Федеральное государственное автономное образовательное
учреждение высшего образования Национальный
исследовательский университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №1
Вычислительная математика

Вариант: 17

Группа	P3208
Студент	Щетинин С.В.
Преподаватель	Машина Е.А.

1 Цель работы

1. Изучить прямые и итерационные методы решения систем линейных алгебраических уравнений
2. Выполнить программную реализацию методов

2 Описание используемого метода

В данной лабораторной работе использовался метод Гаусса с выбором главного элемента по столбцам для поиска решений СЛАУ.

Схема с выбором главного элемента является одной из модификаций метода Гаусса.
Среди ведущих элементов могут оказаться очень маленькие по абсолютной величине. При делении на такие ведущие элементы получается большая погрешность округления (вычислительная погрешность). Идеей метода Гаусса с выбором главного элемента является такая перестановка уравнений, чтобы на k-ом шаге исключения ведущим элементом a_{ii} оказывался наибольший по модулю элемент k-го столбца. Т.е. на очередном шаге k в уравнениях, начиная от k до последнего ($i = k, k+1, \ldots, n$) в столбце k выбирают максимальный по модулю элемент и строки i и k меняются местами. Это выбор главного элемента «по столбцу».

3 Расчетные формулы метода

1. Простейшие преобразования матриц
2. Невязка

$$r = Ax^* - b$$

, где A - исходная матрица, x^* - вектор решений методом Гаусса, b - правая часть уравнения

4 Листинг программы

Программа написана на C++. Реализован класс матрицы.
Matrix.h (интерфейс):

```
class Matrix
{
private:
    vector <vector <long double>> mat_;
    vector <vector <long double>> t_mat_;
    vector <long double> res_;
    vector <long double> res_immut_;
    vector <long double> x_;
    vector <long double> r_;
    size_t size_;
    void setMat(const vector <vector<long double>>&);
    void setRes(const vector <long double>&);
    void setImmutRes(const vector <long double>&);
    void generateXandR();
    size_t getPosMaxFromColumn(size_t num_column);
    void swapRow(size_t first_row, size_t second_row);
    void setZeroColumn(int column);
    static inline long double EPS = 1e-8;
public:
    explicit Matrix(vector <vector<long double>>&, const vector <long double>&);
    [[nodiscard]] vector<vector<long double>> getMat() const;
    [[nodiscard]] vector<long double> getRes() const;
    [[nodiscard]] size_t getSize() const;
    void generateTrianMat();
    [[nodiscard]] vector <vector <long double>> getTrianMat() const;
    vector <long double> getX();
    vector <long double> getR();
    long double getDetTrianMatrix();
    void methodGauss();
};

std::ostream& operator << (std::ostream &os, const Matrix& matrix);
std::ostream& operator << (std::ostream &os, const vector <long double> &v);
```

Листинг 1: Matrix.h

Matrix.cpp (Реализация):

```
#include <ostream>
#include <iomanip>
#include <iostream>
#include <algorithm>
#include <map>
#include "Matrix.h"

using namespace std;

Matrix::Matrix(vector <vector<long double>> &new_mat, const vector <long double> &res) {
    this->setMat(new_mat);
    this->setRes(res);
    this->setImmutRes(res);
    this->size_ = new_mat.size();
}

void Matrix::setMat(const vector<vector<long double>> &new_mat) {
    this->mat_ = new_mat;
    this->t_mat_ = new_mat;
}

void Matrix::setRes(const vector<long double> &res) {
    this->res_ = res;
}

void Matrix::setImmutRes(const vector<long double> &res) {
    this->res_immut_ = res;
}

vector<vector<long double>> Matrix::getMat() const {
    return this->mat_;
}

vector<long double> Matrix::getRes() const {
    return this->res_;
}

size_t Matrix::getSize() const {
    return this->size_;
}


size_t Matrix::getPosMaxFromColumn(size_t num_column) {
    size_t pos_max = num_column;
    for (size_t i = num_column; i < this->size_; ++i){
        if (abs(this->t_mat_[pos_max][num_column]) < abs(this->t_mat_[i][num_column])) {
            pos_max = i;
        }
    }
    return pos_max;
}

void Matrix::swapRow(size_t first_row, size_t second_row) {
    swap(t_mat_[first_row], t_mat_[second_row]);
    swap(res_[first_row], res_[second_row]);
}

void Matrix::setZeroColumn(int column) {
    if (this->t_mat_[column][column] == 0) return;
    for (int i = column + 1; i < this->size_; ++i){
        if (this->t_mat_[i][i] == 0) continue;
        long double cf =  - this->t_mat_[i][column] / this->t_mat_[column][column];
        for (int j = column; j < this->size_; ++j){
            this->t_mat_[i][j] += this->t_mat_[column][j] * cf;
        }
        this->res_[i] += this->res_[column] * cf;
    }
}

//only for triangle form
long double Matrix::getDetTrianMatrix() {
    long double res = 1;
    for (int i = 0; i < this->size_; ++i){
        res *= this->t_mat_[i][i];
```

```
    }
    return res;
}

void Matrix::generateTrianMat() {
    for (int j = 0; j < this->size_ - 1; ++j){
        cout << "Turn column #" << j << " to zero:" << endl;
        size_t pos_max = this->getPosMaxFromColumn(j);
        this->swapRow(pos_max, j);
        this->setZeroColumn(j);
        cout << *this << endl;
    }
}

void Matrix::generateXandR() {
    for (int i = (int)this->size_ - 1; i >= 0; --i){
        long double left_sum = 0;
        for (size_t j = i + 1; j < this->size_; ++j){
            left_sum += this->t_mat_[i][j] * this->x_[this->x_.size() - (j - i - 1) - 1];
        }
        long double right_part = this->res_[i] - left_sum;
        this->x_.push_back(right_part/this->t_mat_[i][i]);
        this->r_.push_back(this->x_.back() * this->t_mat_[i][i] + left_sum - this->res_[i]);
    }
    reverse(this->x_.begin(), this->x_.end());
}

vector<long double> Matrix::getX() {
    if (this->x_.empty()) generateXandR();
    return this->x_;
}

vector<long double> Matrix::getR() {
    if (this->r_.empty()) generateXandR();
    return this->r_;
}

vector <vector <long double>> Matrix::getTrianMat() const {
    return this->t_mat_;
}

void Matrix::methodGauss() {
    cout << "The original matrix:" << endl;
    cout << *this << endl;

    this->generateTrianMat();
    long double det = this->getDetTrianMatrix();
    cout << "The determinant of a triangular matrix: " << det << endl;

    if (abs(det) < EPS){
        map <vector <long double>, long double> mp;
        for (size_t i = 0; i < size_; ++i) {
            vector <long double> v = this->mat_[i];
            long double cf = *v.begin();
            for (auto &x : v){
                x /= cf;
            }
            long double res = res_immut_[i] / cf;
            if (mp.find(v) == mp.end()) {
                mp[v] = res;
            } else if (res != mp[v]) {
                cout << "Vector X: empty" << endl;
                return;
            }
        }
        cout << "Vector X: inf" << endl;
        return;
    }

    cout << "Vector X: " << this->getX() << endl;
    cout << "Vector R: " << this->getR() << endl;
}


std::ostream& operator << (std::ostream &os, const Matrix& matrix)
{
```

```
152     int num_res = 0;
153     os << fixed; os.precision(3);
154     for (const vector<long double> &i : matrix.getTrianMat()){
155         for (long double j : i) {
156             os << " | " << setw(10) << j << " ";
157         }
158         os << " | ";
159         os << " = " << setw(4) << matrix.getRes()[num_res];
160         os << endl;
161         ++num_res;
162     }
163     return os;
164 }
165 
166 std::ostream& operator << (std::ostream &os, const vector <long double> &v)
167 {
168     os << "[" << endl;
169     for (size_t i = 0; i < v.size(); ++i){
170         os << "   v" << i + 1 << ": " << v[i] << "," << endl;
171     }
172     os << "]" << endl;
173 }
```

Листинг 2: Matrix.cpp

5 Примеры и результаты работы программы

1. Система неопределена
2. Система неопределена, но все векторы различны
3. Система несовместна
4. Система определена

input.txt:

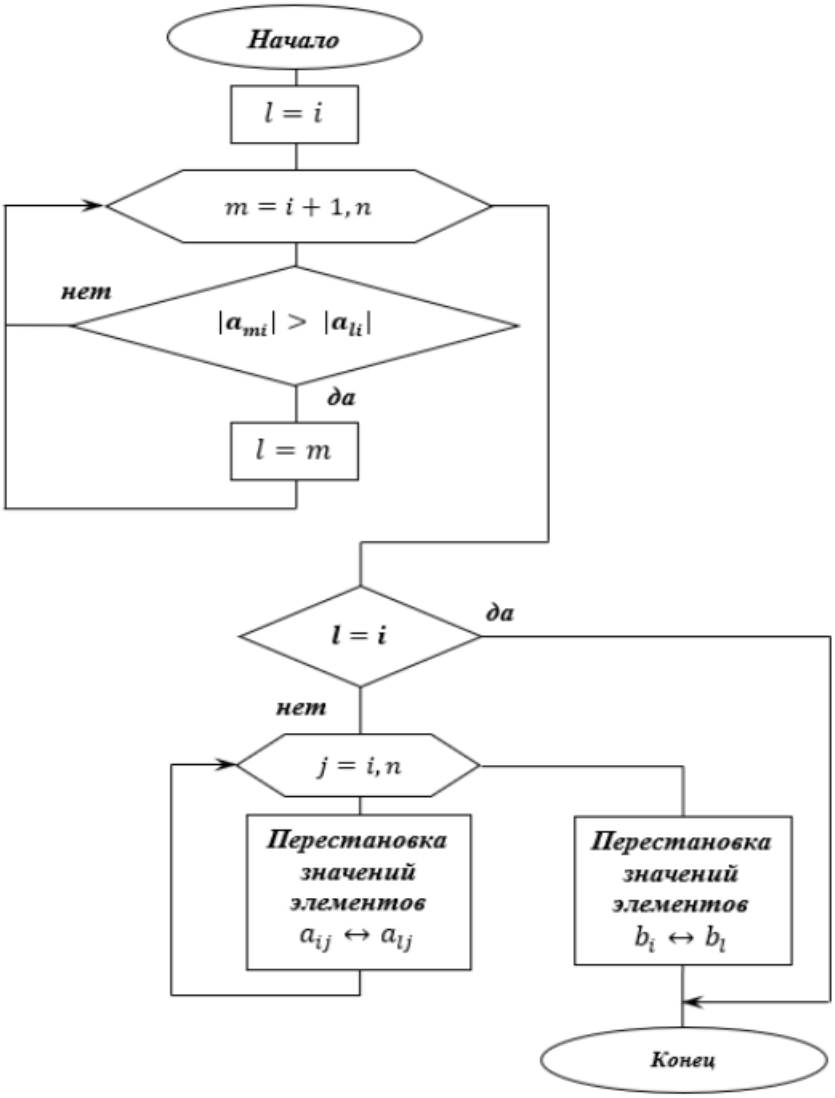
```
4

3
1 1 1 1
1 1 1 1
1 1 1 1

3
1 2 3 4
3 6 9 12
1 1 1 1

4
3 4 9 2 1
1 1 1 1 5
1 343 322 4 2
1 1 1 1 3

5
1 3 8 3 8 9
1 12 3 343 343 12
737 745 38 282 3 3
23 77 32 838 33 82
9 74 73 7333 9 23
```

output:

```
----------------TEST #1----------------

The original matrix:
 |      1.000  |      1.000  |      1.000  |  = 1.000
 |      1.000  |      1.000  |      1.000  |  = 1.000
 |      1.000  |      1.000  |      1.000  |  = 1.000

Turn column #0 to zero:
 |      1.000  |      1.000  |      1.000  |  = 1.000
 |      0.000  |      0.000  |      0.000  |  = 0.000
 |      0.000  |      0.000  |      0.000  |  = 0.000

Turn column #1 to zero:
 |      1.000  |      1.000  |      1.000  |  = 1.000
 |      0.000  |      0.000  |      0.000  |  = 0.000
 |      0.000  |      0.000  |      0.000  |  = 0.000

The determinant of a triangular matrix: 0.000
Vector X: inf
----------------TEST #2----------------

The original matrix:
 |      1.000  |      2.000  |      3.000  |  = 4.000
 |      3.000  |      6.000  |      9.000  |  = 12.000
 |      1.000  |      1.000  |      1.000  |  = 1.000

Turn column #0 to zero:
 |      3.000  |      6.000  |      9.000  |  = 12.000
 |      0.000  |      0.000  |      0.000  |  = 0.000
 |      0.000  |     -1.000  |     -2.000  |  = -3.000

Turn column #1 to zero:
 |      3.000  |      6.000  |      9.000  |  = 12.000
 |      0.000  |     -1.000  |     -2.000  |  = -3.000
 |      0.000  |      0.000  |      0.000  |  = 0.000

The determinant of a triangular matrix: -0.000
Vector X: inf
----------------TEST #3----------------

The original matrix:
 |      3.000  |      4.000  |      9.000  |      2.000  |  = 1.000
 |      1.000  |      1.000  |      1.000  |      1.000  |  = 5.000
 |      1.000  |    343.000  |    322.000  |      4.000  |  = 2.000
 |      1.000  |      1.000  |      1.000  |      1.000  |  = 3.000

Turn column #0 to zero:
 |      3.000  |      4.000  |      9.000  |      2.000  |  = 1.000
 |      0.000  |     -0.333  |     -2.000  |      0.333  |  = 4.667
 |      0.000  |    341.667  |    319.000  |      3.333  |  = 1.667
 |      0.000  |     -0.333  |     -2.000  |      0.333  |  = 2.667

Turn column #1 to zero:
 |      3.000  |      4.000  |      9.000  |      2.000  |  = 1.000
 |      0.000  |    341.667  |    319.000  |      3.333  |  = 1.667
 |      0.000  |      0.000  |     -1.689  |      0.337  |  = 4.668
 |      0.000  |      0.000  |     -1.689  |      0.337  |  = 2.668

Turn column #2 to zero:
 |      3.000  |      4.000  |      9.000  |      2.000  |  = 1.000
 |      0.000  |    341.667  |    319.000  |      3.333  |  = 1.667
```

```
 |      0.000  |      0.000  |     -1.689  |      0.337  |  = 4.668
 |      0.000  |      0.000  |      0.000  |      0.000  |  = -2.000

The determinant of a triangular matrix: -0.000
Vector X: empty
----------------TEST #4----------------

The original matrix:
 |      1.000  |      3.000  |      8.000  |      3.000  |      8.000  |  = 9.000
 |      1.000  |     12.000  |      3.000  |    343.000  |    343.000  |  = 12.000
 |    737.000  |    745.000  |     38.000  |    282.000  |      3.000  |  = 3.000
 |     23.000  |     77.000  |     32.000  |    838.000  |     33.000  |  = 82.000
 |      9.000  |     74.000  |     73.000  |   7333.000  |      9.000  |  = 23.000

Turn column #0 to zero:
 |    737.000  |    745.000  |     38.000  |    282.000  |      3.000  |  = 3.000
 |      0.000  |     10.989  |      2.948  |    342.617  |    342.996  |  = 11.996
 |      0.000  |      1.989  |      7.948  |      2.617  |      7.996  |  = 8.996
 |      0.000  |     53.750  |     30.814  |    829.199  |     32.906  |  = 81.906
 |      0.000  |     64.902  |     72.536  |   7329.556  |      8.963  |  = 22.963

Turn column #1 to zero:
 |    737.000  |    745.000  |     38.000  |    282.000  |      3.000  |  = 3.000
 |      0.000  |     64.902  |     72.536  |   7329.556  |      8.963  |  = 22.963
 |      0.000  |      0.000  |      5.725  |   -222.021  |      7.721  |  = 8.292
 |      0.000  |      0.000  |    -29.258  |  -5240.941  |     25.483  |  = 62.889
 |      0.000  |      0.000  |     -9.333  |   -898.410  |    341.478  |  = 8.108

Turn column #2 to zero:
 |    737.000  |    745.000  |     38.000  |    282.000  |      3.000  |  = 3.000
 |      0.000  |     64.902  |     72.536  |   7329.556  |      8.963  |  = 22.963
 |      0.000  |      0.000  |    -29.258  |  -5240.941  |     25.483  |  = 62.889
 |      0.000  |      0.000  |      0.000  |  -1247.584  |     12.708  |  = 20.598
 |      0.000  |      0.000  |      0.000  |    773.424  |    333.349  |  = -11.953

Turn column #3 to zero:
 |    737.000  |    745.000  |     38.000  |    282.000  |      3.000  |  = 3.000
 |      0.000  |     64.902  |     72.536  |   7329.556  |      8.963  |  = 22.963
 |      0.000  |      0.000  |    -29.258  |  -5240.941  |     25.483  |  = 62.889
 |      0.000  |      0.000  |      0.000  |  -1247.584  |     12.708  |  = 20.598
 |      0.000  |      0.000  |      0.000  |      0.000  |    341.227  |  = 0.816

The determinant of a triangular matrix: 595784423504.000
Vector X: [
   v1: -1.360,
   v2: 1.315,
   v3: 0.806,
   v4: -0.016,
   v5: 0.002,
]

Vector R: [
   v1: 0.000,
   v2: 0.000,
   v3: 0.000,
   v4: 0.000,
   v5: 0.000,
]
```

6 Блок-схема алгоритма



l – номер наибольшего по абсолютной величине элемента матрицы в столбце с номером i (т. е. среди элементов a_{ii}, ... , a_{mi}, ... , a_{ni});
m – текущий номер элемента, с которым происходит сравнение;
Выбор главного элемента осуществляется по столбцу

7 Вывод

В ходе выполнения лабораторной работы, я вспомнил метод Гаусса для нахождения решений СЛАУ, а также написал реализацию этого алгоритма на языке C++