**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:

, обозначим как систему (1).

Метод левой прогонки основан на идее Гаусса (в вариации «схема единственного деления») и на утверждении, что неизвестные связаны рекуррентным соотношением:   
xi+1 = ξi+1xi + μi+1, i = [0; n - 1].

Введём обозначение прогоночных коэффициентов для первого шага исключения Гаусса:

ξn = , μn = .

Тогда последнее уравнение системы (1) примет вид: .

Рассмотрим два последних уравнения. Домножим последнее на bn-1 и сложим его с предпоследним, дабы исключить из него xn, получив следующее равенство:

Откуда выразим xn-1 = ξn-1xn-2 + μn-1, где ξn-1 = , μn-1 = .

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

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

ξi = , i = [n-1; 1]; μn-1 = i = [n-1; 0], xi+1 = ξi+1xi + μi+1 i = [0; n - 1].

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

, в котором мы также зануляем x1 и выражаем x0 = ,

где μ0 = .

Таким образом мы получили все формулы левой прогонки.

Обратный ход заключается в подсчёте неизвестных по упомянутой выше реккурентной формуле: x0 = , xi+1 = ξi+1xi + μi+1 i = [0; n - 1].

**Обоснование метода прогонки**

Рассмотрев полученные формулы сделаем следующие выводы:

1) В связи с присутствием операции деления метод корректен при ≠ 0.

2) Т.к. решение xi+1 находится по формуле xi+1 = ξi+1xi + μi+1, то погрешность εi+1 = yi’ – yi будет удовлетворять уравнению εi+1 = ξi+1εi при заданном ε0. Следовательно, при ξi+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 ξi+1 которой не превышают единицы. В таком случае должны выполнятся следующие условия:

|c0| > 0, |cn­| > 0; |ai| > 0, |bi| > 0, |ci| ≥ |ai| + |bi| для i = [n-1, 1]; |c0| ≥ |b0|, |cn| ≥ |an|.

Тогда ≠ 0 и ξi+1 < 1.

Для проверки корректности метода прогонки для данной матрицы в программе выведена сама матрица системы, а так же прогоночные коэффициенты ξi. В случае с нашей матрицей, условие можно ослабить, позволив некоторым коэффициентам ai 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) {

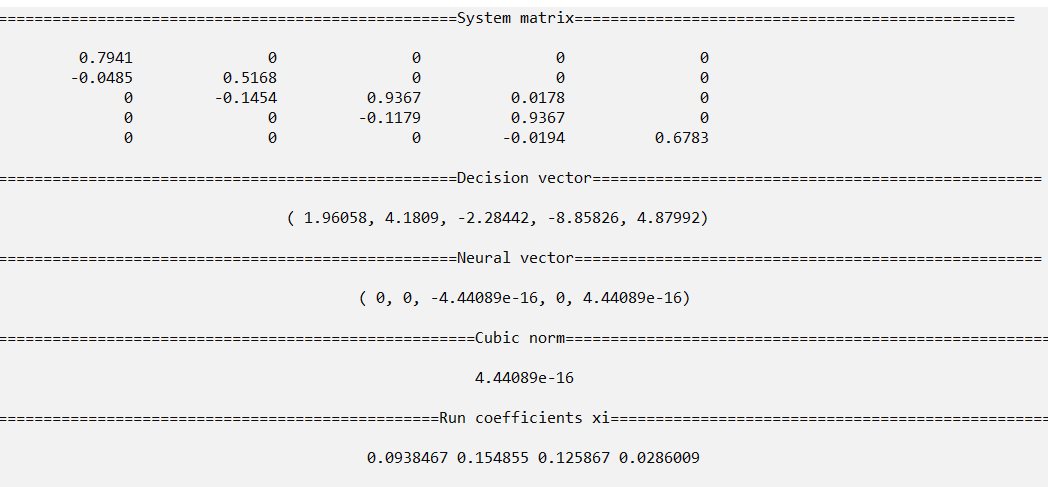
if (cubicNorm < abs(item)) {

cubicNorm = abs(item);

}

}

**Выходные данные**



**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).

**Вывод**