«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия  
Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №1

Вариант 4

Студент:

*Ильин Н. С.*

*Р3210*

Преподаватель:

*Наумова Н. А.*

Санкт-Петербург, 2025 г.

**Оглавление**

[**Цель работы**: 3](#_Toc191337740)

[**Описание метода**: 3](#_Toc191337741)

[**Формулы**: 3](#_Toc191337742)

[**Листинг программы** 4](#_Toc191337743)

[**Примеры и результаты работы программы** 5](#_Toc191337744)

[**Выводы**: 5](#_Toc191337745)

# **Цель работы**:

Изучить численные методы решения систем линейных алгебраических уравнений и реализовать программу для решения СЛАУ методом простых итераций.

# **Описание метода**:

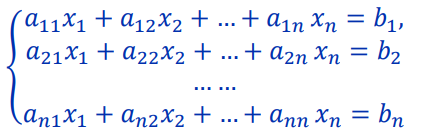
Метод итерации или метод простой итерации — численный метод решения системы линейных алгебраических уравнений. Суть метода заключается в нахождении по приближённому значению величины следующего приближения, являющегося более точным.

Простым языком, основная идея метода заключается в том, что уравнение преобразуется к виду *x*=*g*(*x*), где *g*(*x*) — некоторая функция. Затем мы начинаем с какого-то начального приближения *x*0​ и последовательно вычисляем новые значения по формуле: *xn*+1​=*g*(*xn*​)

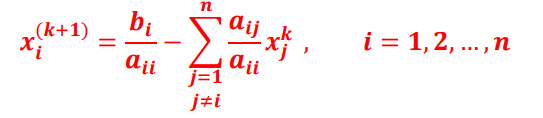
То есть, каждое следующее значение  *xn*+1​ вычисляется на основе предыдущего *xn*​. Этот процесс повторяется до тех пор, пока разница между *xn*+1​ и *xn*​ не станет очень маленькой (меньше заданной точности). Когда это происходит, считается, что мы нашли приближенное решение уравнения.

# **Формулы**:

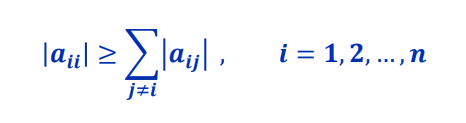
Общий вид СЛАУ:



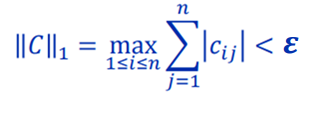
Рабочая формула метода простой итерации:



Достаточное условие сходимости итерационного процесса:

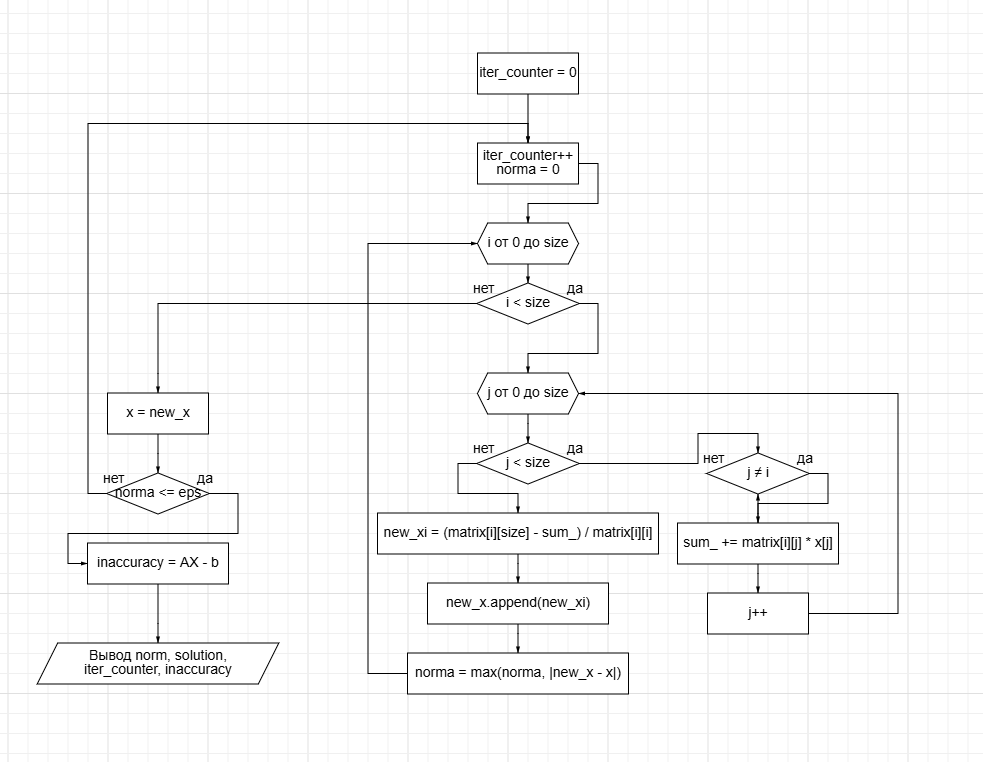


Норма матрицы по строкам:



Вектор невязки:  
, где x\* - вычисленное решение

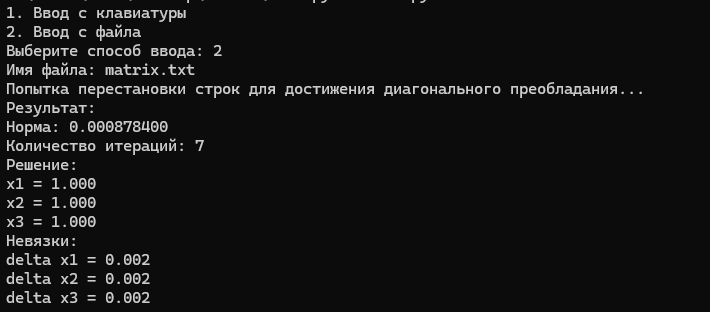
# **Блок схема реализации численного метода**

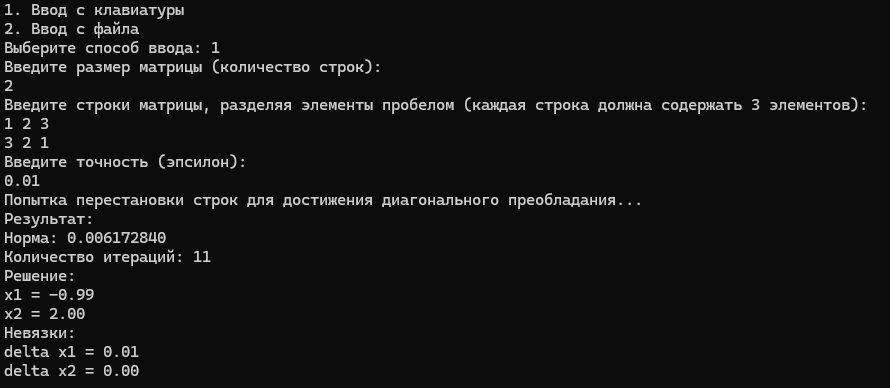


# **Листинг программы**

|  |
| --- |
| def solve(matrix, eps):  size = len(matrix)  for i in range(size):  if matrix[i][i] == 0:  print("Ошибка: нулевой элемент на диагонали. Невозможно решить.")  return None    x = [0.0] \* size  iter\_counter = 0  max\_iterations = 1000    eps\_str = f"{eps:.10f}".rstrip('0').rstrip('.') if 'e' not in str(eps) else str(eps)  if 'e' in eps\_str:  if 'e-' in eps\_str:  decimal\_places = int(eps\_str.split('e-')[1])  else:  decimal\_places = 0  elif '.' in eps\_str:  decimal\_places = len(eps\_str.split('.')[1])  else:  decimal\_places = 0    while True:  iter\_counter += 1  new\_x = []  norma = 0.0  for i in range(size):  sum\_ = sum(matrix[i][j] \* x[j] for j in range(size) if j != i)  new\_xi = (matrix[i][size] - sum\_) / matrix[i][i]  new\_x.append(new\_xi)  norma = max(norma, abs(new\_xi - x[i]))  x = new\_x  if norma <= eps:  break  if iter\_counter >= max\_iterations:  print("Метод не сошелся за максимальное количество итераций.")  return None    format\_str = f"{{:.{decimal\_places}f}}"    result = {  'norm': f"{norma:.9f}",  'iterations': iter\_counter,  'solution': [{'index': i+1, 'value': format\_str.format(x[i])} for i in range(size)],  'inaccuracy': [{'index': i+1, 'value': format\_str.format(sum(matrix[i][j] \* x[j] for j in range(size)) - matrix[i][size])} for i in range(size)]  }  return result |

## **Примеры и результаты работы программы**





# **Выводы**:

В результате выполнения данной лабораторной работой я познакомился с численными методами решения математических задач на примере СЛАУ. Используя Python, реализовал программу для решения СЛАУ методом простых итераций.