Санкт-Петербургский Государственный университет ИТМО
Факультет програмной инженерии и компьютерной техники

Лабораторная работа №1 по дисциплине "Вычислительная математика"

Решение системы линейных алгебраических уравнений

Вариант №7

Работу выполнила:
Д. А. Карасева
Группа: P3217
Преподаватель:
Т. А. Малышева

Санкт-Петербург
2024

Содержание

1. Цель работы

Изучить численные методы решения систем линейных алгебраических уравнений. Программно реализовать *метод простых итераций.*

Метод должен быть реализован в виде отдельной подпрограммы/метода/класса, в который исходные/выходные данные передаются в качестве параметров. Размерность матрицы n<=20 (задается из файла или с клавиатуры - по выбору конечного пользователя). Должна быть реализована возможность ввода коэффициентов матрицы, как с клавиатуры, так и из файла (по выбору конечного пользователя).

Для *итерационного* метода 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 вектора неизвестных: $x_1, x_2, ..., x_n$.

• Вывод количества итераций, за которое было найдено решение.

• Вывод вектора погрешностей: $|x_i^{(k)} - x_i^{(k-1)}|$

2. Описание метода. Расчетные формулы

Идея итерационных методов решения системы уравнений A X = B состоит в преобразовании ее к виду

$$X = \alpha X + \beta.$$

с последующим использованием сходящегося итерационного процесса

$$X^{(k+1)} = \alpha X^{(k)} + \beta, k = 0, 1, 2, ...,$$

где начальное приближение $X^{(0)}$ выбирается произвольно, например, равным нулю или оценкам, найденным другими методами. Процесс итераций заканчивается обнаружением близости очередных приближений.

Достаточным условием сходимости итерационного процесса является требование:

$$||\alpha|| <= K < 1$$

Выполнение этого условия легко обеспечивается, если внедиагональные элементы матрицы A по модулю много меньше соответствующих диагональных элементов $|a_{ij}| << |a_{ii}|$, например, при всех i

$$\sum_{j=1, j \neq i}^{n} |a_i j| < a_i i$$

3. Листинг программы

Репозиторий на GitHub

Функция, отвечающая за реализацию метода простых итераций

```
def simple_iteration(a, b, x0, tol=0.01, max_iter=1000):
    n = len(b)
    rows = len(a)
    x = x0.copy()
    # If the matrix is not square
    if isSquare(a) is False or [0 for _ in range(rows)] in a:
        exit("The matrix is not square")
    A = np.array(a)
    B = np.array(b)
    # If the dimension of the matrices does not match
    if rows != n:
        exit("The dimensions of the matrices do not match")
    # If the determinant is 0
    if np.linalg.det(A) == 0:
        exit("The determinant of the entered matrix A is zero")
    # Let's bring the matrix to the dominance of the diagonal
    A = np.array(rearrange_matrix(a, b)[0])
    B = np.array(rearrange_matrix(a, b)[1])
    # Let's check a sufficient condition for the convergence of the
    iterative process
    if [0.0 for j in range(rows)] in A:
        print("The condition for the predominance of diagonal elements
        is not fulfilled")
        A = np.array(a)
    print("The transformed matrix:")
    for i in range(rows):
        print(*A[i])
    for k in range(max_iter):
        x_new = np.zeros(n)
        for i in range(n):
            s = np.dot(A[i], x)
            x_new[i] = x[i] + (B[i] - s) / A[i][i]
        print(f"Iteration {k+1} {[abs(x[i]-x_new[i]) for i in
        range(len(A))]}")
        if np.linalg.norm(x_new - x) < tol:
            return x_new, k, max([abs(x[i]-x_new[i]) for i in
            range(len(A))]), A, B
        x = x_new
    exit(f"Couldn't find a solution in {max_iter} iterations")
```

4. Примеры и результаты работы программы

Как задать матрицу? 1-Самостоятельно / 2-Из файла / 3-Задать случайно: 2
Введите путь до файла: matrix.txt
Введите точность: 0.01
Исходная матрица:

5.5	1.6	1.7	1.0
2.4	-2.0	-4.5	-1.5
0.8	3.4	0.9	3.0

Преобразованная матрица:

5.5	1.6	1.7	1.0
0.8	3.4	0.9	3.0
2.4	-2.0	-4.5	-1.5

Итерация 1 [15.818181818181818, 14.117647058823529, 8.777777777777779]

Итерация 2 [6.820083184789067, 6.045454545454546, 2.161853832442067]

Итерация 3 [1.0904683195592286, 1.032470029009822, 0.9505090116854819]

Итерация 4 [0.006561223009162967, 0.004975454450132144, 0.12270753531611206]

Итерация 5 [0.036480378712214385, 0.030937589228579476, 0.001288005849272647]

Итерация 6 [0.00939813685627102, 0.008924561245328477, 0.0057061623227012415]

Итерация 7 [0.0008325130988969323, 0.0007008715866428927, 0.0010458679920874459]

Вектор неизвестных: [-0.05409097 0.92302681 -0.10588183], количество итераций 7, погрешность 0.0010458679920874459

Решение найдено. Проверим наши вычисления, подставив вектор решения в исходную систему уравнений

Невязки:
0.99934341895208 ≈ 1.0
-1.4994037117359331 ≈ -1.5
2.999724729286239 ≈ 3.0

5. Вывод

В результате выполнения лабораторной работы я изучила численные методы решения систем линейных алгебраических уравнений, смогла программно реализовать один из методов на языке Python.