Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №1**

по «Вычислительная математика»

Выполнил:

Студент группы P3207

Алферов Глеб Александрович

Вариант: 1

Преподаватели:

Рыбаков С.Д.

Санкт-Петербург

2025

**1. Цель работы**

Цель данной работы – изучение и реализация метода простых итераций для решения систем линейных алгебраических уравнений (СЛАУ). Также проверяется возможность приведения матрицы к диагональному преобладанию и оценивается сходимость метода.

**2. Описание метода, расчетные формулы**

Метод простых итераций используется для решения системы уравнений вида:

Ax = b

Преобразуем систему в итерационный вид:

x= Cx + d

где:

C=I−D-1 \* A

d = D-1 \* b

Здесь D – диагональная матрица, содержащая диагональные элементы A, а I – единичная матрица.

Метод сходится, если выполнено условие:

∣∣C∣∣1 <1

Итерационный процесс:

x(k+1) = Cx(k) + d

Повторяется, пока разность между последовательными итерациями не станет меньше ϵ.

**3. Листинг программы**

Программа реализована на Python.

import sys  
  
  
def from\_keyboard():  
 while True:  
 try:  
 n = int(input("Размерность системы: "))  
 e = float(input("Введите точность системы: "))  
 if 1 <= n <= 20:  
 break  
 else:  
 print("Размерность должна быть целым числом от 1 до 20")  
 if e < 0:  
 break  
 else:  
 print("Точность не может быть меньше 0")  
 except ValueError:  
 print("Введите целое число!")  
 except EOFError:  
 print("Завершаем выполнение программы.")  
 sys.exit(0)  
  
 A = []  
 d = []  
 print("Введите коэффициенты A для уравнений и свободный член b")  
 print("Формат ввода: A[1] A[2] ... A[n] b")  
 for i in range(n):  
 while True:  
 try:  
 data = list(map(float, input(f"Уравнение {i + 1}: ").split()))  
 if len(data) != n + 1:  
 print(f"Нужно {n + 1} чисел. Повторите ввод")  
 else:  
 A.append(data[:n])  
 d.append(data[-1])  
 break  
 except ValueError:  
 print("Ошибка ввода. Повторите строку")  
 except EOFError:  
 print("Завершаем выполнение программы.")  
 sys.exit(0)  
 return n, e, A, d  
  
  
def from\_file(filename):  
 try:  
 with open(filename, "r", encoding="utf-8") as f:  
 lines = [line.strip() for line in f if line.strip()]  
 n = int(lines[0])  
 try:  
 e = float(lines[1])  
 except ValueError:  
 print("Ошибка при вводе формата float поменяйте , на .")  
 sys.exit(1)  
 if n < 1 or n > 20:  
 print("Размерность должна быть целым числом от 1 до 20")  
 sys.exit(1)  
 if e < 0:  
 print("Точность не может быть меньше 0")  
 sys.exit(1)  
 A = []  
 d = []  
 if len(lines) < n + 2:  
 print("В файле недостаточно строк")  
 sys.exit(1)  
 for i in range(2, n + 2):  
 data = list(map(float, lines[i].split()))  
 if len(data) != n + 1:  
 print(f"Ошибка в строке {i + 1}, ожидается {n + 1} чисел, получено {len(data)}")  
 sys.exit(1)  
 A.append(data[:n])  
 d.append(data[-1])  
 return n, e, A, d  
 except Exception as e:  
 print("Ошибка при чтении файла: ", e)  
 sys.exit(1)  
  
  
def check\_diagonal(matrix, n):  
 for i in range(n):  
 row\_sum = sum(abs(matrix[i][j]) for j in range(n)) - abs(matrix[i][i])  
 if row\_sum >= abs(matrix[i][i]):  
 return False  
 return True  
  
  
def permute\_rows(matrix, index, n):  
 if index >= len(matrix) - 1:  
 if check\_diagonal(matrix, n):  
 return [rows[:] for rows in matrix]  
 else:  
 return None  
  
 for i in range(index, len(matrix)):  
 matrix[index], matrix[i] = matrix[i], matrix[index]  
 solution = permute\_rows(matrix, index + 1, n)  
 if solution is not None:  
 return solution  
 matrix[index], matrix[i] = matrix[i], matrix[index]  
 return None  
  
  
def express\_vars(matrix, n):  
 B = []  
 d = []  
 for i in range(n):  
 diag = matrix[i][i]  
 if diag == 0:  
 raise ValueError("Diagonal element is zero")  
 new\_row = []  
 for j in range(n):  
 if j == i:  
 new\_row.append(0)  
 else:  
 new\_row.append(- matrix[i][j] / diag)  
 d.append(matrix[i][n] / diag)  
 B.append(new\_row)  
 norm = max(sum(abs(x) for x in row[:n]) for row in B)  
 print("Бесконечная норма преобразованной матрицы -", norm, "\n")  
 return B, d  
  
  
def simply\_iter(matrix, initial, n, e):  
 x = []  
 for i in range(len(initial)):  
 x.append(initial[i])  
  
 for k in range(1000):  
 new\_x = [0] \* n  
 for i in range(n):  
 s = 0  
 for j in range(n):  
 s += matrix[i][j] \* x[j]  
 new\_x[i] = initial[i] + s  
  
 error\_vector = []  
 for i in range(n):  
 diff = new\_x[i] - x[i]  
 if diff < 0:  
 diff = -diff  
 error\_vector.append(diff)  
  
 max\_error = error\_vector[0]  
 for i in range(1, n):  
 if error\_vector[i] > max\_error:  
 max\_error = error\_vector[i]  
  
 if max\_error < e:  
 print("Итерация", k + 1, "x =", new\_x, "\n", "Вектор погрешностей =", error\_vector, max\_error, "<", e)  
 print("Сходимость достигнута! Всего итераций:", k + 1)  
 return new\_x  
 else:  
 print("Итерация", k + 1, "x =", new\_x, "\n", "Вектор погрешностей =", error\_vector, max\_error, ">", e)  
 x = new\_x[:]  
 return x  
  
print("Размерность матрицы должна быть <= 20\n"  
 "Первая строка файла должна быть размерностью\n"  
 "Вторая строка файла должна быть точностью\n"  
 "Все последующие n строк должны быть в формате: A[1] A[2] A[3] ... A[n] b\n"  
 "Где A коэффициент перед xn, b - свободный член\n")  
  
print("Выберите способ ввода данных:")  
print("1 - с клавиатуры")  
print("2 - файла")  
  
try:  
 choice = input().strip()  
  
 if choice == "1":  
 n, e, A, d = from\_keyboard()  
  
 full\_matrix = [A[i] + [d[i]] for i in range(n)]  
 sol = permute\_rows(full\_matrix, 0, n)  
  
 if sol is not None:  
 print("\nПерестановка найдена")  
 for row in sol:  
 print(row)  
 after\_exp, initial\_app = express\_vars(sol, n)  
 result = simply\_iter(after\_exp, initial\_app, n, e)  
 print("Приближенное решение:", result)  
 else:  
 print("Диагональное преобладание невозможно")  
 print("\n")  
  
 elif choice == "2":  
 print("Введите имя файла")  
 filename = input().strip()  
 n, e, A, d = from\_file(filename)  
  
 full\_matrix = [A[i] + [d[i]] for i in range(n)]  
 sol = permute\_rows(full\_matrix, 0, n)  
  
 if sol is not None:  
 print("\nПерестановка найдена")  
 for row in sol:  
 print(row)  
 after\_exp, initial\_app = express\_vars(sol, n)  
 result = simply\_iter(after\_exp, initial\_app, n, e)  
 print("Приближенное решение:", result)  
 else:  
 print("Диагональное преобладание невозможно")  
 else:  
 print("Неизвестная команда, введите 1 или 2")  
except EOFError:  
 print("Завершаем выполнение программы.")  
 sys.exit(0)

Основные части кода:

* Ввод данных вручную или из файла.
* Проверка диагонального преобладания матрицы и перестановка строк при необходимости.
* Вычисление C и d, а также нормы
* Итерационный процесс с отслеживанием погрешности на каждом шаге.

**4. Примеры и результаты работы программы**

**Пример входных данных:**

3

0.01

2 2 10 14

10 1 1 12

2 10 1 13

**Выходные данные:**

Бесконечная норма преобразованной матрицы - 0.4

Итерация 1 x = [0.9299999999999999, 0.92, 0.8999999999999999]

Вектор погрешностей = [0.27, 0.38, 0.5] 0.5 > 0.01

Итерация 2 x = [1.018, 1.024, 1.0299999999999998]

Вектор погрешностей = [0.08800000000000008, 0.10399999999999998, 0.1299999999999999] 0.1299999999999999 > 0.01

Итерация 3 x = [0.9945999999999999, 0.9934000000000001, 0.9915999999999999]

Вектор погрешностей = [0.023400000000000087, 0.03059999999999996, 0.03839999999999988] 0.03839999999999988 > 0.01

Итерация 4 x = [1.0015, 1.0019200000000001, 1.0024]

Вектор погрешностей = [0.006900000000000128, 0.008520000000000083, 0.010800000000000032] 0.010800000000000032 > 0.01

Итерация 5 x = [0.999568, 0.99946, 0.9993159999999999]

Вектор погрешностей = [0.0019320000000000448, 0.0024600000000001288, 0.0030840000000000867] 0.0030840000000000867 < 0.01

Сходимость достигнута! Всего итераций: 5

Приближенное решение: [0.999568, 0.99946, 0.9993159999999999]

**5. Выводы**

В ходе выполнения данной работы была изучена и реализована численная методика решения систем линейных алгебраических уравнений (СЛАУ) методом простых итераций. Реализация программы позволила автоматизировать процесс проверки диагонального преобладания матрицы, выполнения условий сходимости и нахождения решения.

В результате работы были закреплены навыки работы с методами численного анализа, алгоритмами обработки матриц и написания программ на Python. Проведенные тесты подтвердили корректность работы программы и показали, что метод простых итераций является эффективным при выполнении условий сходимости.