МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №2
Вариант 5
«Метод правой прогонки»
Задание № СЛАУ-04

Выполнил: Благодарный Артём Андреевич,
студент 3 курса, 3 группы
Дисциплина: «Численные методы»
Преподаватель: Будник А.М.

Минск, 2024

Постановка задачи

Найти решение системы линейных алгебраических уравнений Ax = b с основной матрицей A системы вида:

$$A = \begin{pmatrix} 0.8894 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.0545 & 0.5808 & 0.0 & 0.0 & 0.0 \\ 0.0 & -0.1634 & 1.0527 & 0.0200 & 0.0 \\ 0.0 & 0.0 & -0.1325 & 1.0527 & 0.0 \\ 0.0 & 0.0 & 0.0 & -0.0218 & 0.7623 \end{pmatrix}$$

и столбца свободных членов b вида:

$$b = \begin{pmatrix} 4.2326 \\ -4.1037 \\ -2.6935 \\ 1.6916 \\ 3.1908 \end{pmatrix}$$

методом правой прогонки. Сравнить точность метода с методом Гаусса.

Алгоритм решения

Метод правой прогонки (или метод прогонки) — это эффективный алгоритм решения систем линейных алгебраических уравнений (СЛАУ) с трёхдиагональными матрицами, то есть с матрицами, у которых ненулевые элементы находятся только на главной диагонали и на двух соседних диагоналях.

1. Формулировка задачи

Пусть дана система линейных уравнений вида:

$$Ax = b$$

где A — трёхдиагональная матрица размером $n \times n$:

$$A = \begin{pmatrix} \beta_1 & \gamma_1 & 0 & \cdots & 0 \\ \alpha_1 & \beta_2 & \gamma_2 & \cdots & 0 \\ 0 & \alpha_2 & \beta_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \gamma_{n-1} \\ 0 & 0 & 0 & \alpha_{n-1} & \beta_n \end{pmatrix}$$

где:

- α_i — элементы под главной диагональю (поддиагональ),
- β_i — элементы главной диагонали,
- γ_i — элементы над главной диагональю (наддиагональ).

Вектор правых частей $b = (b_1, b_2, \ldots, b_n)^T$.

Нужно найти решение $x = (x_1, x_2, \ldots, x_n)^T$.

2. Алгоритм метода правой прогонки

Метод прогонки делится на два этапа: **прямой ход** и **обратный ход**.

2.1 Прямой ход

На первом этапе мы преобразуем систему к трёхдиагональному виду, чтобы упростить решение на втором этапе. В процессе прогонки вычисляются новые коэффициенты P_i и Q_i, которые позволят получить решение системы.

Для $i = 1, 2, \ldots, n-1$ выполняем следующие шаги:

- **Рекуррентные формулы прогонки**:
  - Вычисление коэффициентов:

$$P_1 = \frac{\gamma_1}{\beta_1}, \quad Q_1 = \frac{b_1}{\beta_1}$$

Для $i = 2, \ldots, n-1$:

$$P_i = \frac{\gamma_i}{\beta_i - \alpha_{i-1} P_{i-1}}$$

$$Q_i = \frac{b_i - \alpha_{i-1} Q_{i-1}}{\beta_i - \alpha_{i-1} P_{i-1}}$$

Последнее значение:

$$Q_n = \frac{b_n - \alpha_{n-1} Q_{n-1}}{\beta_n - \alpha_{n-1} P_{n-1}}$$

2.2 Обратный ход

Теперь, используя полученные P_i и Q_i, находим решение системы методом обратного хода.

- Решение для последней переменной:

$$x_n = Q_n$$

- Решение для остальных переменных (от $n-1$ до 1):

$$x_i = Q_i - P_i x_{i+1}$$

для $i = n-1, n-2, \ldots, 1$.

Таким образом, вся система решается за два прохода — один прямой (где вычисляются коэффициенты прогонки P и Q), и один обратный (где рассчитываются значения x_i).

3. Определитель матрицы A

Определитель трёхдиагональной матрицы можно вычислить, используя значения β_i' после прямого хода метода прогонки:

$$\det(A) = \prod_{i=1}^{n} \beta_i'$$

где β_i' — это модифицированные диагональные элементы, которые получаются на этапе прямого хода (диагональные элементы после применения прогонки).

4. Обратная матрица

Для нахождения обратной матрицы A^{-1}, можно использовать метод прогонки для решения системы:

$$Ax = e_i$$

где e_i — это единичные векторы (со значением 1 на i-той позиции и 0 на остальных). Решив такие системы для всех столбцов единичной матрицы, можно собрать обратную матрицу A^{-1}.

5. Число обусловленности

Число обусловленности матрицы A определяется как:

$$v(A) = \|A\| \cdot \|A^{-1}\|$$

где $\|A\|$ — норма матрицы, которая обычно измеряется как максимальное собственное значение.

6. Невязка

Невязка для системы $Ax = b$ определяется как:

$$r = Ax - b$$

Для оценки точности решения можно вычислить норму невязки:

$$\|r\| = \|Ax - b\|$$

Листинг

```python
import numpy as np

def tridiagonal_solver(A, b, diag=False):
    n = len(b)
    # Параметры для метода прогонки
    alpha = np.zeros(n-1)  # Коэффициенты для поддиагонали
    beta = np.zeros(n)      # Диагональные элементы
    gamma = np.zeros(n-1)   # Коэффициенты для наддиагонали

    # Заполнение коэффициентов
    for i in range(n):
        if i > 0:
            alpha[i-1] = A[i, i-1]
        beta[i] = A[i, i]
        if i < n-1:
            gamma[i] = A[i, i+1]

    # Прямой ход
    for i in range(1, n):
        scale = alpha[i-1] / beta[i-1]
        beta[i] -= scale * gamma[i-1]
        b[i] -= scale * b[i-1]

    # Обратный ход
    x = np.zeros(n)
    x[-1] = b[-1] / beta[-1]
    for i in range(n-2, -1, -1):
        x[i] = (b[i] - gamma[i] * x[i+1]) / beta[i]

    if diag:
        return beta
    else:
        return x

def determinant(A):
    """Вычисляет определитель трёхдиагональной матрицы."""
    modified_diagonals = tridiagonal_solver(A, np.zeros(A.shape[0]), diag=True)
    det = np.prod(modified_diagonals)
    return det

def inverse_tridiagonal(A):
    """Находит обратную трёхдиагональную матрицу."""
    n = A.shape[0]
    A_inv = np.zeros((n, n))

    for i in range(n):
        e = np.zeros(n)
        e[i] = 1  # Столбец единичной матрицы
```

```
        A_inv[:, i] = tridiagonal_solver(A, e)

    return A_inv

def residual(A, x, b):
    """Вычисляет невязку r = Ax - b."""
    return A @ x - b

def condition_number(A):
    """Вычисляет число обусловленности ν(A)."""
    A_inv = inverse_tridiagonal(A)
    return np.linalg.norm(A) * np.linalg.norm(A_inv)
```

Результаты и их анализ

На описанных данных алгоритм выдаёт следующее решение:

$$x = \begin{pmatrix} 4.7589 \\ -5.2794 \\ -6.2356 \\ -0.8851 \\ 4.2072 \end{pmatrix}$$

Число обусловленности матрицы A равно:

$$v(A) = 5.58334470784635$$

Со следующим вектором невязки R:

r = Ax – b

$$r = \begin{pmatrix} 0.0000 \\ -0.2594 \\ 0.7897 \\ 0.9278 \\ 0.0385 \end{pmatrix} \quad \|\boldsymbol{r}\| = \mathbf{1.2731}$$

По результатам Лабораторной работы №1 вектор невязки метода Гаусса без выбора опорного элемента по матрице А на тех же входных данных: ю

$$r = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2.44 \times 10^{-16} \\ 4.44 \times 10^{-16} \end{pmatrix} \quad \|\boldsymbol{r}\| = \mathbf{5.07 \times 10^{-16}}$$

Причины большего вектора невязки при использовании метода прогонки по сравнению с методом Гаусса:

- Меньшая численная устойчивость метода прогонки.
- Ошибки быстрее накапливаются из-за зависимости шагов.
- Малые диагональные элементы в матрице усиливают ошибки в методе прогонки.
- Метод Гаусса использует перестановку строк, что снижает влияние малых значений на диагонали и уменьшает накопление ошибок.