Московский государственный технический университет им. Н.Э. Баумана
Факультет «Радиоэлектроника и лазерная техника (РЛ)»
Кафедра «Технология приборостроения (РЛ6)»

Лабораторная работа №2
"Метод прогонки решения трехдиагональной СЛАУ."
по дисциплине "Численные методы"
Вариант №1

Выполнили студенты группы РЛ6-71
Филимонов С.В.

Преподаватель Чигирева О.Ю.

Москва, 2023

Исходные данные:

СЛАУ:

$$\begin{pmatrix} 50 & 1 & 0 & 0 & 0 & 0 \\ -1 & 90 & 1 & 0 & 0 & 0 \\ 0 & 1 & 125 & -1 & 0 & 0 \\ 0 & 0 & 1 & 110 & 0 & 0 \\ 0 & 0 & 0 & -1 & 85 & 1 \\ 0 & 0 & 0 & 0 & 1 & 70 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 10 \\ -9 \\ 12 \\ 11 \\ 9 \\ 8 \end{pmatrix}$$

Цель работы: изучение метода прогонки решения СЛАУ с трехдиагональной матрицей.

Содержание работы:

1. Реализовать метод прогонки; проверить выполнение достаточных условий применимости метода.
2. Провести решение системы линейных алгебраических уравнений методом прогонки и вычислить норму его невязки (при расчетах пользоваться 1-нормой и inf-нормой).
3. Экспериментально исследовать устойчивость найденного решения к малым возмущениям исходных данных, для чего изменить несколько коэффициентов в правой части на ±0.01, найти решение возмущенной системы и сравнить его с решением невозмущенной системы.

Алгоритм метода прогонки

Метод применяется для решения СЛАУ с трехдиагональной матрицей:

$$\begin{pmatrix} \beta_1 & c_1 & 0 & & & 0 & 0 & 0 \\ \alpha_2 & \beta_2 & c_2 & \cdots & & 0 & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & & \\ 0 & 0 & 0 & & \beta_{n-2} & c_{n-2} & 0 & \\ 0 & 0 & 0 & \cdots & \alpha_{n-1} & \beta_{n-1} & c_{n-1} & \\ 0 & 0 & 0 & & 0 & \alpha_n & \beta_n & \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-2} \\ d_{n-1} \\ d_n \end{pmatrix}$$

Выразим из 1-ого уравнения x_1:

$$x_1 = \alpha_1 \cdot x_2 + \beta_1, \alpha_1 = -\frac{c_1}{\beta_1}, \beta_1 = -\frac{d_1}{\beta_1};$$

Подставим x_1 во 2-ое уравнение и выразим из него x_2:

$$x_2 = \alpha_2 \cdot x_3 + \beta_2, \alpha_2 = -\frac{c_2}{\beta_2 + \alpha_2 \cdot \alpha_1}, \beta_1 = -\frac{d_2 - \alpha_2 \beta_1}{\beta_2 + \alpha_2 \alpha_1};$$

Подставим x_{n-1} в n-е уравнение и выразим из него x_n:

$$x_n = \beta_n = \frac{d_n - \alpha_n \beta_{n-1}}{\beta_n + \alpha_n \alpha_{n-1}}.$$

Алгоритм метода

Прямая прогонка: вычисление прогонки k-ов:

$$\alpha_k, k = \overline{1, n-1} \text{ и } \beta_k, k = \overline{1, n}$$

По следующим формулам:

$$\mathrm{k} = 1: \gamma_1 = \beta_1, \alpha_1 = -\frac{\mathrm{c}_1}{\gamma_1}, \beta_1 = -\frac{\mathrm{d}_1}{\gamma_1};$$

$$\mathrm{k} = \overline{2, \mathrm{n}-1}: \gamma_\mathrm{k} = \beta_\mathrm{k} + \alpha_\mathrm{k}\alpha_{\mathrm{k}-1}, \alpha_\mathrm{k} = -\frac{\mathrm{c}_\mathrm{k}}{\gamma_\mathrm{k}}, \beta_\mathrm{k} = -\frac{\mathrm{d}_\mathrm{k} - \alpha_\mathrm{k}\beta_{\mathrm{k}-1}}{\gamma_\mathrm{k}};$$

$$\mathrm{k} = \mathrm{n}: \gamma_\mathrm{n} = \beta_\mathrm{n} + \alpha_\mathrm{n}\alpha_{\mathrm{n}-1}, \beta_\mathrm{n} = -\frac{\mathrm{d}_\mathrm{n} - \alpha_\mathrm{n}\beta_{\mathrm{n}-1}}{\gamma_\mathrm{n}}.$$

Обратная прогонка: вычисление неизвестных $x_n, x_{n-1}, \ldots, x_1$ по формулам:

$$x_n = \beta_n, x_k = \beta_k + \alpha_k x_{k+1}, k = n-1, \ldots, 1.$$

Теорема (достаточное условие приминения метода прогонки).

Пусть к-т СЛАУ удовлетворяет условиям диагонального проеобладания $|\beta_k| \geq |\alpha_k| + |c_k|, k = \overline{1, n}$, причем хотя бы для одного k выполнено строгое неравенство. Тогда алгоритм метода прогонки корректно ($\gamma_k \neq 0, \forall k = \overline{1, n}$) и устойчив ($|\alpha_k| \leq 1, \forall k = \overline{1, n}$).

Вывод:

В данной работе мы реализовали метод прогонки и проверку выполнения достаточных условий применимости метода. Далее с помощью написанной программы мы нашли решение исходной СЛАУ.

В результате выполнения программы мы получили следующее решение СЛАУ:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 0.20197680 \\ -0.0988400 \\ 0.09758362 \\ 0.09911287 \\ 0.10572161 \\ 0.11277540 \end{pmatrix}$$

Вычислили невязку и её нормы (1-норму и inf-норму):

$$r_0 = d_0 - d_0^* = d_0 - A \cdot x^* = \begin{pmatrix} 0 \\ 0 \\ -1.7763568 \cdot 10^{-15} \\ 0 \\ 1.7763568 \cdot 10^{-15} \\ 0 \end{pmatrix}$$

$$\|r_0\|_1 = 3.5527136 \cdot 10^{-15};\ \|r_0\|_\infty = 1.7763568 \cdot 10^{-15}$$

Также мы экспериментально исследовали устойчивость решения к малым возмущениям исходных данных, для чего изменили несколько коэффициентов в правой части на ∓ 0.01:

$$d_1 = \begin{pmatrix} 10.01 \\ 9 \\ 12.01 \\ 10.99 \\ 9.01 \\ 8 \end{pmatrix}$$

В результате получаем следующее решение СЛАУ:

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \\ x_4' \\ x_5' \\ x_6' \end{pmatrix} = \begin{pmatrix} 0.20217677 \\ -0.09883873 \\ 0.09766287 \\ 0.09902124 \\ 0.10583820 \\ 0.11277374 \end{pmatrix}$$

Вычислим относительную погрешность полученного решения возмущенной и невозмущенной СЛАУ:

$$\frac{\|x - x'\|_1}{\|x\|_1} = 5.36 \cdot 10^{-4}$$

Найденная относительная погрешность СЛАУ мала, что может говорить о высокой устойчивости СЛАУ.

Текст программы:

```
static Matrix getSleForThroughStraight(const Matrix& a, const Matrix& b, const Matrix& c, const Matrix& y) {
    Matrix sle(b.getRows(), 4);
    const size_t ROW = sle.getRows();

    sle(0, 0) = 0; sle(0, 1) = b(0, 0); sle(0, 2) = c(0, 0); sle(0, 3) = y(0, 0);
    for (size_t i = 1, _row = (ROW - 1); i < _row; ++i) {
        sle(i, 0) = a((i - 1), 0); sle(i, 1) = b(i, 0); sle(i, 2) = c(i, 0); sle(i, 3) = y(i, 0);
    }
    sle((ROW - 1), 0) = a((ROW - 2), 0);
    sle((ROW - 1), 1) = b((ROW - 1), 0);
    sle((ROW - 1), 3) = y((ROW - 1), 0);
    return sle;
}

static bool checkConditionThroughStraight(const Matrix& sle) {
    for (size_t i = 0; i < sle.getRows(); ++i)
        if((std::abs(sle(i, 1)) < (std::abs(sle(i, 0)) + std::abs(sle(i, 2)))))
            return false;
    return true;
}

static Matrix SLEmethodThroughStraightRunning(const Matrix& sle) {
    const size_t ROW = sle.getRows();
    Matrix albe(ROW, 2);

    //// Для i = 0
    albe(0, 0) = (-sle(0, 2) / sle(0, 1)); albe(0, 1) = (sle(0, 3) / sle(0, 1));

    //// Для i = 1:(n - 1)
    double yi = 0.0;
    for (size_t i = 1; i < (ROW - 1); ++i) {
        yi = (sle(i, 1) + (sle(i, 0) * albe((i - 1), 0)));
        albe(i, 0) = (-sle(i, 2) / yi);
        albe(i, 1) = ((sle(i, 3) - (sle(i, 0) * (albe((i - 1), 1)))) / yi);
    }

    //// Для i = n
    yi = (sle((ROW - 1), 1) + (sle((ROW - 1), 0) * albe((ROW - 2), 0)));
    albe((ROW - 1), 1) = ((sle((ROW - 1), 3) - (sle((ROW - 1), 0) * albe((ROW - 2), 1))) / yi);

    return albe;
}

static Matrix SLEmethodThroughReverseCourse(const Matrix& sle, const Matrix& albe) {
    Matrix x(sle.getRows(), 1);
    x((sle.getRows() - 1), 0) = albe((sle.getRows() - 1), 1);
    for (size_t i = (sle.getRows() - 2); i < (sle.getRows()); --i)
        x(i, 0) = (albe(i, 1) + (albe(i, 0) * x((i + 1), 0)));
```

```
    return x;
}

Matrix SLEmethodRunThrough(const Matrix& a, const Matrix& b, const Matrix& c, const Matrix& y) {
    const Matrix sle = getSleForThroughStraight(a, b, c, y);

    if (!checkConditionThroughStraight(sle))  // Достаточной условие примененния матода прогонки
        throw NonFulfillmentOfConditions();

    const Matrix albe = SLEmethodThroughStraightRunning(sle);  // Прямой ход
    const Matrix x = SLEmethodThroughReverseCourse(sle, albe);  // Обратный ход
    return x;
}
```