Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
Московский государственный технический университет имени Н.Э. Баумана

Лабораторная работа №1
по курсу «Численные методы»
«Решение СЛАУ. Метод прогонки»

Выполнил:
студент группы ИУ9-62
Иванов Георгий

Проверила:
Домрачева А.Б.

Москва, 2017

Цель:

Анализ метода прогонки и решение СЛАУ с трёхдиагональной матрицей с помощью данного метода.

Постановка задачи:

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

Найти: Решение СЛАУ с помощью метода прогонки, т.е $\overline{x}$ - ? при известных $A, \overline{d}$

Тестовый пример:

В качестве трехдиагональной матрицы A возьмем:

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

А в качестве вектора $\overline{d}$:

$$\overline{d} = \begin{pmatrix} 5 \\ 6 \\ 6 \\ 5 \end{pmatrix}$$

При этом уравнение примет вид:

$$A\overline{x} = \overline{d} \quad \Leftrightarrow \quad \begin{pmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \\ 6 \\ 5 \end{pmatrix}$$

Теоретические сведения:

Метод прогонки используется для решения систем линейных уравнений вида $A\overline{x} = \overline{d}$, где A - трёхдиагональная матрица. Представляет собой вариант метода последовательного исключения неизвестных. Метод прогонки был предложен И. М. Гельфандом и О. В. Локуциевским (в 1952 году; опубликовано в 1960 и 1962 годах), а также независимо другими авторами.

Описание алгоритма:

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

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

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

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

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

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

Вводим замену переменных:

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

Проводим аналогичные действия для остальных x_i, т. о. можно предположить, что решение

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

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

$$a_{i-1}x_{i-1} + b_ix_i + c_ix_{i+1} = d_i$$

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

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

Вводим замену:

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

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

Получили α_i, β_i, необходимые для решения.

Вычисление $\alpha_i, \beta_i, i = \overline{2,n}$ называется прямым ходом метода прогонки, при этом начальные значения α_1, β_1 вычисляются по формулам:

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

Соответственно формула:

$$a_{n-1}x_{n-1} + b_nx_n = d_n$$

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

называется обратным ходом метода прогонки.

В качестве начального приближения x_n выбирается значение β_n в предположении, что

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

Необходимое условие метода:

$$b_1 \neq 0$$

Достаточные условия метода:

1. $|b_i| \geq |a_{i-1}| + |c_i|$, $i = \overline{2, n-1}$ - без него возможно решение, но может и не быть.
2. $|d_i| > |c_i|$, $i = \overline{2, n-1}$

Оценка погрешности для решения СЛАУ при отсутствии точного решения:
Найти приближенный вектор $\overline{x}^*$.
При вычислении $\widetilde{a_{ij}}$ и $\widetilde{d_i}$ с плавающей точкой возникает погрешность:

$$\begin{array}{rc} & A\overline{x}^* = \overline{d}^* \\ - & A\overline{x} = \overline{d} \\ \hline & A(\overline{x}^* - \overline{x}) = (\overline{d}^* - \overline{d}) \end{array}$$

или:

$$A\overline{\varepsilon} = \overline{r} \quad \Leftrightarrow \quad \boxed{\overline{\varepsilon} = A^{-1}\overline{r}}$$

где $\overline{\varepsilon}$ - вектор ошибки

Тогда исходный вектор $\overline{x} = \overline{x}^* - \overline{\varepsilon}$.

Практическая реализация:

Листинг 1. Метод прогонки для решения СЛАУ с трёхдиагональной матрицей

```
#!python -v
# -*- coding: utf-8 -*-

import numpy

def tridiagonalMatrixAlgorithm(d,c,a,b):
    alpha = [0]
    beta = [0]
    n = len(b)

    for i in range(1,n-1):
        if i == 1:
```

```
            alpha.append(-c[i]/d[i])
            beta.append(b[i]/d[i])
        else:
            alpha.append(-c[i]/(d[i]+alpha[i-1]*a[i-1]))
            beta.append((b[i]-a[i-1]*beta[i-1])/(d[i]+alpha[i-1]*a[i-1]))

    x = [0 for _ in range(n)]
    n -= 1
    x[n] = (b[n]-a[n-1]*beta[n-1])/(d[n]+a[n-1]*alpha[n-1])
    for i in range(n-1,0,-1):
        x[i]=x[i+1]*alpha[i]+beta[i]
    return x[1:]

def init_array(matrix,bb):
    a, b, d, c = [0], [0], [0], [0]
    d.extend([matrix[i][i] for i in range(len(matrix[0]))])
    c.extend([matrix[i][i + 1] for i in range(len(matrix[0]) - 1)])
    a.extend([matrix[i + 1][i] for i in range(len(matrix[0]) - 1)])
    b.extend(bb)
    return a,b,c,d

def valueB(matrix,x):
    b = [0 for _ in range(len(x))]
    for j in range(len(matrix)):
        b[j] = sum(matrix[j][i] * x[i] for i in range(len(matrix[j])))
    return b

def find_error(matrix,d1,d2,x2):
    r = [d2[i] - d1[i] for i in range(len(matrix))]
    matrix_rev = numpy.matrix(matrix).I
    matrix_rev = matrix_rev.tolist()

    e = [0 for _ in range(0, len(d1))]
    for j in range(0, len(matrix_rev)):
        e[j] = sum(matrix_rev[j][i] * r[i] for i in range(len(matrix_rev[j])))

    x = [x2[i] - e[i] for i in range(0, len(matrix_rev))]
    return x,e[0:4]
```

```
if __name__=="__main__":
    matrix = [[1,2,0,0],
              [2,−1,−1,0],
              [0,1,−1,1],
              [0,0,1,1]]
    D = [5,−3,3,7]
    a,b,c,d = init_array(matrix,D)
    x = tridiagonalMatrixAlgorithm(d,c,a,b)
    print(x)
    D_ = valueB(matrix,x)
    print(D_)
    x_real,error = find_error(matrix,D,D_,x)
    print(error)
    print(x_real)
```

Результаты:

Для тестирования полученной программы было выбрано в качестве трехдиагональной матрицы A:

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

В качестве вектора $\overline{d}$:

$$\overline{d} = \begin{pmatrix} 5 \\ 6 \\ 6 \\ 5 \end{pmatrix}$$

В результате работы программы (Листинг 1) получаем значения:

$$\overline{\varepsilon} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \overline{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Как видно выше, не всегда вектор $\overline{\varepsilon}$ может содержать погрешность (нулевой вектор ошибок). Поэтому чтобы вектор ошибок не был ненулевым, протестируем нашу программу на других

входных данных, где:

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 2 & -1 & -1 & 0 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad \overline{d} = \begin{pmatrix} 5 \\ -3 \\ 3 \\ 7 \end{pmatrix}$$

В результате работы программы (Листинг 1) получаем значения:

$$\overline{\varepsilon} = \begin{pmatrix} 8.074349270001139e^{-17} \\ -4.0371746350005693e^{-17} \\ 2.0185873175002846e^{-16} \\ -2.0185873175002846e^{-16} \end{pmatrix} \quad \overline{x} = \begin{pmatrix} 0.9999999999999999 \\ 2.0 \\ 3.0 \\ 3.9999999999999996 \end{pmatrix}$$

Выводы:

В ходе выполнения лабораторной работы был рассмотрен метод решения СЛАУ с трёхдиагональной матрицей: метод прогонки, так же для метода была написана реализация на языке программирования Python.

Для метода прогонки можно отметить то, что отсутствует методологическая (логическая) погрешность, но присутствует вычислительная погрешность в связи с использованием чисел с плавающей запятой (нахождение обратной матрицы A^{-1}) , ведущая к высокому накоплению вычислительной ошибки.