Белорусский государственный университет

Факультет прикладной математики и информатики

Лабораторная работа №2

Решение СЛАУ методом левой прогонки

Вариант №8

Выполнил:

Студент 2 курса 7 группы ПМ ФПМИ

Шевцов Евгений

Преподаватель:

Будник Анатолий Михайлович

Минск – 2021

Описание метода нахождения решения СЛАУ методом левой прогонки

Будем рассматривать СЛАУ, матрица системы которой трёхдиагональная, т.е. все элементы равны нулю, за исключением элементов, стоящих на главной диагонали, а так же над и под элементами главной диагонали. Тогда система линейных алгебраических уравнений выглядит следующим образом:

$$\begin{cases} c_0x_0 - b_0x_1 + 0 \ldots + 0 = f_0 \\ -a_1x_0 + c_1x_1 - b_1x_2 + 0 + \cdots + 0 = f_1 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0 + \cdots + 0 - a_{n-1}x_{n-2} + c_{n-1}x_{n-1} - b_{n-1}x_n = f_{n-1} \\ 0 + \cdots + 0 - a_nx_{n-1} + c_nx_n = f_n \end{cases}$$, обозначим как систему (1).

Метод левой прогонки основан на идее Гаусса (в вариации «схема единственного деления») и на утверждении, что неизвестные связаны рекуррентным соотношением: $x_{i+1} = \xi_{i+1}x_i + \mu_{i+1}$, i = [0; n - 1].

Введём обозначение прогоночных коэффициентов для первого шага исключения Гаусса:

$\xi_n = \frac{a_n}{c_n}$, $\mu_n = \frac{f_n}{c_n}$.

Тогда последнее уравнение системы (1) примет вид: $\xi_n x_{n-1} + x_n = \mu_n$.

Рассмотрим два последних уравнения. Домножим последнее на b_{n-1} и сложим его с предпоследним, дабы исключить из него x_n, получив следующее равенство:

$$-a_{n-1}x_{n-2} + (c_{n-1} - b_{n-1}\mu_n)x_{n-1} = f_{n-1} + b_{n-1}\mu_n$$

Откуда выразим $x_{n-1} = \xi_{n-1}x_{n-2} + \mu_{n-1}$, где $\xi_{n-1} = \frac{a_{n-1}}{(c_{n-1} - b_{n-1}\xi_n)}$, $\mu_{n-1} = \frac{f_{n-1} + b_{n-1}\mu_n}{(c_{n-1} - b_{n-1}\xi_n)}$.

На этом первый шаг исключения заканчивается, т.к. по условию все остальные уравнения не будут содержать x_n.

Аналогичные действия проводим для остальных переменных, получая формулы для нахождения прогоночных коэффициентов и неизвестных:

$\xi_i = \frac{a_i}{(c_i - b_i\xi_{i+1})}$, i = [n-1; 1]; $\mu_{n-1} = \frac{f_i + b_i\mu_{i+1}}{(c_i - b_i\xi_{i+1})}$ i = [n-1; 0], $x_{i+1} = \xi_{i+1}x_i + \mu_{i+1}$ i = [0; n - 1].

Рассмотрим последний шаг прямого хода, т.е. первое уравнение системы и второе, преобразованное на n-1 шаге:

$\begin{cases} c_0x_0 - b_0x_1 = f_0 \\ -\xi_1x_0 + x_1 = \mu_1 \end{cases}$, в котором мы также зануляем x_1 и выражаем $x_0 = \mu_0$,

где $\mu_0 = \frac{f_0 + b_0\mu_1}{(c_0 - b_0\xi_1)}$.

Таким образом мы получили все формулы левой прогонки.

Обратный ход заключается в подсчёте неизвестных по упомянутой выше реккурентной формуле: $x_0 = \mu_0$, $x_{i+1} = \xi_{i+1}x_i + \mu_{i+1}$ i = [0; n - 1].

Обоснование метода прогонки

Рассмотрев полученные формулы сделаем следующие выводы:

1) В связи с присутствием операции деления метод корректен при $c_i + b_i\xi_{i+1} \neq 0$.

2) Т.к. решение x_{i+1} находится по формуле $x_{i+1} = \xi_{i+1}x_i + \mu_{i+1}$, то погрешность $\varepsilon_{i+1} = y_i' - y_i$ будет удовлетворять уравнению $\varepsilon_{i+1} = \xi_{i+1}\varepsilon_i$ при заданном ε_0. Следовательно, при $\xi_{i+1} > 1$ может произойти сильное увеличение погрешности и при очень больших системах реальное решение y' будет значительно отличаться от решения найденного решения y.

Дабы избежать такой ситуации, метод используется для решения системы, прогоночные коэффициенты ξ_{i+1} которой не превышают единицы. В таком случае должны выполнятся следующие условия:

$|c_0| > 0$, $|c_n| > 0$; $|a_i| > 0$, $|b_i| > 0$, $|c_i| \geq |a_i| + |b_i|$ для i = [n-1, 1]; $|c_0| \geq |b_0|$, $|c_n| \geq |a_n|$.

Тогда $c_i + b_i\xi_{i+1} \neq 0$ и $\xi_{i+1} < 1$.

Для проверки корректности метода прогонки для данной матрицы в программе выведена сама матрица системы, а так же прогоночные коэффициенты ξ_i. В случае с нашей матрицей, условие можно ослабить, позволив некоторым коэффициентам a_i быть равным 0.

Листинг

```
std::vector<std::vector<double>> systemM =
    { {0.7941, 0.0000, 0.0000, 0.0000, 0.0000},
      {-0.0485, 0.5168, 0.0000, 0.0000, 0.0000},
      {0.0000, -0.1454, 0.9367, 0.0178, 0.0000},
      {0.0000, 0.0000, -0.1179, 0.9367, 0.0000},
      {0.0000, 0.0000, 0.0000, -0.0194, 0.6783} };

    std::vector<double> columnN = { 1.5569, 2.0656, -2.9054, -8.0282, 3.4819 };

    std::vector<double> answer = { 0, 0, 0, 0, 0 };
    std::vector<double> xi = { 0, 0, 0, 0, 0 };
    std::vector<double> nu = { 0, 0, 0, 0, 0 };

    //Вычисление прогоночных коэффициентов (прямой ход)
    xi[4] = -systemM[4][3] / systemM[4][4];
    nu[4] = columnN[4] / systemM[4][4];

    for (int i = 3; i > 0; --i) {
        xi[i] = -systemM[i][i - 1] / (systemM[i][i] + xi[i + 1] * systemM[i][i + 1]);
        nu[i] = (columnN[i] - systemM[i][i + 1] * nu[i + 1]) / (systemM[i][i] + xi[i + 1]
* systemM[i][i + 1]);
    }

    nu[0] = (columnN[0] - systemM[0][1] * nu[1]) / (systemM[0][0] + xi[1] *
systemM[0][1]);

    //Нахождение решения (обратный ход)
    answer[0] = nu[0];

    for (int i = 0; i < 4; ++i) {
        answer[i + 1] = xi[i + 1] * answer[i] + nu[i + 1];
    }

    //Посчитаем невязку
    std::vector<double> discrepancy = { 0, 0, 0, 0, 0 };
    for (int i = 0; i < systemM.size(); ++i) {
        for (int j = 0; j < systemM.size(); ++j) {
            discrepancy[i] += systemM[i][j] * answer[j];
        }
        discrepancy[i] -= columnN[i];
    }

    //Кубическая норма невязки
    double cubicNorm = 0;
    for (double item : discrepancy) {
        if (cubicNorm < abs(item)) {
            cubicNorm = abs(item);
        }
    }
```

Выходные данные

```
=================================================System matrix==============================================

        0.7941              0              0              0              0
       -0.0485         0.5168              0              0              0
             0        -0.1454         0.9367         0.0178              0
             0              0        -0.1179         0.9367              0
             0              0              0        -0.0194         0.6783

=================================================Decision vector=================================================

                                ( 1.96058, 4.1809, -2.28442, -8.85826, 4.87992)

=================================================Neural vector===================================================

                                        ( 0, 0, -4.44089e-16, 0, 4.44089e-16)

===================================================Cubic norm====================================================

                                                  4.44089e-16

===============================================Run coefficients xi===============================================

                                         0.0938467 0.154855 0.125867 0.0286009
```

Вывод

Метод прогонки является самым оптимальным среди других точных методов, применимых для трёх-диагональных матриц в связи с довольно малым количеством операций деления и умножения (а именно: 8n - 1) чисел с плавающей точкой, дающее погрешность. В сравнении с методом Гаусса (с выбором главного элемента по матрице) норма вектора невязки аналогичная и приблизительно равна $4*10^{-16}$.

Для матрицы, рассматриваемой нами, метод прогонки является корректным, т.к. выполяются все условия обоснования метода прогонки (если рассмотреть матрицу системы, выполнив указанные выше сравнения, и вектор прогоночных коэффициентов xi).