Университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Вычислительная математика»

Отчёт

Лабораторная работа №1

Вариант 9

Выполнил:

Прокофьев Арсений Александрович

P3213

Преподаватель:

Машина Екатерина Алексеевна

Санкт-Петербург, 2024 г

.

Цель работы

Разработать программу для решения СЛАУ методом Гаусса-Зейделя. Для итерационных методов 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: $|x_i^{(k)}-x_i^{(k-1)}|$.

Описание метода

Итерационные методы - это методы последовательных приближений. Задается некоторое начальное приближение. Далее с помощью определенного алгоритма проводится один цикл вычислений - итерация. В результате итерации находят новое приближение. Итерации проводятся до получения решения с требуемой точностью.

В методе Гаусса-Зейделя сначала исходную СЛАУ:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}\,x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}\,x_n = b_2 \\ \dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}\,x_n = b_n \end{cases}$$

представляют в виде:

$$x_1 = \frac{1}{a_{1,1}}(b_1 - a_{1,2}x_2 - a_{1,3}x_3 - \dots - a_{1,n}x_n) \ ,$$

$$x_2 = \frac{1}{a_{2,2}}(b_2 - a_{2,1}x_1 - a_{2,3}x_3 - \dots - a_{2,n}x_n) \ ,$$

....

$$x_n = \frac{1}{a_{n,n}}(b_n - a_{n,1}x_1 - a_{n,2}x_2 - \dots - a_{n,n-1}x_{n-1}) \ ,$$

или вообще для любого i :

$$x_i = \frac{1}{a_{i,i}}(b_i - a_{i,1}x_1 - a_{i,2}x_2 - \dots - a_{i,i-1}x_{i-1} - a_{i,i+1}x_{i+1} - \dots - a_{i,n}x_n), \quad i = \overline{1,n} \ .$$

В результате мы получим следующую рекуррентную формулу, которая и составляет метод Гаусса–Зейделя:

$$x_i^{(k+1)} = \frac{1}{a_{i,i}}(b_i - a_{i,1}x_1^{(k+1)} - \dots - a_{i,i-1}x_{i-1}^{(k+1)} - a_{i,i+1}x_{i+1}^{(k)} - \dots - a_{i,n}x_n^{(k)}), \quad i = \overline{1,n} \ .$$

Расчеты по последней формуле продолжаются при $k = 1,2,3,\dots$ до тех пор, пока не будет выполняться условие

$$\max_i |x_i^{(k+1)} - x_i^{(k)}| < \varepsilon$$

С каждой k-ой итерацией решения будут приближаться к действительному решению СЛАУ. Важным критерием этого является то, что матрица коэффициентов СЛАУ должна иметь диагональное преобладание.

Код программы

```
def exitstr():
    print("Неверный ввод")


def trytoint(x):
    a = 0
    try:
        a = int(x)
    except Exception:
        a = None
    return a


def trytofloat(x):
    a = 0
    try:
        a = float(x)
    except Exception:
        a = None
    return a


def output_k(X, N, delta, k):
    global roundconst
    print(f"После k = {k}:")
    output(X, N)
    print(f"delta =", round(delta, roundconst))
    print()


def output(X, N, rounds = False):
    global roundconst
    if isinstance(X, str):
        print(X)
    else:
        for i in range(N):
            x = X[i]
            if rounds:
                x = round(x, roundconst) #для вывода округляем
            print(f"X[{i+1}] = {x}")


def max_to_diagonal(A, B, N):
    for i in range(N):
        max_val = max(A[i], key=abs)  # Находим максимальное значение в строке по
модулю
        max_index = A[i].index(max_val)  # Находим индекс этого значения
        if max_index != i:  # Если максимум не на диагонали, то меняем строки местами
            A[i], A[max_index] = A[max_index], A[i]
            B[i], B[max_index] = B[max_index], B[i]

    #проверка диагонализации
    strog = False
    for i in range(N):
        Sum = 0
        for j in range(N):
            if (i != j):
                Sum += A[i][j]
        if A[i][i] < Sum:
            return False
```

```
        elif A[i][i] > Sum:
            strog = True # для одной строки хотя бы должно быть строго
    return strog


def gauss_zeidel(N, A, B, eps, M, k = 0, X = []):
    global roundconst
    delta = 0
    if (X == []): # 0 итерация
        X = [0]*N
        diagonalize = max_to_diagonal(A, B, N)
        if not diagonalize:
            return "Не удалось диагонализировать"
        #начальное приближение
        for i in range(N):
            X[i] = B[i]/A[i][i]
        output_k(X, N, delta, k)
        k = 1

    for i in range(N):
        s = 0
        for j in range(i): # до i-1
            s += A[i][j]*X[j]
        for j in range(i+1,N):# с i+1 до конца
            s += A[i][j]*X[j]
        x = (B[i] - s)/ A[i][i]
        d = abs(x - X[i])
        if d > delta:
            delta = d
        X[i] = x

    output_k(X, N, delta, k)

    if delta < eps:
        return X
    else:
        if (k < M):
            return gauss_zeidel(N, A, B, eps, M, k + 1, X)
        else:
            return "Итерации расходятся"


def run():
    global roundconst
    print("""
    N — порядок матрицы
    ε — погрешность вычислений
    Aij ,Bi — коэффициенты и правые части уравнений системы
    Xi — начальные приближения
    M — максимально допустимое число итераций
    k — порядковый номер итерации;
    i — номер уравнения, а также переменного, которое вычисляется в соответствующем
цикле;
    j — номер элемента вида AijXj (k) или AijXj (k-1) в правой части соотношения.

    Итерационный процесс прекращается либо при выполнения условия:
    max |Xi(k) - Xi (k-1)| < ε,
    1≤i≤n

    либо при K = M, т.е. итерации не сходятся

    """)
```

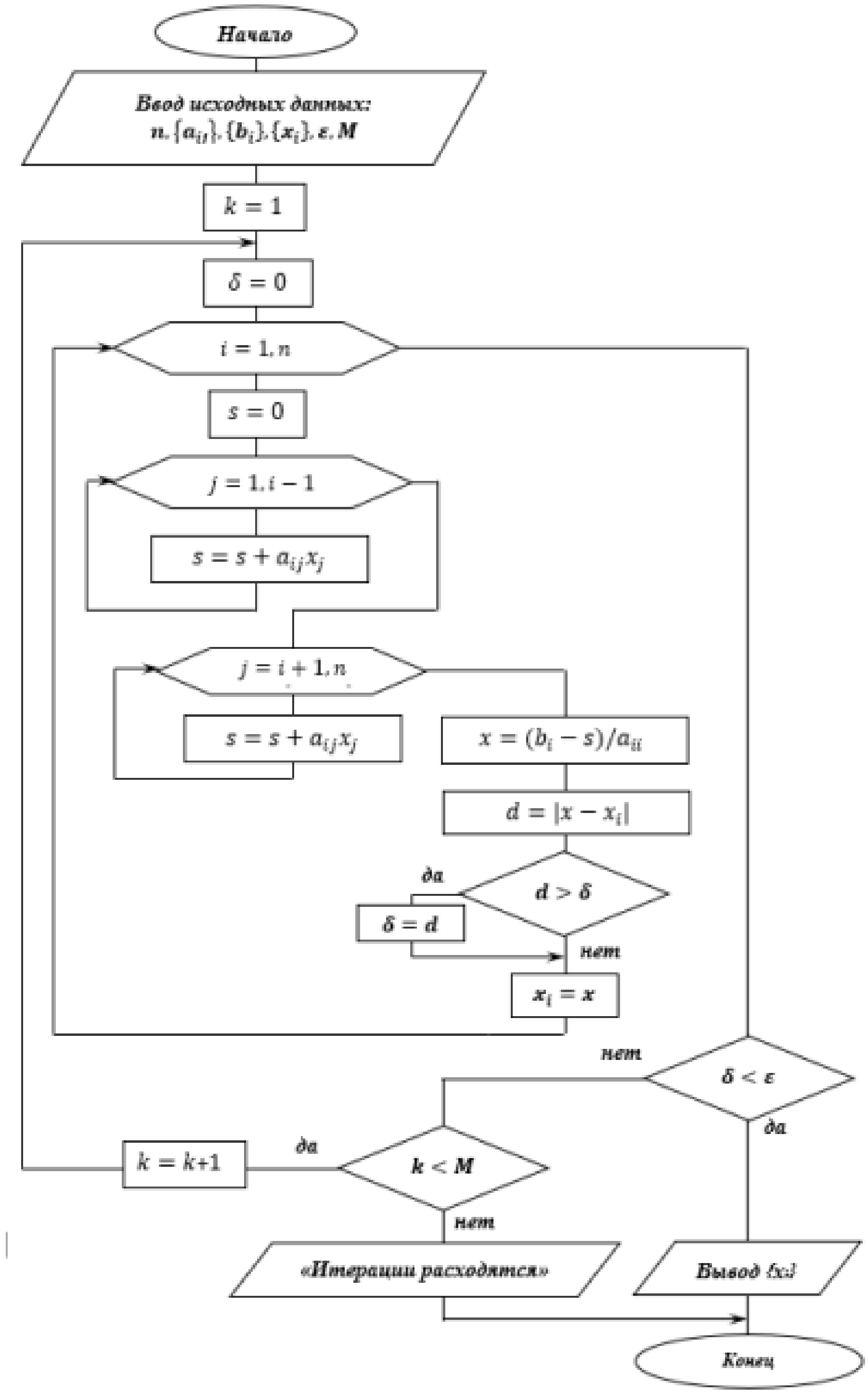
```
    roundconst = 3 #trytoint(input("Точность вывода (количество знаков после запятой) =
"))
    eps = trytofloat(input("Eps="))
    while eps is None:
        exitstr()
        eps = trytofloat(input("Eps="))
    M = trytoint(input("M="))
    while M is None:
        exitstr()
        M = trytoint(input("M="))
    N = trytoint(input("N="))
    while N is None:
        exitstr()
        N = trytoint(input("N="))

    A = []
    for i in range(N):
        A.append([])
        for j in range(N):
            A[i].append(j)
            A[i][j] = trytofloat(input(f"A[{i+1}][{j+1}] = "))
            while A[i][j] is None:
                exitstr()
                A[i][j] = trytofloat(input(f"A[{i+1}][{j+1}] = "))

    B = []
    for i in range(N):
        B.append(trytofloat(input(f"B[{i+1}] = ")))
        while B[i] is None:
            exitstr()
            B[i] = trytofloat(input(f"B[{i+1}] = "))
    print()
    print()
    X = gauss_zeidel(N, A, B, eps, M)
    print()
    print()
    output(X, N, True)


if __name__ == "__main__":
    run()
```

Блок схема алгоритма



Пример работы программы

Входные данные:

Eps=0.01
M=5
N=3
A[1][1] = 2
A[1][2] = 2

A[1][3] = 10
A[2][1] = 10
A[2][2] = 1
A[2][3] = 1
A[3][1] = 2
A[3][2] = 10
A[3][3] = 1
B[1] = 14
B[2] = 12
B[3] = 13

Результат:

После k = 0:
X[1] = 1.2
X[2] = 1.3
X[3] = 1.4
delta = 0

После k = 1:
X[1] = 0.93
X[2] = 0.974
X[3] = 1.0192
delta = 0.381

После k = 2:
X[1] = 1.00068
X[2] = 0.997944
X[3] = 1.0002752000000001
delta = 0.071

После k = 3:
X[1] = 1.00017808
X[2] = 0.999936864
X[3] = 0.9999770112
delta = 0.002

X[1] = 1.0
X[2] = 1.0
X[3] = 1.0

Вывод

В результате выполнения данной лабораторной работы был изучен метод Гаусса-Зейделя для решения СЛАУ.

Основное преимущество данного метода: он не требует много памяти и представляет собой модификацию метода простых итераций. Также данный метод Гаусса-Зейделя обычно сходится быстрее, чем другие итерационные методы.

Однако он имеет и недостатки: в результате получается неточное решение (точность задается пользователем), время сходимости увеличивается с увеличением размера системы, матрица коэффициентов СЛАУ должна преобразовываться в диагональное преобладание.