Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

Лабораторная работа № 1

«Решение СЛАУ с трёхдиагональной матрицей»

по курсу «Численные методы»

Студент *Головкин Д.С. ИУ9-62Б*

Преподаватель *Домрачева А.Б.*

Москва, 2023 г.

СОДЕРЖАНИЕ

Постановка задачи

Цель: проанализировать метод решения систем линейных алгебраических уравнений (СЛАУ) с трёхдиагональной матрицей с помощью метода прогонки.

Дано: $A\overline{x} = \overline{d},\ \overline{d},\ \overline{x} \in R^{n}$, где

$$A = \begin{pmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{pmatrix}, \quad \overline{d} = \begin{pmatrix} 5 \\ 6 \\ 6 \\ 5 \end{pmatrix}$$

Уравнение имеет вид:

$$\begin{pmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \\ 6 \\ 5 \end{pmatrix}$$

Теоретическая часть

1. Выведем формулы, использующиеся в методе прогонки для решения СЛАУ.

Пусть массив a – элементы под диагональю, b – на диагонали c – над диагональю

$$\begin{pmatrix} b_1 & c_1 & 0 & \dots & \dots & 0 \\ a_1 & b_2 & c_2 & \dots & \dots & 0 \\ 0 & a_2 & b_3 & c_3 & \dots & 0 \\ \vdots & \dots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & a_{n-2} & b_{n-1} & c_{n-1} \\ 0 & \dots & \dots & \dots & a_{n-1} & b_n \end{pmatrix}$$

$$\begin{cases} b_1x_1 + c_1x_2 = d_1 \\ a_1x_1 + b_2x_2 + c_2x_3 = d_2 \\ \dots \\ a_{n-1}x_{n-1} + b_nx_n = d_n \end{cases}$$

$$x_1 = \frac{d_1 - c_1x_2}{b_1}$$

$$a_1\frac{d_1 - c_1x_2}{b_1} + b_2x_2 + c_2x_3 = d_2$$

$$x_1 = \frac{d_1}{b_1} - \frac{c_1}{b_1}x_2$$

$$\alpha_1 = \frac{c_1}{b_1}, \beta_1 = \frac{d_1}{b_1}$$

Продолжим рассуждения для x_i и получим решение:

$$x_{i-1} = \alpha_{i-1}x_i + \beta_{i-1}, i = \overline{2, n}$$

$$x_i = \alpha_i x_{i+1} + \beta_i, i = \overline{n-1, 1}$$

$$a_{i-1}x_{i-1} + b_i x_i + c_i x_{i+1} = d_i$$

$$(a_{i-1}\alpha_{i-1} + \beta_i)x_i + c_i x_{i+1} = d_i - a_{i-1}\beta_{i-1}$$

$$x_i = \frac{d_i - a_{i-1}\beta_{i-1}}{a_{i-1}\alpha_{i-1} + b_i} - \frac{c_i}{a_{i-1}\alpha_{i-1} + b_i)x_{i+1}}$$

$$\alpha_i = -\frac{c_i}{a_{i-1}\alpha_{i-1} + b_i}$$

$$\beta_i = d_i - \frac{a_{i-1}\beta_{i-1}}{a_{i-1}\alpha_{i-1} + b_i}, i = \overline{2, n}$$

Если вычислять α_i, β_i, $i = \overline{2, n}$, то алгоритм называется прямым ходом метода прогонки. Заметим, что α_1, β_1 можно вычислить по формулам:

$$\alpha_1 = -\frac{c_1}{b_1}, \quad \beta_1 = \frac{d_1}{b_1}$$

Ниже представлена формула обратного хода методом прогонки:

$$a_{n-1}x_{n-1} + b_n x_n = d_n$$

$$x_n = \frac{d_n - a_{n-1}x_{n-1}}{b_n}$$

β_n выбирается в качестве начального приближения x_n при условии:

$$a_{n-1}\alpha_{n-1} = 0$$

2. Определим необходимые условия метода

$$b_1 \neq 0$$

3. Определим достаточные условия метода

1) $\left|b_i\right| \geq \left|a_{i-1}\right| + \left|c_i\right|, \ i = \overline{2,..n}$

2) $\frac{\left|a_i-1\right|}{\left|c_i\right|} \leq 1$

3) $\frac{\left|c_i\right|}{\left|b_i\right|} \leq 1$

Практическая часть

Решение трехдиагональной СЛАУ методом прогонки

```
import numpy as np
from numpy import linalg as LA

def calculate_x_n(a, b, c, d, n):
    #alpha = [0.0] * n
    #beta = [0.0] * n
    alpha = np.zeros(n, dtype=np.float16)
    beta = np.zeros(n, dtype=np.float16)
    alpha[0] = -c[0] / b[0]
    beta[0] = d[0] / b[0]
    for i in range(1,n-1):
        #print(i)
        alpha[i] = -c[i] / (a[i-1]*alpha[i-1] + b[i])
        beta[i] = (d[i] - a[i-1]*beta[i-1]) / (a[i-1]*alpha[i-1] + b[i])
    x_n = (d[n-1] - a[n-2]*beta[n-2]) / (a[n-2]*alpha[n-2] + b[n-1])
    #print(alpha, beta)
    return x_n, alpha, beta

def calculate_x_vector(x_n, alpha, beta, n):
    x = [0.0] * (n-1)
    x.append(x_n)
    for i in range(n-2, -1, -1):
        x[i] = alpha[i]*x[i+1] + beta[i]
    return x

def main():
    n = 4
    a_coefs = [1.0] * (n-1)
    b_coefs = [4.0] * n
    c_coefs = [1.0] * (n-1)
    d_coefs = np.float16([5.0, 6.0, 6.0, 5.0])
    x_n, alpha, beta = calculate_x_n(a_coefs, b_coefs, c_coefs, d_coefs,
n)
    x = calculate_x_vector(x_n, alpha, beta, n)
    print('Приближенное решение ', x)
    print('e1 ', np.array([1.0,1.0,1.0,1.0]) - np.array(x))
    A_matrix = np.float32([[4, 1, 0, 0],
                [1, 4, 1, 0],
                [0, 1, 4, 1],
```

```
                [0, 0, 1, 4]])
    d_new = np.dot(A_matrix, x)
    epsilon = d_coefs - d_new
    #print(epsilon)
    A_inv = LA.inv(A_matrix).astype(np.float16)
    res = np.dot(A_inv, epsilon).astype(np.float16)
    print('e2 ', res)

if __name__ == '__main__':
    main()
```

Тестирование

Для тестирования программы были выбраны трёхдиагональная матрица А и вектор d:

$$A = \begin{pmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{pmatrix}, \quad \overline{d} = \begin{pmatrix} 5 \\ 6 \\ 6 \\ 5 \end{pmatrix}$$

Метод прогонки показал ошибку в 5 знаке после запятой. Для получения априорной погрешности необходимо воспользоваться вектором невязки.

Вектор невязки:

$$A\overline{x^*} = \overline{d^*}$$

$$A\left(\overline{x} - \overline{x^*}\right) = \overline{d} - \overline{d^*}$$

$$A\overline{\varepsilon 2} = \overline{r}, \text{ где } \overline{r} - \text{ вектор невязки}$$

Результат выполнения:

$$\overline{x^*} = (0.999984,\ 1.00006,\ 0.9997,\ 1.00006)$$

$$\overline{\varepsilon 1} = (1.74397e-05\ \ -6.97590e-05\ \ 2.61660e-04\ \ -6.54150e-05)$$

$$\overline{\varepsilon 2} = (1.746e-05\ \ -6.974e-05\ \ 2.618e-04\ \ -6.545e-05)$$

Вывод

В результате выполнения данной лабораторной работы были реализован метод решения трехдиагональной СЛАУ методом прогонки, а также было найдено значение ошибки.

Метод прогонки не имеет методологической ошибки. Однако при вычислении может возникнуть вычислительная погрешность. На данных, которые были в задании к лабораторной работе, вычислительная погрешность была выявлена с помощью уменьшения разрядной сетки.