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 АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ (СЛАУ) МЕТОДОМ ПРОСТЫХ ИТЕРАЦИЙ И МЕТОДОМ ЗЕЙДЕЛЯ»**

БГУИР 6-05 0612 02 86

|  |
| --- |
| Выполнил студент группы 353505  МАРТЫНКЕВИЧ Евгений Дмитриевич |

|  |
| --- |
| (дата, подпись студента) |
| Проверил доцент кафедры информатики  АНИСИМОВ Владимир Яковлевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

**Содержание**

1. Цель работы
2. Задание
3. Программная реализация
4. Полученные результаты
5. Оценка полученных результатов
6. Вывод

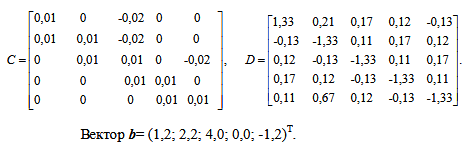
**Цель работы**

* изучить метод простых итераций и метод Зейделя, получить численное решение заданной СЛАУ;
* составить программу решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ;
* выполнить тестовые примеры и проверить правильность работы программы

**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.

Исходные данные:



Вариант 3

**Программная реализация**

Для проверки решения умножим исходную матрицу на полученный вектор решений и сравним с изначальным вектором свободных членов.

*Исходные данные:*

Матрица A, полученная в результате вычисления A = 3C + D:



Код простых итераций:

def iteration\_method():  
 global X, X\_next  
 X = numpy.zeros(len(A))  
 X\_next = numpy.zeros(len(A))  
 check = True  
 amount = 0  
 while check:  
 for i in range(0, len(A)):  
 summ = 0.  
 summ += b[i]  
 for j in range(0, len(A)):  
 if i != j:  
 summ -= A[i][j] \* X[j]  
 X\_next[i] = summ / A[i][i]  
 eps = X\_next - X  
 for i in range(0, len(eps)):  
 eps[i] = abs(eps[i])  
 max\_eps = max(eps)  
 if max\_eps < 10 \*\* (-4):  
 check = False  
 X = X\_next.copy()  
 amount += 1  
 print("Кол-во итераций: ", amount)

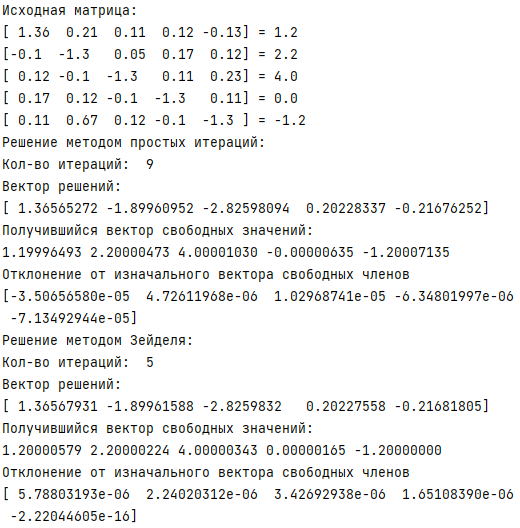
Код метода Зейделя:

def zeidel\_method():  
 global X, X\_next  
 X = numpy.zeros(len(A))  
 check = True  
 amount = 0  
 while check:  
 X\_next = X.copy()  
 for i in range(0, len(A)):  
 summ = 0.  
 summ += b[i]  
 for j in range(0, len(A)):  
 if i != j:  
 summ -= A[i][j] \* X\_next[j]  
 X\_next[i] = summ / A[i][i]  
 eps = X\_next - X  
 for i in range(0, len(eps)):  
 eps[i] = abs(eps[i])  
 max\_eps = max(eps)  
 if max\_eps < 10 \*\* (-4):  
 check = False  
 X = X\_next.copy()  
 amount += 1  
 print("Кол-во итераций: ", amount)

Код проверок для возможности применения данных методов:

def all\_check():  
 return check\_on\_row() or check\_on\_column() or check\_norm()  
  
  
def check\_on\_row():  
 matrix = A.copy()  
 for i in range(0, len(matrix)):  
 summ = 0.  
 for j in range(0, len(matrix)):  
 if i != j:  
 summ += abs(matrix[i][j])  
 if summ > abs(matrix[i][i]):  
 print(f"Сумма модулей по строке {i} ({summ}) больше модуля диагонального элемента {A[i][i]}")  
 return False  
 return True  
  
  
def check\_on\_column():  
 matrix = A.transpose()  
 for i in range(0, len(matrix)):  
 summ = 0.  
 for j in range(0, len(matrix)):  
 if i != j:  
 summ += abs(matrix[i][j])  
 if summ > abs(matrix[i][i]):  
 print(f"Сумма модулей по столбцу {i} ({summ}) больше модуля диагонального элемента {A[i][i]}")  
 return False  
 return True  
  
  
def check\_norm():  
 matrix = A.copy()  
 for i in range(0, len(matrix)):  
 summ = 0.  
 for j in range(0, len(matrix)):  
 if i != j:  
 summ += (matrix[i][j]/matrix[i][i]) \*\* 2  
 if summ > 1:  
 print(f"||B|| больше 1")  
 return False  
 return True

**Полученные результаты**



*Результаты вычислений (вектор решений):*

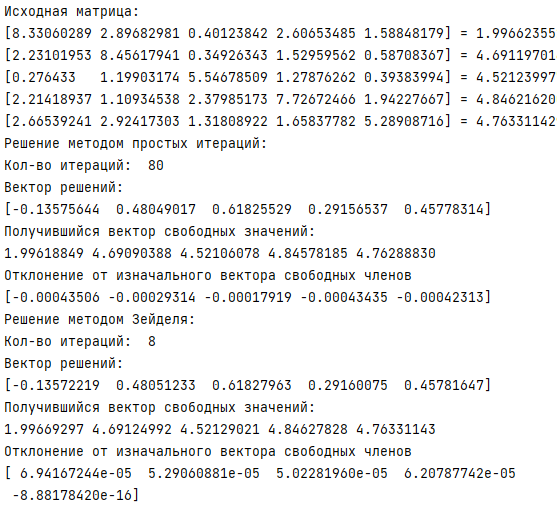
|  |  |  |
| --- | --- | --- |
| *Метод простых итераций* | *Метод Зейделя* | *Кол-во итераций* |
| 1.36565272 | 1.36567931 | Метод простых итераций: 9 |
| -1.89960952 | -1.89961588 | Метод Зейделя: 5 |
| -2.82598094 | -2.82598320 |  |
| 0.20228337 | 0.20227558 |  |
| -0.21676252 | -0.21681805 |  |

*Результаты вычислений (получившийся вектор свободных членов):*

|  |  |
| --- | --- |
| *Метод простых итераций* | *Метод Зейделя* |
| 1.19996493 | 1.20000579 |
| 2.20000473 | 2.20000224 |
| 4.00001030 | 4.00000343 |
| -0.00000635 | 0.00000165 |
| -1.20007135 | -1.20000000 |

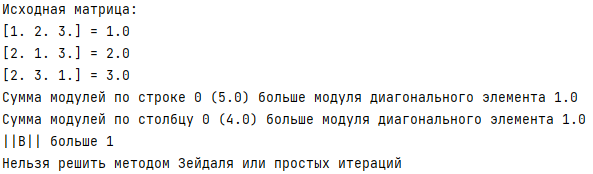
*Тестовый пример 1.*

С помощью пакета numpy создадим матрицу и вектор свободных членов и заполним их случайными числами:

**

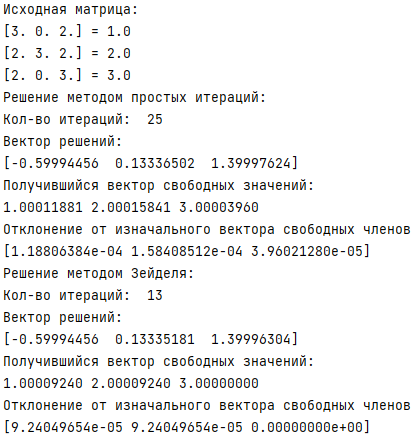
*Тестовый пример 2.*

*В данном примере мы видим матрицу, в которой диагональ не является преимущественной.*

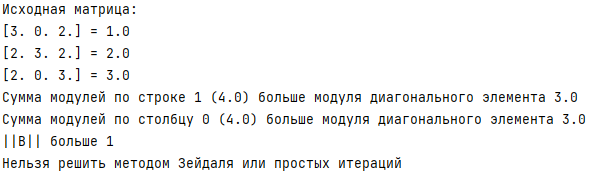


*Тестовый пример 3.*

*Так как проверки являются достаточным, но не необходимыми, то может существовать матрица с не преимущественной диагональю, решаемая с помощью метода простых итераций или метода Зейделя.   
Вывод с отключенными проверками:*

*:*

*Обычный вывод программы:*



**Вывод**

В ходе выполнения лабораторной работы я изучил метод простых итераций и метод Зейделя, написал программу их реализации на языке Python для решения СЛАУ, правильность работы программы проверил на тестовых примерах.

На основании тестов можно сделать следующие выводы:

* Программа позволяет получить решения системы с заданной точностью (заданная точность в условиях лабораторной работы 10^-4);
* Метод Зейделя эффективнее по сравнению с методом простых итераций, так как затрачивает меньшее число итераций;
* Имеет ограничение в использовании (главная диагональ должна быть преимущественной), однако существуют матрицы, которые имеют не преимущественную диагональ и решаются с помощью метода простых итераций или метода Зейделя.
* Метод Зейделя более точен, так как использует уже найденные значения вектора решения на данной итерации, в отличие метода простых итераций.