =*** (x11 , x21 , … , xn1 )T , которые опять подставим в систему (1.2) и т.д. Таким образом, организуется итерационный процесс

***xk*** = ***Bxk-1 + c*** , k = 1,2, …. ,. Известно, что система (1.1) имеет единственное решение ***х\**** и последовательность {***xk***} сходится к этому решению со скоростью геометрической прогрессии, если || ***B*** || < 1 в любой матричной норме. Т.е. для того, чтобы последовательность простых итераций сходилась к единственному решению достаточно, чтобы выполнялось одно из следующих условий:

1) max ( **|** bij **|** ) < 1 ; 2)   bij 2 < 1; 3) max ( **|** bij **|** ) < 1.

i

j = 1

n

n

n

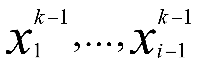
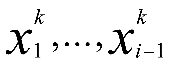
i = 1 j=1

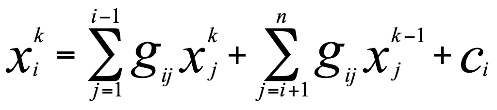
n

ji

i = 1

**Метод Зейделя**.

Метод Зейделя является модификацией метода простых итераций. Суть его состоит в том, что при вычислении следующего  : 2 в формуле ***xk*** = ***Bxk-1 + c*** , k = 1,2, …. используются вместо  уже вычисленные ранее  , т.е.

 . (1.3)

Такое усовершенствование позволяет ускорить сходимость итераций почти в два раза. Кроме того, данный метод может быть реализован на ЭВМ без привлечения дополнительного массива, т.к. полученное новое  сразу засылается на место старого.

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 – … – *a*1*nxn*),

из второго уравнения – неизвестное *x*2:

*x*2 = *a*21–1 (*b*2 – *a*22*x*2 – *a*23*x*3 – … – *a*2*nxn*),

и т. д. В результате получим систему

*x*1 = *b*12*x*2 + *b*13*x*3 + … + *b*1,*n*–1*xn*–1 + *b*1*nxn*+ *c*1 ,

*x*2 = *b*21*x*1 + *b*23*x*3 + … + *b*2,*n*–1*xn*–1 + *b*2*nxn*+ *c*2 ,

*x*3 = *b*31*x*1 + *b*32*x*2 + … + *b*3,*n*–1*xn*–1 + *b*3*nxn*+ *c*3 ,

. . . . . . . . . . . . . . . . . . . . . . .

*xn* = *bn*1*x*1 + *bn*2*x*2 + *bn*3*x*3 + … + *bn*,*n*–1*xn*–1 + *cn* ,

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

*bij* = –*aij* / *aii, ci* = *bi* / *aii* (*i, j* = 1, 2, …, *n, j ≠ i*) (1.4)

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

Введем нижнюю ***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 ***B*** = ***B***1+ ***B***2 и поэтому решение ***x*** исходной системы удовлетворяет равенству

***x*** = ***B***1***x + B***2 ***x + c*** (1.5)

Выберем начальное приближение ***x***(0) = [*x*1(0), *x*2(0), …, *xn*(0)]T. Подставляя его в правую часть равенства при верхней треугольной матрице ***B***2и вычисляя полученное выражение, находим первое приближение

***x***(1) = ***B***1***x***(0) + ***B***2***x***(1) (1.6)

Подставляя приближение ***x***(1), получим

***x***(2) = ***B***1***x***(1) + ***B***2***x***(2) (1.7)

Продолжая этот процесс далее, получим последовательность ***x***(0), ***x***(1), …, ***x***(*n*), … приближений к вычисляемых по формуле

***x***(*k*+1) = ***B***1(*k*+1) + ***B***2(*k*) + ***c*** (1.8)

или в развернутой форме записи

x1(*k*+1) = *b*12*x*2(*k*) + *b*13*x*2(*k*) + … + *b*1*nxn*(*k*) + *c*1 ,

x2(*k*+1) = *b*21*x*1(*k*+1) + *b*23*x*3(*k*) + … + *b*2*nxn*(*k*) + *c*2 ,

x3(*k*+1) = *b*31*x*1(*k*+1) + *b*32*x*2(*k*+1) + … + *b*3*nxn*(*k*) + *c*3 ,

. . . . . . . . . . . . . . . . . . . . . . . . . .

xn(*k*+1) = *bn*1*x*1(*k*+1) + *bn*2*x*2(*k*+1) + *bn*3*x*3(*k*+1) + … + c*n* .

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

*xi*(*k*+1) = *x*i(*k*) – *aii*–1(∑*j*=1*i*–1 *aijxj*(*k*+1) + ∑*j*=1*n* *aijxi*(*k*) – *bi*). (1.9)

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

*aii>ai1+ ……..+ain**для всех i=1,…n,*

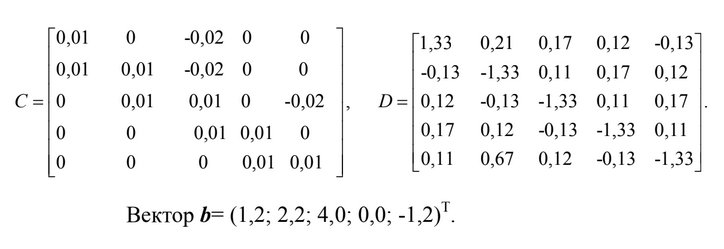
*или*

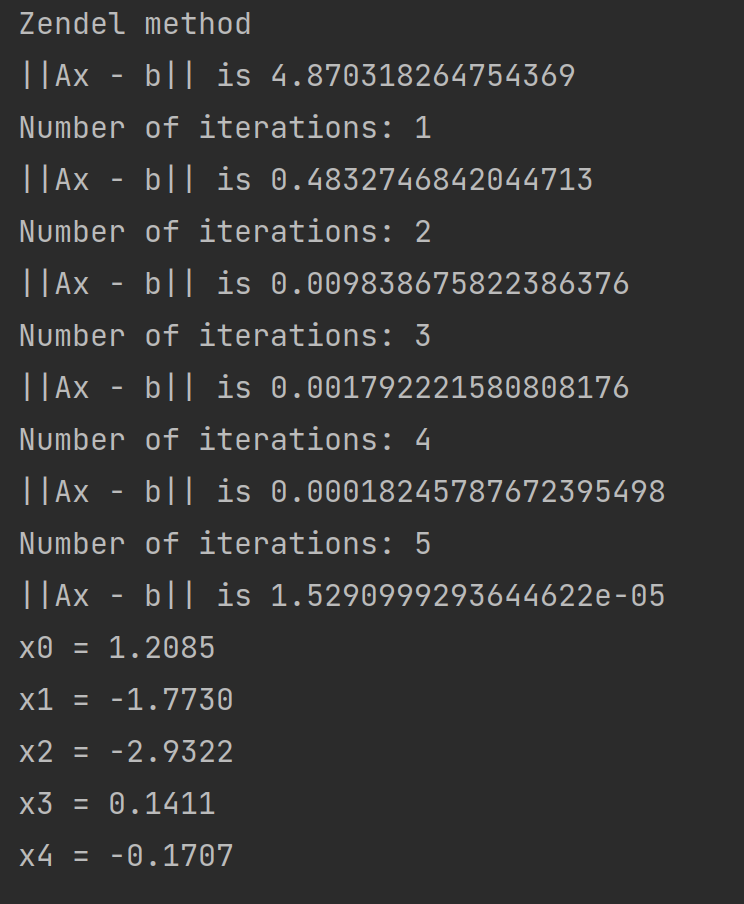
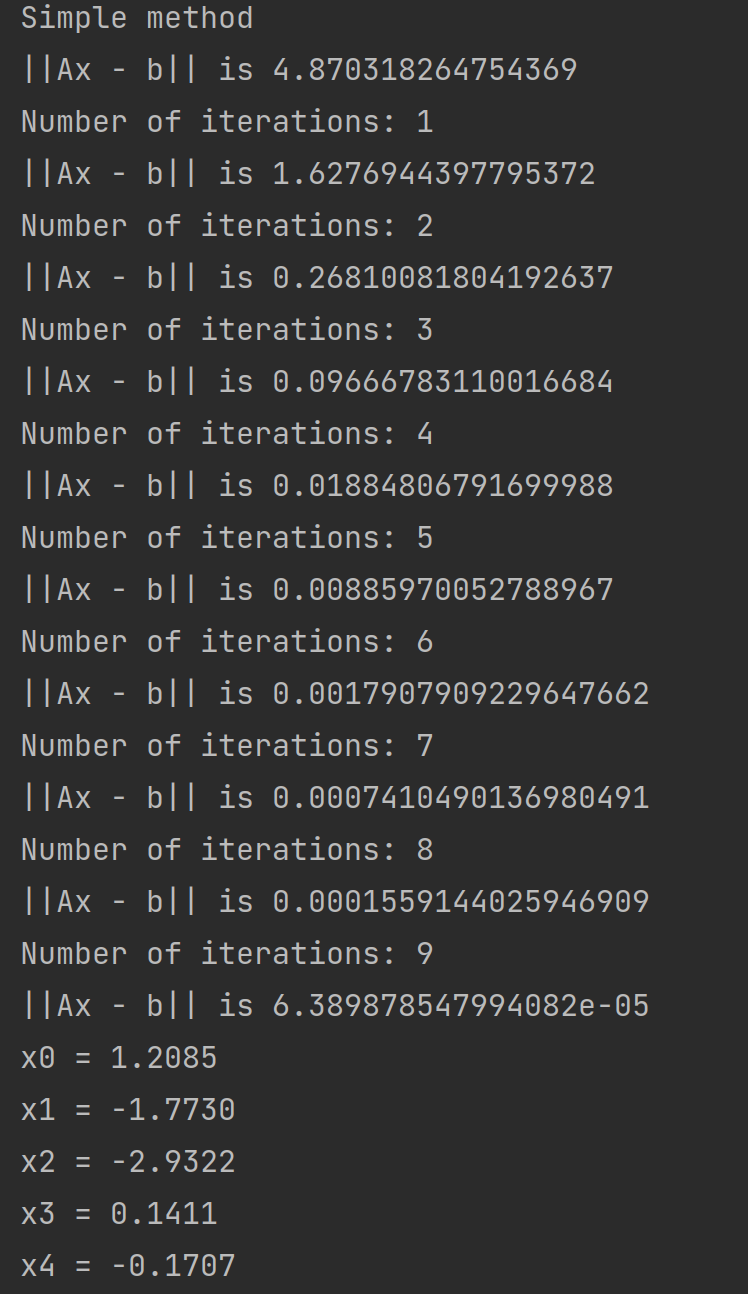
*ajj>a1j+…….+anj для всех j=1,….,n.*

## **Задание**

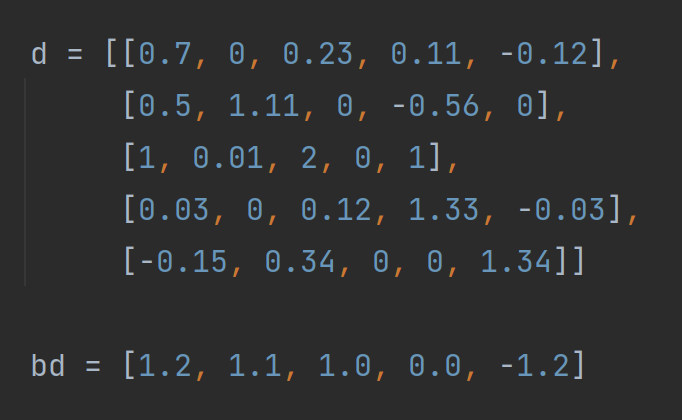
Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение системы ***Ax=b***, где A=kC+D, A-исходная матрица для расчета, k-номер варианта, матрицы C, D и вектор свободных членов и b задаются ниже.

Исходные данные:

Задание лабораторной работы:

Tecт 1:



Simple method

||Ax - b|| is 2.2561028345356955

Number of iterations: 1

||Ax - b|| is 1.2232641781685065

Number of iterations: 2

||Ax - b|| is 0.4667009889003781

Number of iterations: 3

||Ax - b|| is 0.3095530870603618

Number of iterations: 4

||Ax - b|| is 0.06612005306260657

Number of iterations: 5

||Ax - b|| is 0.06067104925567544

Number of iterations: 6

||Ax - b|| is 0.011649240963191884

Number of iterations: 7

||Ax - b|| is 0.01397099232861815

Number of iterations: 8

||Ax - b|| is 0.002181701908812748

Number of iterations: 9

||Ax - b|| is 0.0032066639791223784

Number of iterations: 10

||Ax - b|| is 0.0005124849837033682

Number of iterations: 11

||Ax - b|| is 0.0007632326002524085

Number of iterations: 12

||Ax - b|| is 0.00015135637560155526

Number of iterations: 13

||Ax - b|| is 0.00018716117646757434

Number of iterations: 14

||Ax - b|| is 4.733498939653092e-05

x0 = 1.5505

x1 = 0.2606

x2 = 0.1175

x3 = -0.0634

x4 = -0.7881

Zendel method

||Ax - b|| is 2.2561028345356955

Number of iterations: 1

||Ax - b|| is 0.7595306952617925

Number of iterations: 2

||Ax - b|| is 0.08804850485688666

Number of iterations: 3

||Ax - b|| is 0.025847065094275312

Number of iterations: 4

||Ax - b|| is 0.0072375077535157565

Number of iterations: 5

||Ax - b|| is 0.002154329587809607

Number of iterations: 6

||Ax - b|| is 0.0006381314968396639

Number of iterations: 7

||Ax - b|| is 0.00018900413575525144

Number of iterations: 8

||Ax - b|| is 5.599243489001609e-05

x0 = 1.5506

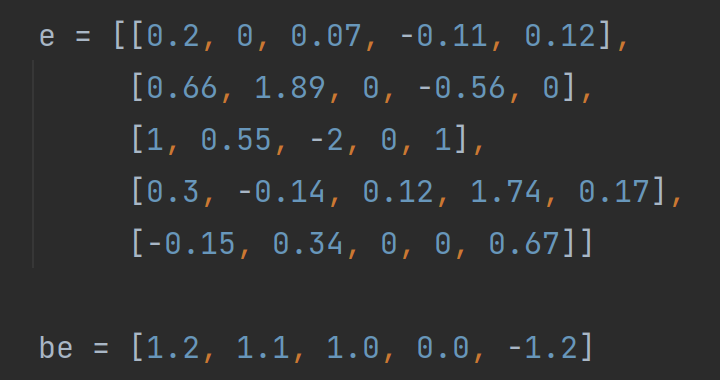
x1 = 0.2606

x2 = 0.1174

x3 = -0.0633

x4 = -0.7881

Tecт 2:



Simple method

||Ax - b|| is 2.2561028345356955

Number of iterations: 1

||Ax - b|| is 6.211509546889524

Number of iterations: 2

||Ax - b|| is 2.2576669709076898

Number of iterations: 3

||Ax - b|| is 1.2675810351332766

Number of iterations: 4

||Ax - b|| is 1.6511062505698562

Number of iterations: 5

||Ax - b|| is 0.24949497495783954

Number of iterations: 6

||Ax - b|| is 0.6797281577957591

Number of iterations: 7

||Ax - b|| is 0.31934401677851804

Number of iterations: 8

||Ax - b|| is 0.1802125345754457

Number of iterations: 9

||Ax - b|| is 0.20324088239055874

Number of iterations: 10

||Ax - b|| is 0.03004588450285538

Number of iterations: 11

||Ax - b|| is 0.08566167060894216

Number of iterations: 12

||Ax - b|| is 0.03863744620266843

Number of iterations: 13

||Ax - b|| is 0.023219925491652176

Number of iterations: 14

||Ax - b|| is 0.02516375563613823

Number of iterations: 15

||Ax - b|| is 0.0036697921020144232

Number of iterations: 16

||Ax - b|| is 0.01077225647346486

Number of iterations: 17

||Ax - b|| is 0.004675785769242145

Number of iterations: 18

||Ax - b|| is 0.0029897316642360614

Number of iterations: 19

||Ax - b|| is 0.0031133775749128774

Number of iterations: 20

||Ax - b|| is 0.00045427269288187593

Number of iterations: 21

||Ax - b|| is 0.0013537289683042781

Number of iterations: 22

||Ax - b|| is 0.0005650599970342904

Number of iterations: 23

||Ax - b|| is 0.00038439205085117673

Number of iterations: 24

||Ax - b|| is 0.0003849383222693914

Number of iterations: 25

||Ax - b|| is 5.7025384534948246e-05

x0 = 4.8604

x1 = -1.4299

x2 = 1.5482

x3 = -1.0620

x4 = 0.0227

Zendel method

||Ax - b|| is 2.2561028345356955

Number of iterations: 1

||Ax - b|| is 0.8616920570546911

Number of iterations: 2

||Ax - b|| is 0.5153425077943603

Number of iterations: 3

||Ax - b|| is 0.22922176643169848

Number of iterations: 4

||Ax - b|| is 0.06872550287972813

Number of iterations: 5

||Ax - b|| is 0.014228067761026442

Number of iterations: 6

||Ax - b|| is 0.0014100216876501565

Number of iterations: 7

||Ax - b|| is 0.000745955439412635

Number of iterations: 8

||Ax - b|| is 0.00042385146457277716

Number of iterations: 9

||Ax - b|| is 0.00014552948629820403

Number of iterations: 10

||Ax - b|| is 3.524991942913205e-05

x0 = 4.8603

x1 = -1.4299

x2 = 1.5483

x3 = -1.0620

x4 = 0.0227

**Листинг кода**

methods.py

def simple(a, b, x):  
 x\_temp = [0, 0, 0, 0, 0]  
 for i in range(5):  
 x\_temp[i] = b[i]  
 for j in range(5):  
 if j != i:  
 x\_temp[i] -= a[i][j] \* x[j]  
 x\_temp[i] /= a[i][i]  
 return x\_temp  
  
  
def zendel(a, b, x):  
 next\_x = [0, 0, 0, 0, 0]  
 for i in range(5):  
 next\_x[i] = b[i]  
 for j in range(5):  
 if j != i:  
 if j < i:  
 next\_x[i] -= a[i][j] \* next\_x[j]  
 else:  
 next\_x[i] -= a[i][j] \* x[j]  
  
 next\_x[i] /= a[i][i]  
 return next\_x  
  
  
def calculate\_norm(a, b, x):  
 norm = 0.0  
 for j in range(0, 5):  
 result = 0.0  
 for i in range(0, 5):  
 result += x[i] \* a[j][i]  
 result -= b[j]  
 norm += result \* result  
  
 norm \*\*= 0.5  
 print(f"||Ax - b|| is {norm}")  
 return norm

main.py

import methods  
import numpy as np  
import copy  
  
  
def solve(method, a, b, x, need\_accuracy=0.0001):  
 iteration\_number = 0  
 if x is None:  
 x = [0, 0, 0, 0, 0]  
  
 while iteration\_number >= 0:  
 next\_x = method(a, b, x)  
 norm = methods.calculate\_norm(a, b, x)  
 iteration\_number += 1  
 x = copy.deepcopy(next\_x)  
 if norm < need\_accuracy:  
 for i in range(5):  
 print(f"x{i} = {x[i]:.4f}")  
 break  
 else:  
 print(f"Number of iterations: {iteration\_number}")  
 print()  
  
  
def main():  
 a = [[1.39, 0.21, 0.05, 0.12, -0.13],  
 [-0.07, -1.27, -0.01, 0.17, 0.12],  
 [0.12, -0.07, -1.27, 0.11, 0.05],  
 [0.17, 0.12, -0.07, -1.27, 0.11],  
 [0.11, 0.67, 0.12, -0.07, -1.27]]  
  
 ba = [1.2, 2.2, 4.0, 0.0, -1.2]  
 d = [[0.7, 0, 0.23, 0.11, -0.12],  
 [0.5, 1.11, 0, -0.56, 0],  
 [1, 0.01, 2, 0, 1],  
 [0.03, 0, 0.12, 1.33, -0.03],  
 [-0.15, 0.34, 0, 0, 1.34]]  
 bd = [1.2, 1.1, 1.0, 0.0, -1.2]  
 e = [[0.2, 0, 0.07, -0.11, 0.12],  
 [0.66, 1.89, 0, -0.56, 0],  
 [1, 0.55, -2, 0, 1],  
 [0.3, -0.14, 0.12, 1.74, 0.17],  
 [-0.15, 0.34, 0, 0, 0.67]]  
 be = [1.2, 1.1, 1.0, 0.0, -1.2]  
 print(np.array(a))  
  
 print("Simple method")  
 solve(methods.simple, a, ba, None)  
 print("Zendel method")  
 solve(methods.zendel, a, ba, None)  
  
 print("Simple method")  
 solve(methods.simple, d, bd, None)  
 print("Zendel method")  
 solve(methods.zendel, d, bd, None)  
  
 print("Simple method")  
 solve(methods.simple, e, be, None)  
  
 print("Zendel method")  
 solve(methods.zendel, e, be, None)  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**ВЫВОД И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ**

Исследуя полученные результаты, можно сразу обратить внимание на то, что число итераций различается, а именно: алгоритму Зейделя потребовалось меньшее число итераций, чтобы решить СЛАУ с определенной точностью.

Метод простых итераций работает таким образом, что для вычисления (k+1)-го приближения всех неизвестных, учитываются вычисленные ранее их

k-е приближения, в то время как итерации по методу Зейделя отличаются от простых итераций тем, что при нахождении (k+1)-го приближения, используются уже найденные компоненты того же (k+1)-го приближения. Вот почему метод Зейделя требуется меньшее число итераций, чтобы прийти к решению СЛАУ.

Методы Зейделя и простых итераций работают не для всех систем. Для обеспечения сходимости этих методов, требуется преобразовать систему так, чтобы диагональные элементы матрицы системы преобладали над остальными. Как правило, метод Зейделя обеспечивает лучшую сходимость, чем метод простых итераций (за счет накопления информации, полученной при решении предыдущих уравнений).