МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

Лабораторная работа №1 по дисциплине

Вычислительная математика

Вариант №20

*Выполнил:*

Сущенко Роман

P32131

*Преподаватель:*

Бострикова Дарья Константиновна

Санкт-Петербург, 2023

**Цель работы**

Изучить численные методы решения систем линейных алгебраических уравнений и реализовать один из них средствами программирования.

# Задание:

Реализовать метод Гаусса-Зейделя. Требования:

1. Точность задается с клавиатуры/файла
2. Проверка диагонального преобладания (в случае, если диагональное преобладание в исходной матрице отсутствует, сделать перестановку строк/столбцов до тех пор, пока преобладание не булет достигнуто). В случае невозможности достижения диагонального преобладания - выводить соответствующее сообщение.
3. Вывод вектора неизвестных: 𝑥1, 𝑥2, … 𝑥𝑛
4. Вывод количества итераций, за которое было найдено решение.
5. Вывод вектора погрешностей: [𝑥(𝑘) − 𝑥(𝑘−1)]

# Описание метода

Метод Гаусса-Зейделя является модификацией метода простой итерации и обеспечивает более быструю сходимость к решению системы уравнений. Идея метода: при вычислении

компонента 𝑥(𝑘+1) вектора неизвестных на (k+1)-й итерации используются 𝑥(𝑘+1), 𝑥(𝑘+1),

𝑖 1 2

… , 𝑥(𝑘+1), уже вычисленные на (k+1)-й итерации. Значения остальных компонент 𝑥(𝑘+1),

𝑛

𝑥(𝑘+1), 𝑥(𝑘+1) берутся из

𝑖+1 𝑛

предыдущей итерации.

𝑖+1

# Листинг программы

*import output\_printer*

*def is\_matrix\_diagonalized(matrix: list[list[float]]) -> bool:*

*for i, matrix\_row in enumerate(matrix):*

*row\_sum = 0*

*for j, matrix\_value in enumerate(matrix\_row[:-1]):*

*if i != j:*

*row\_sum += abs(matrix\_value)*

*if abs(matrix\_row[i]) < row\_sum:*

*return False*

*return True*

*def diagonalize\_matrix(matrix: list[list[float]], matrix\_size: int) -> None:*

*if is\_matrix\_diagonalized(matrix):*

*return*

*possible\_row\_locations = [[] for \_ in range(matrix\_size)]*

*for i, matrix\_row in enumerate(matrix):*

*row\_sum = sum([abs(value) for value in matrix\_row[:-1]])*

*for j, row\_value in enumerate(matrix\_row[:-1]):*

*if abs(row\_value) > row\_sum - abs(row\_value):*

*possible\_row\_locations[j].append(i)*

*# print(possible\_row\_locations)*

*new\_matrix = [[] for \_ in range(matrix\_size)]*

*if try\_to\_place\_rows(possible\_row\_locations, 0, matrix, new\_matrix):*

*matrix[::] = new\_matrix[::]*

*def try\_to\_place\_rows(possible\_row\_locations: list[list[int]], current\_index: int, old\_matrix: list[list[float]], new\_matrix: list[list[float]]) -> bool:*

*if current\_index == len(possible\_row\_locations):*

*return True*

*for possible\_row\_location in possible\_row\_locations[current\_index]:*

*if len(new\_matrix[current\_index]) == 0:*

*new\_matrix[current\_index] = old\_matrix[possible\_row\_location]*

*if try\_to\_place\_rows(possible\_row\_locations, current\_index + 1, old\_matrix, new\_matrix):*

*return True*

*return False*

*def solve\_matrix\_by\_gauss\_seidel\_method(matrix: list[list[float]], matrix\_size: int, accuracy: float) -> list[float]:*

*diagonalize\_matrix(matrix=matrix, matrix\_size=matrix\_size)*

*if is\_matrix\_diagonalized(matrix):*

*print("Условие преобладания диагональных элементов достигнуто: ")*

*output\_printer.print\_matrix(matrix)*

*else:*

*print("Условие преобладания диагональных элементов не достигнуто")*

*max\_iterations = 100*

*current\_iteration = 0*

*c = [[-1 \* row\_value / matrix\_row[i] if j != i else 0 for j, row\_value in enumerate(matrix\_row[:-1]) ] for i, matrix\_row in enumerate(matrix)]*

*d = [matrix\_row[-1] / matrix\_row[i] for i, matrix\_row in enumerate(matrix)]*

*print("Вектор неизвестных:")*

*print(c)*

*print(d)*

*x\_k = [value for value in d]*

*# output\_printer.print\_iteration\_values\_accuracy(current\_iteration=current\_iteration, values=x\_k)*

*accuracy\_vector = []*

*max\_accuracy = 100*

*while max\_accuracy > accuracy and current\_iteration < max\_iterations:*

*current\_iteration += 1*

*next\_x\_k = []*

*for i in range(matrix\_size):*

*next\_x\_k\_value = d[i]*

*for j in range(i):*

*next\_x\_k\_value += c[i][j] \* next\_x\_k[j]*

*for j in range(i, matrix\_size):*

*next\_x\_k\_value += c[i][j] \* x\_k[j]*

*next\_x\_k.append(next\_x\_k\_value)*

*accuracy\_vector = [abs(next\_x\_k[i] - x\_k[i]) for i in range(matrix\_size)]*

*max\_accuracy = round(max(accuracy\_vector), 5)*

*x\_k = [round(value, 5) for value in next\_x\_k]*

*# output\_printer.print\_iteration\_values\_accuracy(current\_iteration=current\_iteration, values=x\_k, accuracy=max\_accuracy)*

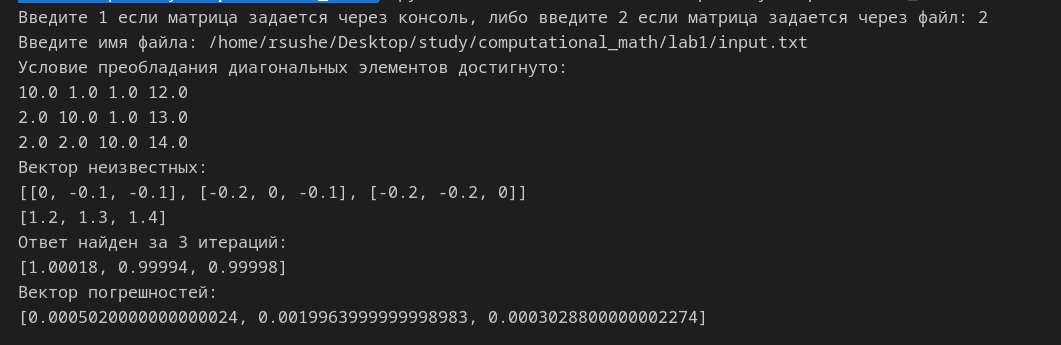
*print("Ответ найден за {} итераций: ".format(current\_iteration))*

*print(x\_k)*

*print("Вектор погрешностей:")*

*print(accuracy\_vector)*

**Примеры работы программы**

****

# Вывод:

В результате выполнения данной лабораторной работы я познакомился с численными методами решения математических задач на примере систем алгебраических уравнений, реализовав на языке программирования Python метод Гаусса-Зейделя.