Практическая задача, пункт а

Натальченко Александр

11 ноября 2020 г.

Содержание

1. Постановка задачи
2. Краткое описание алгоритма
3. Результаты
4. Код программы

1 Постановка задачи

12. ПРАКТИЧЕСКАЯ ЗАДАЧА 3 курс 5 семестр

Решить систему

$$\begin{cases} c_1x_1 + d_1x_2 + e_1x_3 & = & f_1 \\ b_2x_1 + c_2x_2 + d_2x_3 + e_2x_4 & = & f_2 \\ a_3x_1 + b_3x_2 + c_3x_3 + d_3x_4 + e_3x_5 & = & f_3 \\ \quad a_4x_2 + b_4x_3 + c_4x_4 + d_4x_5 + e_4x_6 & = & f_4 \\ \quad \cdots & \cdots & \cdots \\ \quad\quad a_mx_{m-2} + b_mx_{m-1} + c_mx_m + d_mx_{m+1} + e_mx_{m+2} & = & f_m \\ \quad\quad \cdots & \cdots & \cdots \\ \quad\quad\quad a_{n-1}x_{n-3} + b_{n-1}x_{n-2} + c_{n-1}x_{n-1} + d_{n-1}x_n & = & f_{n-1} \\ \quad\quad\quad\quad a_nx_{n-2} + b_nx_{n-1} + c_nx_n & = & f_n \end{cases}, \quad n = 20, \ c_i = 10, \ f_i = i, \ i = 1, \dots, n;$$

$b_{i+1} = d_i = 1$, $i = 1, \dots, n-1$; $a_{i+2} = e_i = 0$, $i = 1, \dots, n-2$.

a. **прямым методом (методом Гаусса):**
вывести на печать решение и невязки;

b. **итерационным методом (методом Зейделя) с точностью** $\varepsilon = 10^{-4}$:
вывести на печать решение;

c. **определить число обусловленности матрицы системы** $\mu = \|A\| \cdot \|A^{-1}\|$

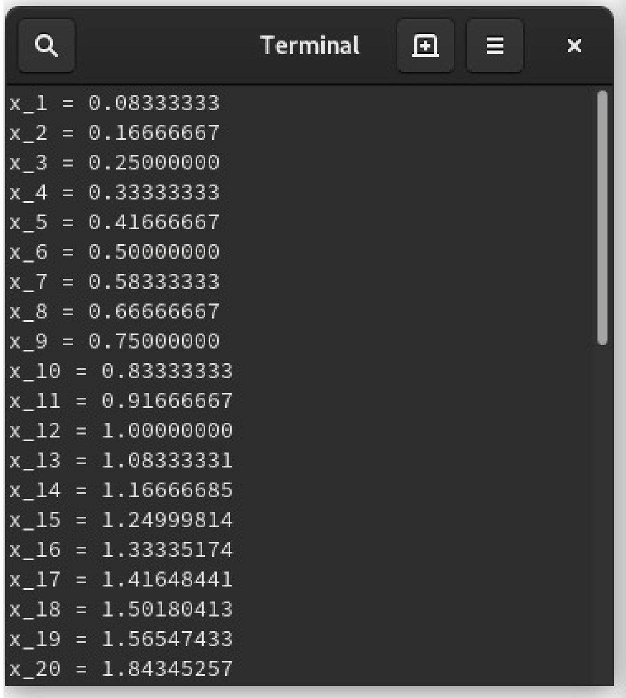
(УКАЗАНИЕ: найти $\lambda_{\max}$ степенным методом, затем $\lambda_{\min}$, используя сдвиг спектра матрицы, для определения числа обусловленности воспользоваться евклидовой нормой).

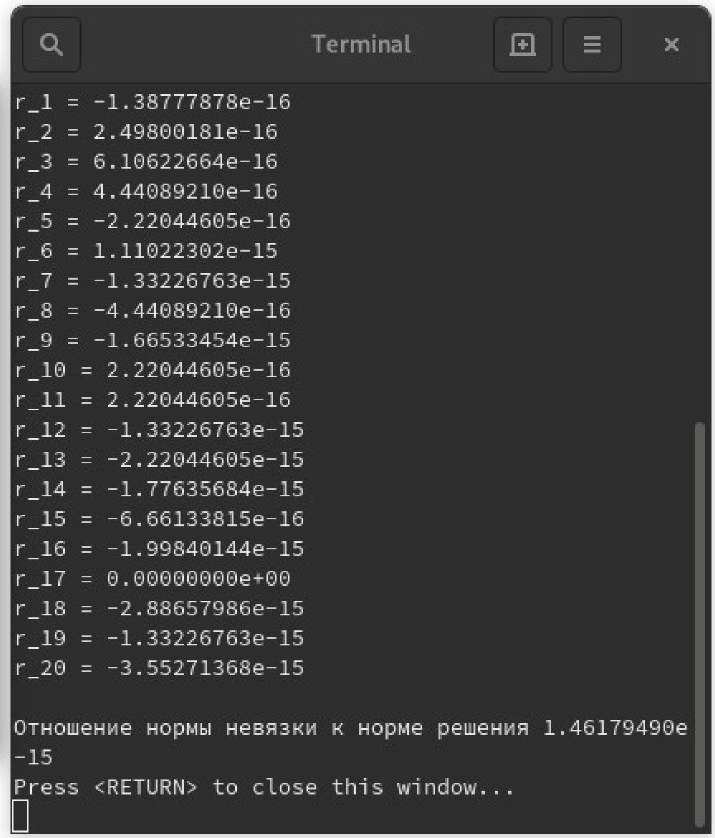
2 Краткое описание алгоритма

Сначала задаём значения элементов расширенной матрицы системы (с. 21-30), в матрице A_0 мы будем хранить её, а в матрице A будем её преобразовывать.

Прямым ходом метода Гаусса (с. 34-65) мы «обнуляем» элементы под нижней диагональю матрицы, затем обратным ходом (с. 67-73) «обнуляем» — под верхней. Причём в обратном ходе вовсе не обязательно (с. 71) преобразовывать верхнюю часть матрицы, так как это ни на что не влияет. Достаточно работать только с последним столбцом (с. 70). Затем получаем вектор-решение (с. 75-81), вектор-невязку (с. 83-92) и считаем отношение их норм (с. 95-100).

3 Результаты





Мы получили решение и невязки, и вычислили отношение нормы невязки к норме решения:

$$\frac{\|r\|}{\|x^*\|} = 1.4618 \cdot 10^{-15}$$

Поскольку число обусловленности матрицы системы, которое требовалось рассчитать в другом пункте, равно $\mu = 1.48 \sim 1$, то следует ожидать, что отношение нормы невязки к норме решения по порядку совпадёт с машинным эпсилоном:

$$\frac{\|r\|}{\|x^*\|} \sim \varepsilon_m$$

Для типа double в языке C++ величина машинного эпсилона составляет $\varepsilon = 2^{-52}$

И действительно,

$$\frac{1.4618 \cdot 10^{-15}}{2^{-52}} \approx 6.6,$$

значит, как и ожидалось,

$$\frac{\|r\|}{\|x^*\|} \approx 7\varepsilon_m.$$

4 Код программы (C++)

```
#include <iostream>
#include <vector>
#include <cmath>
#include <iomanip>

const double c_i = 10; // Значения элементов матрицы на главной диагонали
const double d_i = 1; // Значения элементов матрицы на соседних диагоналях
const size_t N = 20; // Размер матрицы
const double epsilon = 1e-10;

using namespace std;

inline bool equals0(double x) {
    return(fabs(x) < epsilon);
}

int main() {
    vector<vector<double> > A_0(N), A(N);

    // Задаём значения элементов матрицы
    for (size_t i = 0; i < N; i++) {
        A_0[i].resize(N + 1);
        A[i].resize(N + 1);
        A_0[i][i] = c_i;
        if (i < N - 1)
            A_0[i][i + 1] = d_i;
        if (i > 0)
            A_0[i][i - 1] = d_i;
        A_0[i][N] = i + 1;
    }

    A = A_0;

    //Прямой ход метода Гаусса
    for (size_t i = 0; i < N; i++) {

        //Если диагональный элемент равен нулю с точностью 1e-10,
        //пытаемся переставить строки
        if (equals0(A[i][i])) {
            vector<double> tmp(N + 1);
            tmp = A[i];
            for (size_t j = i + 1; j < N; j++) {
                if (!equals0(A[j][i])) {
                    A[i] = A[j];
                    A[j] = tmp;
                }
            }
        }

        if (equals0(A[i][i])) {
            return 0;
        } else { //Если диагональный элемент не равен нулю
            //Делим элементы строки на диагональный элемент
            for (size_t j = N; j >= i && j <= N; j--) {
```

```
                A[i][j] /= A[i][i];
            }

            // Вычитаем эту строку из нижних с коэффициентами A[i][j]
            for (size_t j = i + 1; j < N; j++) {
                for (size_t k = N; k >= i && k + 1 != 0; k--) {
                    A[j][k] -= A[i][k]*A[j][i];
                }
            }
        }
    }

    //Обратный ход метода Гаусса
    for (size_t j = N - 1; j + 1 != 0; j--) {
        for (size_t i = j - 1; i + 1 != 0; i--) {
            A[i][N] -= A[j][N]*A[i][j];
            //A[i][j] = 0;
        }
    }

    //Выводим решение
    vector<double> X(N);
    for (size_t i = 0; i < N; i++) {
        X[i] = A[i][N];
        cout << fixed << setprecision(8)
                  << "x_" << i + 1 << " = " << X[i] << endl;
    }

    //Выводим невязки
    vector<double> r(N);
    for (size_t i = 0; i < N; i++) {
        r[i] = A_0[i][N];
        for (size_t j = 0; j < N; j++) {
            r[i] -= A_0[i][j]*X[j];
        }
        cout << "r_" << i + 1 << " = " << scientific
                  << r[i] << endl;
    }
    cout << endl;

    //Считаем квадраты норм
    double NX = 0, Nr = 0;
    for (size_t i = 0; i < N; i++) {
        NX += X[i]*X[i];
        Nr += r[i]*r[i];
    }

    cout << "Отношение нормы невязки к норме решения " << sqrt(Nr/NX) << endl;
    return 0;
}
```