v

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

Отчет по лабораторной работе № 1
«Решение СЛАУ с трехдиагональной матрицей методом прогонки»
по курсу «Численные методы»

Выполнил:

студент группы ИУ9-61Б

Баев Д. А.

Проверила:

Домрачева А. Б.

Москва, 2023

1. Цель

Цель данной работы: изучение накопления погрешности в решении системы линейных алгебраических уравнений (СЛАУ) с трехдиагональной матрицей методом прогонки.

2. Постановка задачи

Дано: $A\overline{x} = \overline{d}$, где $A \in R^{n\times n}$, $\overline{x} \in R^{n}$, $\overline{d} \in R^{n}$, A – трехдиагональная матрица.

Найти: Решение СЛАУ с помощью метода прогонки.

3. Основные теоретические сведения

Описание метода прогонки:

Пусть a – элементы под диагональю, b – элементы главной диагонали, c – элементы над диагональю.

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

Получается следующая СЛАУ:

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

Из ее следует:

$$x_1 = \frac{d_1 - c_1x_2}{b_1} = -\frac{c_1}{b_1}x_2 + \frac{d_1}{b_1}, \quad b_1 \neq 0$$

Произведя замену $\alpha_1 = -\frac{c_1}{b_1}$, $\beta_1 = \frac{d_1}{b_1}$, получаем $x_1 = \alpha_1x_2 + \beta_1$.

Подставляем полученные значения во второе уравнение системы:

$$a_1(\alpha_1x_2 + \beta_1) + b_2x_2 + c_2x_3 = d_2$$

Получаем:

$$x_2 = -\frac{c_2}{a_1\alpha_1 + b_2}x_3 + \frac{d_2 - a_1\beta_1}{a_1\alpha_1 + b_2}$$

Произведя замену $\alpha_2 = -\frac{c_2}{a_1\alpha_1 + b_2}$, $\beta_2 = \frac{d_2 - a_1\beta_1}{a_1\alpha_1 + b_2}$, получаем $x_2 = \alpha_2 x_3 + \beta_2$.

Продолжая аналогичные рассуждения, получаем формулы для нахождения α_i и β_i:

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

$$\alpha_i = -\frac{c_i}{a_{i-1}\alpha_{i-1} + b_i}\ ,\ \beta_2 = \frac{d_i - a_{i-1}\beta_{i-1}}{a_{i-1}\alpha_{i-1} + b_i}$$

где $i = \overline{2,\ n-1}$

x_i вычисляется следующим образом:

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

$$x_n = \frac{d_n - a_{n-1}\beta_{n-1}}{a_{n-1}\alpha_{n-1} + b_n} = \beta_n.$$

Вычисление α_i и β_i называется прямым ходом метода прогонки, а вычисление x_i – обратным ходом метода прогонки.

Условия диагонального приближения (достаточные условия):

$$\left|\frac{a_{i-1}}{b_i}\right| <= 1,\ \left|\frac{c_i}{b_i}\right| <= 1,\ b_i >= \left|a_{i-1}\right| + \left|c_i\right|,\ i = \overline{2,\ n}$$

Достаточное и необходимое условие:

$b_1 \neq 0$

Оценка погрешности:

В результате работы программы нашли приближенное решение – вектор $\overline{x^*}$. Для оценки погрешности вычисления вычисляем:

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

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

$$\overline{r} = (\overline{d} - \overline{d^*})$$

$$\overline{e} = (\overline{x} - \overline{x^*})$$

$$A\overline{e} = \overline{r}$$

где $\overline{e}$ – искомый вектор ошибок.

Тогда $\overline{e} = A^{-1}\overline{r}$

Точное решение $\overline{x} = \overline{x^*} - \overline{e}$.

4. Реализация

Листинг 1. Метод прогонки для решения трехдиагональной СЛАУ.

```python
import numpy as np

type = np.longdouble

def make_matrix(mid, top, bot):
    n = len(mid)
    a = np.zeros((n, n), dtype=type)
    a[0][0] = mid[0]
    a[0][1] = top[0]
    a[n - 1][n - 1] = mid[n - 1]
    a[n - 1][n - 2] = bot[n - 2]
    for i in range(1, n - 1):
        a[i][i] = mid[i]
        a[i][i - 1] = bot[i - 1]
        a[i][i + 1] = top[i]
    return a


def run(mid, top, bot, b):
    n = len(b)
    x = np.zeros(n, dtype=type)
    v = np.zeros(n, dtype=type)
```

```python
    u = np.zeros(n, dtype=type)

    v[0] = -top[0] / mid[0]
    u[0] = b[0] / mid[0]
    for i in range(1, n):
        if i == n - 1:
            v[i] = 0
        else:
            v[i] = -top[i] / (bot[i - 1] * v[i - 1] + mid[i])
        u[i] = (b[i] - bot[i - 1] * u[i - 1]) / (bot[i - 1] * v[i - 1] +
mid[i])

    x[n - 1] = u[n - 1]
    for i in range(n - 1, 0, -1):
        x[i - 1] = v[i - 1] * x[i] + u[i - 1]
    return x


if __name__ == "__main__":
    n = 4
    mid = np.array([4, 4, 4, 4], dtype=type)
    top = np.array([1, 1, 1], dtype=type)
    bot = np.array([1, 1, 1], dtype=type)
    b = np.array([5, 6, 6, 5], dtype=type)

    x = run(mid, top, bot, b)
    a = make_matrix(mid, top, bot)
    res_err = a @ x
    r = b - res_err

    a_inv = np.linalg.inv(a.astype(np.float32))
    e = a_inv @ r
    print(x)
    print(e)
```

5. Тестирование

Для тестирования полученной программы в качестве входных данных были выбраны матрица А:

$$\begin{pmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{pmatrix}$$

И вектор $\overline{d}$:

$$\overline{d} = (5\ 6\ 6\ 5\ )$$

В качестве типа данных используется long double (128 бит).

В результате работы программы было получено значение вектора ошибок:

$$\overline{e} = (-\ 8.30011202e - 21,\ 3.32004481e - 20,\ -\ 1.24501681e - 19,\ 3.11254203e - 20)$$

Видно, что этот вектор не является нулевым. Это связано с использованием слишком точного типа данных long double.

6. Вывод

В ходе выполнения лабораторной работы был изучен и программно реализован метод решения трехдиагональных СЛАУ методом прогонки.

Хотя метод прогонки не имеет методологической погрешности, и априорное решение тестовой задачи соответствует единичному вектору, в практической реализации накапливается ошибка округления из-за особенностей представления чисел с плавающей точкой в памяти компьютера, что приводит к ненулевому вектору ошибок.