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:

**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:

Выразим из 1-ого уравнения :

Подставим во 2-ое уравнение и выразим из него :

Подставим в n-е уравнение и выразим из него :

**Алгоритм метода**

*Прямая прогонка:* вычисление прогонки k-ов:

По следующим формулам:

*Обратная прогонка:* вычисление неизвестных по формулам:

*Теорема (достаточное условие приминения метода прогонки).*

Пусть к-т СЛАУ удовлетворяет условиям диагоняльного проеобладания , причем хотя бы для одного k выполнено строгое неравенство. Тогда алгоритм метода прогонки корректно () и устойчив ().

**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:

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

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

В результате получаем следующее решение СЛАУ:

Вычислим относительную погрешность полученного решения возмущенной и невозмущенной СЛАУ:

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 SLEmethodThroughReverseСourse(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 = SLEmethodThroughReverseСourse(sle, albe); // Обратный ход**

**return x;**

**}**