МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

Первое практичское задание по курсу лекций «Численные методы линейной алгебры»

ОТЧЕТ

о выполненном задании

студента 301 учебной группы факультета ВМК МГУ

Мартьянова Артема Олеговича

гор. Москва

2023 год

Содержание

Постановка задачи

Дана система уравнений $Ax = f$, где $A \in \mathcal{R}^{n\times n}$ - невырожденная квадратная матрица. Элементы матрицы $a_{i,j}$ являются вещественными числами, расположенными на отрезке $[-1, 1]$. Матрица предоставляется в виде файла в формате csv.
Требуется написать программу на языке программирования C (или C++), реализующую метод решения СЛАУ с помощью LU-разложения матрицы A. Также нужно определить время, затраченное на вычисление решения, найти погрешность решения и вычислить её максимум-норму.

Описание метода решения задачи

Покажем, что метод Гаусса эквивалентен разложению матрицы A в произведение нижней L и верхней U треугольных матриц с последующим решением вспомогательных систем с этими матрицами. Рассмотрим подробнее на примере n = 3. Введем матрицы

$$E_{21} = \begin{bmatrix} 1 & 0 & 0 \\ -l_{21} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad E_{31} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -l_{31} & 0 & 1 \end{bmatrix}, \qquad L_2^{-1} := E_{32} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -l_{32} & 1 \end{bmatrix}$$

где

$$l_{ik} = a_{ik}^{(k-1)}/a_{kk}^{(k-1)}, i = k+1, ..., n.$$

Введем некоторые обозначения:

$$\begin{cases} a_{11}^{(0)}x_1 + a_{12}^{(0)}x_2 + a_{13}^{(0)}x_3 = b_1^{(0)} \\ a_{21}^{(0)}x_1 + a_{22}^{(0)}x_2 + a_{23}^{(0)}x_3 = b_2^{(0)} \\ a_{31}^{(0)}x_1 + a_{32}^{(0)}x_2 + a_{33}^{(0)}x_3 = b_3^{(0)} \end{cases} \tag{1}$$

$$\begin{cases} a_{11}^{(0)}x_1 + a_{12}^{(0)}x_2 + a_{13}^{(0)}x_3 = b_1^{(0)} \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 = b_2^{(1)} \\ a_{31}^{(0)}x_1 + a_{32}^{(0)}x_2 + a_{33}^{(0)}x_3 = b_3^{(0)} \end{cases} \tag{2}$$

где

$$a_{2j}^{(1)} = a_{2j}^{(0)} - l_{21}a_{1j}^{(0)}, l_{21} = a_{21}^{(0)}/a_{11}^{(0)}, b_2^{(1)} = b_2^{(0)} - l_{21}b_1^{(0)},$$

$$\begin{cases} a_{11}^{(0)}x_1 + a_{12}^{(0)}x_2 + a_{13}^{(0)}x_3 = b_1^{(0)} \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 = b_2^{(1)} \\ a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 = b_3^{(1)} \end{cases} \tag{3}$$

Легко проверить, что переход от системы (1) к системе (2) может быть осуществлен умножением матрицы $A^{(0)}$ системы(1) и ее правой части $b^{(0)}$ на матрицу E_{21}, а от (2) к (3) - умножением соотвествующей матрицы и вектора на матрицу E_{31}. Очевидно также, что

$$L_1^{-1} := E_{31}E_{21} = E_{21}E_{31} = \begin{bmatrix} 1 & 0 & 0 \\ -l_{21} & 1 & 0 \\ -l_{31} & 0 & 1 \end{bmatrix}$$

и, следовательно, переход от (1), минуя (2), сразу к (3) осуществляется умножением матрицы системы (1) и ее правой части на L_1^{-1}. Аналогично (далее уже только о матрицах)

$$A^{(2)} = L_2^{-1}A^{(1)} = L_2^{-1}L_1^{-1}A^{(0)},$$

где $A^{(k)}$ - матрицы, получаемые на $k-$ом шаге, и, следовательно,

$$A = A^{(0)} = L_1L_2A^{(2)}. \tag{4}$$

Легко проверить, что

$$L_1 = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & 0 & 1 \end{bmatrix}, \qquad L_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & l_{32} & 1 \end{bmatrix},$$

т.е. при обращении L_k меняется только знак перед поддиагональными элементами $l_{ik}, i > k$. Далее,

$$L := L_1L_2 = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix},$$

а матрица $A^{(2)}$ из (4) есть верхняя треугольная матрица. Обозначая ее через U и принимая во внимание вышесказанное, приходим к искомому расзложению

$$A = LU, \tag{5}$$

где L - нижняя треугольная матрица с единичной главной диагональю.
Все вышесказанное остается справедливым и в общем случае матрицы A порядка n. Теперь матрицы L и U из (5) имеют вид

$$L = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & l_{n3} & \dots & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & u_{nn} \end{bmatrix}, \tag{6}$$

где элементы l_{ik} матрицы L вычисляются по формулам

$$l_{ik} = a_{ik}^{(k-1)}/a_{kk}^{(k-1)}, \qquad i = k+1, \dots, n,$$

а элементы $u_{kj} := a_{kj}^{(k-1)}$ матрицы U по формулам

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - l_{ik}a_{kj}^{(k-1)}, \qquad i, j = k+1, \dots, n.$$

Имея разложение (5), систему можно переписать в виде

$$Ax = LUx = Ly = b, \qquad Ux = y,$$

после чего решение системы распадается на решение двух систем с треугольными матрицами

$$Ly = b \qquad \text{и} \qquad Ux = y.$$

Решение первой их этих систем заменяет преобразования правой части прямого хода метода Гаусса по формулам

$$b_i^{(k)} = b_i^{(k-1)} - l_{ik} b_k^{(k-1)}, \qquad i = k+1, \ldots, n.$$

Решение же второй системы определяется формулами обратной подстановки

$$x_i = \left[b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j \right] \Bigg/ \quad a_{ii}^{(i-1)}, \qquad i = n, \ldots, 1,$$

где $b_i^{(i-1)} = y_i$, а $a_{ij}^{(i-1)} = u_{ij}$.

Описание программы

Здесь приведем описание основных функций программы(помимо них были реализованы вспомогательные для более удобной отладки, на них мы обращать внимания не будем).

- ```
  void LU_decomposition(std::vector<std::vector<double>>& A, std::vector<std::vector<double>>& L, std::vector<std::vector<double>>& U)
  ```

  Функция принимает в себя ссылки на векторы, выполняет LU-разложение матрицы A, результаты складывает в соответствующие векторы и ничего не возвращает.

- ```
  std::vector<double> solve_system(std::vector<std::vector<double>>& L, std::vector<std::vector<double>>& U, std::vector<double>& b)
  ```

  Функция решает систему уравнений с треугольными матрицами способом, описанным ранее. Возвращает вектор-решение.

- ```
  std::vector<double> matrix_vector_multiply(const std::vector<std::vector<double>>& matrix, const std::vector<double>& vector)
  ```

  Функция, умножающая матрицу на вектор-столбец. Возвращает вектор-результат.

- ```
  std::vector<std::vector<double>> read_csv(const std::string& filename)
  ```

  Функция, считывающая матрицу из csv файла. Возвращает представление матрицы в виде вектора векторов.

- ```
  double max_norm(const std::vector<double> v)
  ```

  Функция для вычисления нормы. Возвращает посчитанное значение.

- ```
  std::vector<double> generate_random_vect(int s)
  ```

  Функция для генерации случайного вектора с равномерно распределенными на отрезке $[0,1]$ компонентами $x_i, i = 1, 2, \ldots, n$.

Код программы

```cpp
#include <iostream>
#include <vector>
#include <fstream>
#include <sstream>
#include <string>
#include <cmath>
#include <chrono>

enum
{
    LEFT_BOUND = -1,
    RIGHT_BOUND = 1
};

std::ostream& operator<<(std::ostream &out, const std::vector<std::
    vector<double>> &v) {
    for (auto &it : v) {
        for (auto &it2 : it) {
            out << it2 << " ";
        }
        out << "\n";
    }
    return out;
}

std::ostream& operator<<(std::ostream &out, const std::vector<double> &v
    ) {
    for (auto &it : v) {
        out << it << " ";
    }
    return out;
}

void LU_decomposition(std::vector<std::vector<double>>& A, std::vector<
    std::vector<double>>& L, std::vector<std::vector<double>>& U)
{
    int n = A.size();
    L.resize(n, std::vector<double>(n, 0));
    U.resize(n, std::vector<double>(n, 0));
    for (int i = 0; i < n; i++) {
        L[i][i] = 1;
    }
    for (int k = 0; k < n; k++) {
        U[k][k] = A[k][k];
        for (int i = k + 1; i < n; i++) {
            L[i][k] = A[i][k] / U[k][k];
            U[k][i] = A[k][i];
        }
        for (int i = k + 1; i < n; i++) {
            for (int j = k + 1; j < n; j++) {
                A[i][j] = A[i][j] - L[i][k] * U[k][j];
            }
        }
    }
}
```

```
std::vector<double> solve_system(std::vector<std::vector<double>>& L, std::vector<std::vector<double>>& U, std::vector<double>& b)
{
    int n = L.size();
    std::vector<double> y(n, 0);
    for (int i = 0; i < n; i++) {
        double sum = 0;
        for (int j = 0; j < i; j++) {
            sum += L[i][j] * y[j];
        }
        y[i] = b[i] - sum;
    }
    std::vector<double> x(n, 0);
    for (int i = n - 1; i >= 0; i--) {
        double sum = 0;
        for (int j = i + 1; j < n; j++) {
            sum += U[i][j] * x[j];
        }
        x[i] = (y[i] - sum) / U[i][i];
    }
    return x;
}

std::vector<double> matrix_vector_multiply(const std::vector<std::vector<double>>& matrix, const std::vector<double>& vector)
{
    int rows = matrix.size();
    int cols = matrix[0].size();
    std::vector<double> result(rows, 0);
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i] += matrix[i][j] * vector[j];
        }
    }
    return result;
}

std::vector<std::vector<double>> read_csv(const std::string& filename)
{
    std::vector<std::vector<double>> matrix;
    std::ifstream file(filename);
    if (!file.is_open()) {
        std::cerr << "Error when try to open file" << std::endl;
        return matrix;
    }
    std::string line;
    while (std::getline(file, line)) {
        std::vector<double> row;
        std::stringstream ss(line);
        std::string cell;
        while (std::getline(ss, cell, ',')) {
            row.push_back(std::stod(cell));
        }
        matrix.push_back(row);
    }
    return matrix;
```

```
}

double max_norm(const std::vector<double> v)
{
    double max = fabs(v.at(0));
    for (auto &it : v) {
        max = (max > fabs(it)) ? max : fabs(it);
    }
    return max;
}

std::vector<double> operator-(const std::vector<double> &v1, const std::vector<double> &v2)
{
    if (v1.size() != v2.size()) throw "Incorrect sizes!!!\n";
    std::vector<double> ans(v2.size());
    for (int i = 0; i < v1.size(); ++i) {
        ans[i] = v1[i] - v2[i];
    }
    return ans;
}

std::vector<double> generate_random_vect(int s)
{
    std::vector<double> v(s);
    for (int i = 0; i < v.size(); ++i) {
        v[i] = double(rand()) * (RIGHT_BOUND - LEFT_BOUND) / RAND_MAX + LEFT_BOUND;
    }
    return v;
}

int main()
{
    std::string filename = "./SLAU_var_2.csv";
    std::vector<std::vector<double>> A = read_csv(filename);

    // LU-decomposion
    std::vector<std::vector<double>> L;
    std::vector<std::vector<double>> U;
    std::vector<std::vector<double>> tmp_A = A;
    LU_decomposition(tmp_A, L, U);

    // solution generation
    std::vector<double> x = generate_random_vect(A.size());

    // Computing right part of the system
    std::vector<double> f = matrix_vector_multiply(A, x);

    std::chrono::steady_clock::time_point begin = std::chrono::steady_clock::now();
    // Solving system
    std::vector<double> x_computed = solve_system(L, U, f);
    std::chrono::steady_clock::time_point end = std::chrono::steady_clock::now();

    std::cout << "||x_true - x_computed|| = " << max_norm(x - x_computed
```

```
    ) << std::endl;
    std::cout << "time in microseconds spent to find solution: " <<
    std::chrono::duration_cast<std::chrono::microseconds>(end - begin).
    count() << std::endl;

    return 0;
}
```

Листинг 1: main.cpp

Полученные результаты

На данной матрице у меня получились следующие результаты:

- Максимум-норма погрешности $\approx 1.1 * 10^{-15}$
- Примерное время выполнения – 60 микросекунд ≈ 0.06 миллисекунды