Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1
Задание №2 по курсу «Численные методы»

Студент:	С. М. Власова
Преподаватель:	И. Э. Иванов
Группа:	М8О-306Б
Дата:	
Оценка:	
Подпись:	

Москва, 2020

Задание №1.2

Реализовать метод прогонки в виде программы, задавая в качестве входных данных ненулевые элементы матрицы системы и вектор правых частей. Используя разработанное программное обеспечение, решить СЛАУ с трехдиагональной матрицей.
Вариант: 12

$$\begin{cases} -11 \cdot x_1 + 9 \cdot x_2 = -114 \\ x_1 - 8 \cdot x_2 + x_3 = 81 \\ -2 \cdot x_2 - 11 \cdot x_3 + 5 \cdot x_4 = -8 \\ 3 \cdot x_3 - 14 \cdot x_4 + 7 \cdot x_5 = -38 \\ 8 \cdot x_4 + 10 \cdot x_5 = 144 \end{cases}$$

1 Описание метода решения

Метод прогонки является частным случаем метода Гаусса и используется для решения систем линейных уравнений вида $Ax = b$, где A — трёхдиагональная матрица.
Трёхдиагональной матрицей называется матрица такого вида, где во всех остальных местах, кроме главной диагонали и двух соседних с ней, стоят нули.
Системы такого вида часто возникают при решении различных задач математической физики, а также при решении других вычислительных задач (например, приближения функций сплайнами).

Метод прогонки состоит из двух этапов: *прямой прогонки* и *обратной прогонки*. На первом этапе определяются *прогоночные коэффициенты*, а на втором – находят неизвестные x.

СЛАУ имеет вид:

$$\begin{cases} b_1 \cdot x_1 + c_1 \cdot x_2 = d_1, \qquad a_1 = 0 \\ a_2 \cdot x_1 + b_2 \cdot x_2 + c_2 \cdot x_3 = d_2 \\ \quad a_3 \cdot x_2 + b_3 \cdot x_3 + c_3 \cdot x_4 = d_3 \\ \qquad \cdots\cdots\cdots \\ \qquad\quad a_{n-1} \cdot x_{n-2} + b_{n-1} \cdot x_{n-1} + c_{n-1} \cdot x_n = d_{n-1} \\ \qquad\qquad a_n \cdot x_{n-1} + b_n \cdot x_n = d_n \qquad c_n = 0 \end{cases}$$

Решение СЛАУ будем искать в виде

$$x_i = P_i \cdot x_{i+1} + Q_i, \qquad \overline{1, n},$$

где P_i и Q_i — прогоночные коэффициенты.
Прямая прогонка состоит в вычислении этих коэффициентов P_i и Q_i , где i – номер строки матрицы. Этот этап выполняется при $i = \overline{1, n}$ строго по возрастанию значения i.

1) $i = 1$:
Для определения коэффициентов P_1 и Q_1 выразим из первого уравнения СЛАУ x_1 через x_2

$$x_1 = \frac{-c_1}{b_1} \cdot x_2 + \frac{d_1}{b_1} = P_1 \cdot x_2 + Q_1,$$

откуда

$$P_1 = \frac{-c_1}{b_1}, \quad Q_1 = \frac{d_1}{b_1}$$

i) $i = \overline{2, n-1}$:
Для определения коэффициентов P_i и Q_i аналогично выразим из i-го уравнения СЛАУ x_i через x_{i+1}

$$x_i = \frac{-c_i}{b_i + a_i \cdot P_{i-1}} \cdot x_{i+1} + \frac{d_i - a_i \cdot Q_{i-1}}{b_i + a_i \cdot P_{i-1}} = P_i \cdot x_{i+1} + Q_i,$$

откуда

$$P_i = \frac{-c_i}{b_i + a_i \cdot P_{i-1}}, \quad Q_i = \frac{d_i - a_i \cdot Q_{i-1}}{b_i + a_i \cdot P_{i-1}}$$

Отсюда видно, почему так важен порядок поиска коэффициентов — строго по возрастанию значения i — значения коэффициентов, которые мы ищем на текущей итерации, зависят от значений коэффициентов, найденных на предыдущей итерации.

n) $i = n$:
Из последнего уравнения СЛАУ имеем

$$x_n = \frac{-c_n}{b_n + a_n \cdot P_{n-1}} \cdot x_{n+1} + \frac{d_n - a_n \cdot Q_{n-1}}{b_n + a_n \cdot P_{n-1}} = P_n \cdot x_{n+1} + Q_n = 0 \cdot x_{n+1} + Q_n,$$

т.е.

$$P_n = 0, \quad Q_n = \frac{d_n - a_n \cdot Q_{n-1}}{b_n + a_n \cdot P_{n-1}} = x_n$$

Таким образом, прямой ход метода прогонки по определению прогоночных коэффициентов P_i и Q_i, $i = \overline{1, n}$ завершен. Зафиксируем еще раз найденные формулы:

$i = 1$:

$$P_1 = \frac{-c_1}{b_1}, \quad Q_1 = \frac{d_1}{b_1}$$

$i = \overline{2, n-1}$:

$$P_i = \frac{-c_i}{b_i + a_i \cdot P_{i-1}}, \quad Q_i = \frac{d_i - a_i \cdot Q_{i-1}}{b_i + a_i \cdot P_{i-1}}$$

$i = n$:

$$P_n = 0, \quad Q_n = \frac{d_n - a_n \cdot Q_{n-1}}{b_n + a_n \cdot P_{n-1}}$$

Обратный ход метода прогонки заключается в поиске самих значений x_i, согласно уже полученной формуле $x_i = P_i \cdot x_{i+1} + Q_i, \quad \overline{1, n}$. Значения x_i будем искать в обратном порядке.

$$\begin{cases} x_n = P_n \cdot x_{n+1} + Q_n = 0 \cdot x_{n+1} + Q_n = Q_n \\ x_{n-1} = P_{n-1} \cdot x_n + Q_{n-1} \\ x_{n-2} = P_{n-2} \cdot x_{n-1} + Q_{n-2} \\ \dots\dots\dots\dots \\ x_1 = P_1 \cdot x_2 + Q_1 \end{cases}$$

Общее число операций в методе прогонки равно $8 \cdot n + 1$, т.е. пропорционально числу уравнений. Такие методы решения СЛАУ называют *экономичными*. В то же время, число операций в методе Гаусса пропорционально n^3.
Условия устойчивости метода прогонки:

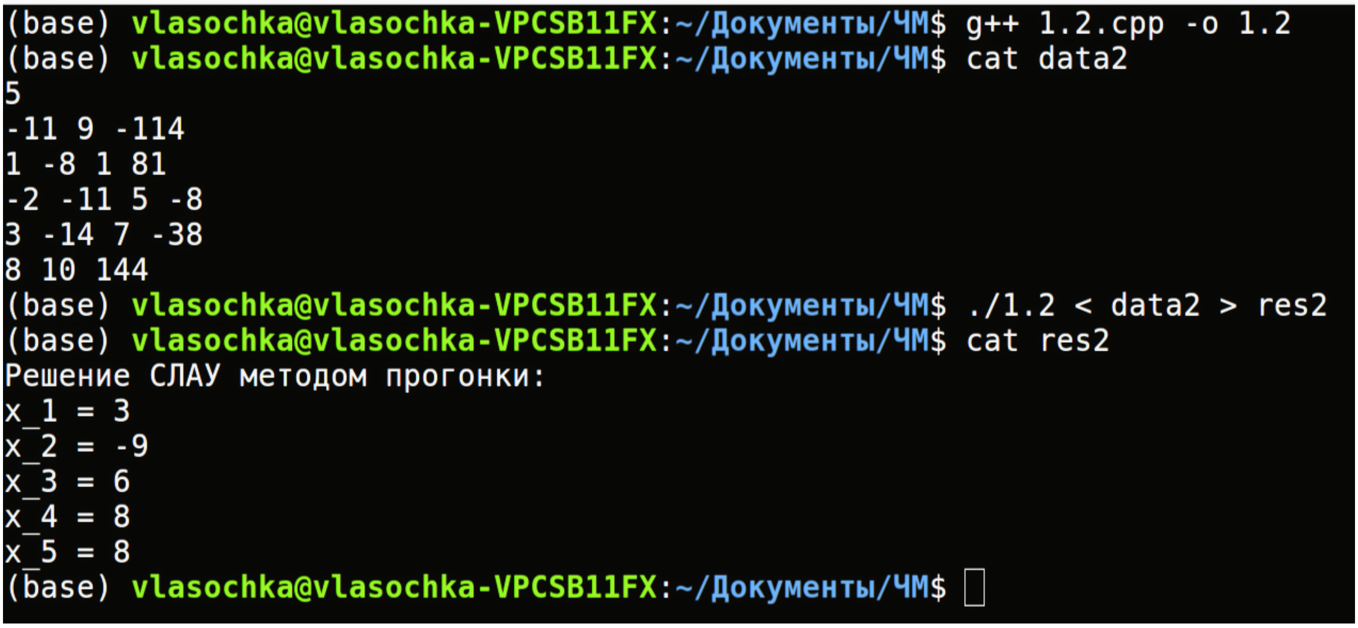
$$a_i \neq 0, \quad c_i \neq 0, \quad i = \overline{2, n-1}$$

$$|b_i| \geq |a_i| + |c_i|, \quad i = \overline{1, n},$$

причем строгое неравенство имеет место хотя бы при одном i.
Здесь устойчивость понимается в смысле не накопления погрешности решения в ходе вычислительного процесса при малых погрешностях входных данных (правых частей и элементов матрицы СЛАУ).

2 Протокол

Входные данные я храню в файле *data2*: первая строка — размерность матрицы, на следующих строках — ненулевые коэффициенты матрицы и коэффициенты вектора правых частей по порядку.
Выходные данные я записываю в файл *res2*.
Скриншот консоли:



```
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ$ g++ 1.2.cpp -o 1.2
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ$ cat data2
5
-11 9 -114
1 -8 1 81
-2 -11 5 -8
3 -14 7 -38
8 10 144
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ$ ./1.2 < data2 > res2
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ$ cat res2
Решение СЛАУ методом прогонки:
x_1 = 3
x_2 = -9
x_3 = 6
x_4 = 8
x_5 = 8
(base) vlasochka@vlasochka-VPCSB11FX:~/Документы/ЧМ$
```

Можно видеть, что программа корректно вычислила решение системы.

$$X = \begin{pmatrix} 3 \\ -9 \\ 6 \\ 8 \\ 8 \end{pmatrix}$$

3 Исходный код

Листинг 1: Метод прогонки

```cpp
#include <iostream>
#include <vector>

void PrintX(std::vector<double>& X, int n)
{
    std::cout << "Solution of a SLAE using the run method:: " << std::endl;
    for(int i = 0; i < n; i++)
        std::cout << "x_" << i + 1 << " = " << X[i] << std::endl;
}

int main()
{
    int n;
    double a, b, c, d = 0;
    std::cin >> n;
    std::vector<double> P(n);
    std::vector<double> Q(n);
    std::vector<double> X(n, 0);

    for( int i = 0; i < n; i++) //straight way
    {
        if( i == 0)
        {
            std::cin >> b >> c >> d;
            P[i] = -c/b;
            Q[i] = d/b;
        }
        else if( i != n-1)
        {
            std::cin >> a >> b >> c >> d;
            P[i] = -c/(b + a * P[i-1]);
            Q[i] = (d - a*Q[i-1])/(b + a*P[i-1]);
        }
        else
        {
            std::cin >> a >> b >> d;
            P[i] = 0;
            Q[i] = (d - a*Q[i-1])/(b + a*P[i-1]);
        }
```

```
    }

    for( int i = n-1; i >= 0; i--) //back way
    {
        if( i == n-1)
            X[i] = Q[i];
        else
            X[i] = P[i]*X[i + 1] + Q[i];
    }
    PrintX(X, n);
    return 0;
}
```

4 Выводы

Выполнив второе задание первой лабораторной работы, я познакомилась с еще одним методом поиска решений СЛАУ. Метод прогонки является эффективным методом решения СЛАУ с трех-диагональными матрицами. Этот метод, как и метод Гаусса, имеет прямой и обратный ход. Во время прямого хода находятся коэффициенты, необходимые для поиска решений СЛАУ. Этот этап реализуется за линейной время. Во время обратного хода ищутся сами корни — в обратном порядке, соответственно. Метод экономичный, т.к. мы храним только ненулевые элементы такой матрицы, и достаточно быстрый, т.к. число операций пропорционально количеству строк матрицы.