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: $x_1, x_2, \dots x_n$
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 $x_i^{(k+1)}$ вектора неизвестных на (k+1)-й итерации используются $x_1^{(k+1)}$, $x_2^{(k+1)}$, … , $x_n^{(k+1)}$, уже вычисленные на (k+1)-й итерации. Значения остальных компонент $x_{i+1}^{(k+1)}$, $x_{i+1}^{(k+1)}$, $x_n^{(k+1)}$ берутся из предыдущей итерации.

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

```
Введите 1 если матрица задается через консоль, либо введите 2 если матрица задается через файл: 2
Введите имя файла: /home/rsushe/Desktop/study/computational_math/lab1/input.txt
Условие преобладания диагональных элементов достигнуто:
10.0 1.0 1.0 12.0
2.0 10.0 1.0 13.0
2.0 2.0 10.0 14.0
Вектор неизвестных:
[[0, -0.1, -0.1], [-0.2, 0, -0.1], [-0.2, -0.2, 0]]
[1.2, 1.3, 1.4]
Ответ найден за 3 итераций:
[1.00018, 0.99994, 0.99998]
Вектор погрешностей:
[0.0005020000000000024, 0.0019963999999998983, 0.0003028800000002274]
```

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.