Федеральное государственное автономное образовательное учреждение высшего образования Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №1

Вариант № 1

Выполнил: студент группы P3208, Васильев Н. А.

Преподаватель: Машина Е.А.

Санкт-Петербург 2025

# Текст задания

Решение системы линейных алгебраических уравнений СЛАУ методом простых итераций.

В программе должно быть реализовано:

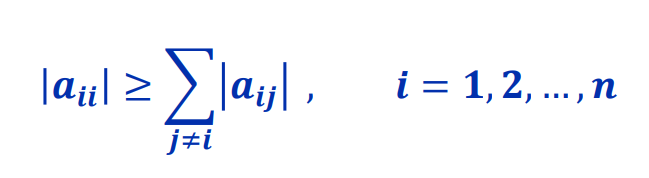
* Точность задается с клавиатуры/файла,
* Проверка диагонального преобладания (в случае, если диагональное
* преобладание в исходной матрице отсутствует, сделать перестановку
* строк/столбцов до тех пор, пока преобладание не будет достигнуто). В
* случае невозможности достижения диагонального преобладания - выводить соответствующее сообщение.
* Вывод нормы матрицы (любой, на Ваш выбор),
* Вывод вектора неизвестных: ,
* Вывод количества итераций, за которое было найдено решение,
* Вывод вектора погрешностей: .

Цель работы

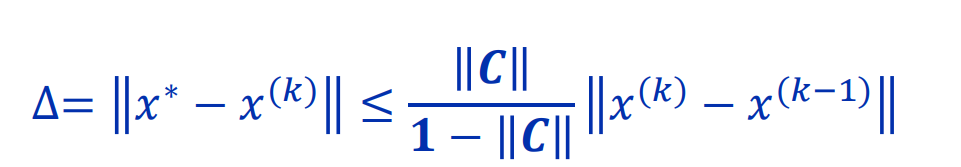
Решение системы линейных алгебраических уравнений СЛАУ методом простых итераций.

Описание метода, расчётные формулы



Диагональное доминирование:  


Норма матрицы:

Погрешность:  


Листинг программы

import csv  
import numpy as np  
import random  
  
  
def generate\_random\_matrix(min\_val=-25, max\_val=25):  
 size = random.randint(2, 20)  
 matrix = [[random.uniform(min\_val, max\_val) for \_ in range(size)] for \_ in range(size)]  
 vector = [random.uniform(min\_val, max\_val) for \_ in range(size)]  
  
 for i in range(size):  
 matrix[i].append(vector[i])  
  
 return matrix  
  
  
def read\_matrix\_from\_file(filename):  
 matrix = []  
 try:  
 with open(filename, newline='') as file:  
 if filename.endswith('.csv'):  
 reader = csv.reader(file)  
 rows = [row for row in reader]  
 else:  
 rows = [line.split() for line in file]  
  
 for row in rows:  
 if not all(num.replace('.', '', 1).replace('-', '', 1).isdigit() for num in row):  
 raise ValueError("Ошибка: все элементы должны быть вещественными числами")  
 matrix.append([float(num) for num in row])  
 return matrix  
 except FileNotFoundError:  
 print("Такой файл не найден. Попробуйте снова.")  
 return read\_matrix\_from\_file(input())  
  
  
def validate\_matrix(matrix):  
 n = len(matrix)  
 for row in matrix:  
 if len(row) != n + 1:  
 raise ValueError("Ошибка: каждая строка должна содержать n элементов матрицы и один элемент вектора")  
  
  
def diagonal\_dominance(matrix):  
 matrix = np.array(matrix)  
 n = len(matrix)  
 for i in range(n):  
 row\_sum = sum(abs(matrix[i, j]) for j in range(n) if j != i)  
 if abs(matrix[i, i]) <= row\_sum:  
 return False  
 return True  
  
  
def rearrange\_matrix(matrix, vector):  
 n = len(matrix)  
 matrix = np.array(matrix, dtype=float)  
 vector = np.array(vector, dtype=float)  
 for i in range(n):  
 max\_index = max(range(i, n),  
 key=lambda k: abs(matrix[k, i]) - sum(abs(matrix[k, j]) for j in range(n) if j != i))  
 if max\_index != i:  
 matrix[[i, max\_index]] = matrix[[max\_index, i]]  
 vector[i], vector[max\_index] = vector[max\_index], vector[i]  
 return matrix, vector  
  
  
def norm\_matrix(matrix):  
 return max(sum(abs(element) for element in row) for row in matrix)  
  
  
def iteration(matrix, x, vector):  
 n = len(matrix)  
 x\_new = np.zeros(n)  
 for i in range(n):  
 sum\_ax = sum(matrix[i][j] \* x[j] for j in range(n) if j != i)  
 x\_new[i] = (vector[i] - sum\_ax) / matrix[i][i]  
 return x\_new  
  
  
def iterative\_method(matrix, vector, precision, max\_iterations=1000):  
 n = len(matrix)  
 x = np.zeros(n)  
 iterations = 0  
 errors = []  
  
 while iterations <= max\_iterations:  
 x\_new = iteration(matrix, x, vector)  
 errors.append(np.abs(x\_new - x))  
  
 iterations += 1  
  
 if np.max(np.abs(x\_new - x)) < precision:  
 return x\_new, iterations, errors  
  
 x = x\_new  
  
 print("Метод не сошёлся за", max\_iterations, "итераций")  
 return x, max\_iterations, errors  
  
  
def calculate\_matrix(matrix, precision):  
 validate\_matrix(matrix)  
 matrix = np.array(matrix, dtype=float)  
 vector = matrix[:, -1]  
 matrix = matrix[:, :-1]  
  
 if not diagonal\_dominance(matrix):  
 matrix, vector = rearrange\_matrix(matrix, vector)  
 if not diagonal\_dominance(matrix):  
 print("Ошибка: матрица не является диагонально доминирующей и не может быть преобразована")  
 return  
  
 norm = norm\_matrix(matrix)  
 x, iterations, errors = iterative\_method(matrix, vector, precision)  
  
 print("Матрица после преобразований: ")  
 for i in range(len(matrix)):  
 print([float('%.2f' % elem) for elem in matrix[i]])  
 print("Вектор свободных членов после преобразований: ", vector)  
 print("Норма матрицы: ", norm)  
 print("Вектор неизвестных: ", x)  
 print("Количество итераций: ", iterations)  
 print("Погрешность: ", errors[-1] if errors else None)  
  
  
def is\_valid\_number(value):  
 try:  
 float(value)  
 return True  
 except ValueError:  
 return False  
  
  
def main():  
 print("Решение системы линейных алгебраических уравнений СЛАУ")  
 while True:  
 try:  
 print("Введите точность:")  
 precision = float(input())  
 if precision <= 0:  
 raise ValueError("Ошибка: точность должна быть положительным числом.")  
 break  
 except ValueError:  
 print("Ошибка: введите корректное положительное число для точности.")  
 print(  
 "Введите матрицу в формате 'a11 a12 a13 ... a1n b1; a21 a22 a23 ... a2n b2; ... a3n1 an2 an3 ... ann bn' или введите имя файла с матрицей (.txt или .csv)")  
 print("Матрица должна быть квадратной. n <= 20")  
 print("Введите имя файла или первую строку матрицы: (Для генерации случайной матрицы введите RANDOM)")  
 while True:  
 try:  
 input\_data = input().strip()  
 if input\_data.endswith('.csv') or input\_data.endswith('.txt'):  
 matrix = read\_matrix\_from\_file(input\_data)  
 elif input\_data == 'RANDOM':  
 matrix = generate\_random\_matrix()  
 print("Сгенерированная матрица:")  
 for i in range(len(matrix)):  
 print([float('%.2f' % elem) for elem in matrix[i]])  
 else:  
 matrix = [[float(num) for num in input\_data.split()]]  
 if len(matrix[0]) > 21:  
 raise ValueError("Ошибка: размерность матрицы должна быть не больше 20.")  
 current\_row = 1  
 while current\_row < len(matrix[0]) - 1:  
 input\_data = input().strip()  
 row = input\_data.split()  
 if len(row) != len(matrix[0]) or not all(is\_valid\_number(num) for num in row):  
 print(f"Ошибка: строка должна содержать {len(matrix[0])} вещественных чисел.")  
 continue  
 matrix.append([float(num) for num in row])  
 current\_row += 1  
 break  
 except ValueError as e:  
 print(f"Ошибка: {e}. Попробуйте снова.")  
 try:  
 calculate\_matrix(matrix, precision)  
 except ValueError as e:  
 print(e)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Примеры и результаты работы программы

**Пример №1**. Ввод матрицы 3\*3 с точностью 0,01:

12 1 3 1

0 0 5 3

1 6 2 2

**Вывод:**

Матрица после преобразований:

[12.0, 1.0, 3.0]

[1.0, 6.0, 2.0]

[0.0, 0.0, 5.0]

Вектор свободных членов после преобразований: [1. 2. 3.]

Норма матрицы: 16.0

Вектор неизвестных: [-0.07908951 0.1461034 0.6 ]

Количество итераций: 4

Погрешность: [0.00246914 0.00297068 0. ]

**Пример №2**. Ввод дробных и отрицательных чисел с точностью 1:

1.3 -2.1 4 3

-8.3 -1.1 3 3

2.1 5.5 0.7 2.2

**Вывод:**

Матрица после преобразований:

[-8.3, -1.1, 3.0]

[2.1, 5.5, 0.7]

[1.3, -2.1, 4.0]

Вектор свободных членов после преобразований: [3. 2.2 3. ]

Норма матрицы: 12.4

Вектор неизвестных: [-0.36144578 0.4 0.75 ]

Количество итераций: 1

Погрешность: [0.36144578 0.4 0.75 ]

**Пример №3**. Ввод матрицы, которую нельзя диагонализировать:

1 1 1 2

2 4 2 2

1 1 1 1

**Вывод:**

Ошибка: матрица не является диагонально доминирующей и не может быть преобразована

**Пример №4**. Ввод матрицы, из файла:

matrix.txt

Содержание файла:

1 22 1 3

12 3 4 2

1 3 10 3

**Вывод:**

Матрица после преобразований:

[12.0, 3.0, 4.0]

[1.0, 22.0, 1.0]

[1.0, 3.0, 10.0]

Вектор свободных членов после преобразований: [2. 3. 3.]

Норма матрицы: 24.0

Вектор неизвестных: [0.16666667 0.13636364 0.3 ]

Количество итераций: 1

Погрешность: [0.16666667 0.13636364 0.3 ]

**Пример №5**. Случайная матрица:

Сгенерированная матрица:

[15.47, -18.42, -13.58, 17.51, -4.01, -17.05, 1.73, 15.2, 20.22, 6.1]

[-4.49, -23.7, 7.38, -24.66, 16.33, -23.35, -20.7, 14.2, -15.03, -19.88]

[8.66, -12.81, -16.26, -5.27, 0.6, 24.57, 15.58, 1.75, 7.06, 3.55]

[21.72, 7.84, 12.2, 1.51, 3.44, 3.44, 1.0, 20.59, -7.8, 18.36]

[5.03, 10.64, -24.3, 5.97, -20.43, -5.95, -24.67, -18.57, -19.05, 3.39]

[5.87, -14.11, -10.43, -19.42, -3.26, 17.02, 13.46, -3.72, -7.5, 16.25]

[16.94, 18.95, 24.99, -5.38, -22.26, 9.0, 0.52, -1.17, -21.26, 6.63]

[14.56, -11.44, -24.71, -23.42, 23.69, 23.63, 13.51, 8.18, -2.69, -15.23]

[-21.04, 18.88, -23.15, 8.39, -18.25, -18.34, -6.65, -15.25, 0.37, 2.78]

Ошибка: матрица не является диагонально доминирующей и не может быть преобразована

Вывод

В ходе выполнения данной работы была реализована программа для решения системы линейных алгебраических уравнений (СЛАУ) с использованием итерационных методов.

Реализована возможность ввода коэффициентов матрицы как с клавиатуры, так и из файла (форматы .txt, .csv). Пользователь может задать точность вычислений вручную.