Отчет

Лабораторная работа №1

«Метод Гаусса»

Профессор кафедры вычислительной математики ФПМИ

Лиходед Николай Александрович

Студент 2 группы 2 курса

Сачек Илья Валерьевич

**Постановка задачи**

Пусть дана система линейных алгебраических уравнений вида:

a11x1 + a12x2 +  … + a1nxn = b1 ,

a21x2 + a22x2 +  … + a2nxn = b2 ,

. . . . . . . . . . . . . . . . . . . . . . . .

an1x1 + an2x2 +  … + annxn = bn .

**Задание №1.** Разработать программу численного решения СЛАУ методом Гаусса без выбора ведущего элемента.

Начальная матрица:

19.01 -1 -3 -2 0 -1 0 -2 -2 -2 0 -1 -1 -1 -3

-1 21 -3 -3 -2 -3 0 -3 0 -2 -1 0 -2 -1 0

-2 -3 33 -3 -2 -3 -2 -1 0 -3 -3 -3 -2 -3 -3

-2 -1 -1 19 0 -1 -3 -1 0 -3 0 -2 -1 -1 -3

-3 -1 -1 -2 18 0 -3 -2 -2 0 -2 0 -2 0 0

-2 -1 -2 -1 -2 20 -2 -2 -1 -1 0 -1 -3 0 -2

0 0 -2 0 -3 0 20 0 -3 -2 -3 -1 -2 -2 -2

-1 -1 -1 -3 -3 -2 0 21 -2 -1 -2 -1 -2 -1 -1

0 0 -2 -1 -1 -1 -1 0 17 -2 -2 -1 0 -3 -3

-3 -2 -3 0 -2 -2 -1 0 -2 27 -2 -3 -3 -3 -1

-1 -3 -1 -3 -1 -1 -1 -2 -1 0 22 -2 -2 -2 -2

-1 0 0 -3 -2 -3 -2 -3 0 -1 0 22 -3 -3 -1

-2 -2 -3 0 -2 0 0 -1 0 -3 0 0 16 -1 -2

-1 -3 -2 -2 0 0 -3 -3 0 -1 0 -3 -3 22 -1

-2 -1 -1 -2 -1 0 -3 -3 0 -1 -3 0 -2 -3 22

Матрица после первого шага алгоритма:

19 -1 -3 -2 0 -1 0 -2 -2 -2 0 -1 -1 -1 -3 -1.4e+02

0 21 -3.2 -3.1 -2 -3.1 0 -3.1 -0.11 -2.1 -1 -0.053 -2.1 -1.1 -0.16 -1.1e+02

0 -2.4e-07 32 -3.7 -2.3 -3.6 -2 -1.7 -0.23 -3.5 -3.1 -3.1 -2.4 -3.3 -3.3 -2.1e+02

0 0 0 18 -0.21 -1.4 -3.1 -1.5 -0.23 -3.5 -0.2 -2.3 -1.3 -1.3 -3.5 -1.2e+02

0 0 0 0 18 -0.72 -3.6 -2.8 -2.4 -1.1 -2.2 -0.65 -2.6 -0.57 -1.2 -87

0 0 0 0 0 19 -2.9 -3 -1.6 -2.1 -0.6 -1.6 -3.9 -0.6 -3 -94

0 0 0 0 0 0 19 -0.67 -3.5 -2.5 -3.6 -1.4 -2.7 -2.3 -2.5 -89

0 0 0 0 0 2.4e-07 0 19 -3.1 -2.7 -3 -2.1 -3.8 -1.9 -2.8 -76

0 0 0 0 0 0 1.2e-07 0 16 -3 -2.8 -1.7 -1.1 -3.6 -4.1 -61

0 0 0 0 0 0 0 0 0 24 -4.3 -4.8 -5.6 -5.2 -4.2 -73

0 0 0 0 0 0 0 0 0 0 19 -4.3 -5.2 -4.6 -5.4 -50

0 0 0 2.4e-07 0 0 -2.4e-07 0 0 -2.4e-07 0 19 -7.3 -6.2 -5.4 -36

0 0 0 -6e-08 2.4e-07 0 0 0 0 0 0 0 11 -4.9 -6.3 -18

0 0 0 0 0 0 0 0 0 -2.4e-07 0 0 0 12 -12 -12

0 0 0 0 0 0 0 0 0 0 -4.8e-07 2.4e-07 -4.8e-07 0 0.01 0.17

Вектор приближенного решения:

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Относительная погрешность:

R = 3e-07

**Задание 2.** Разработать программу численного решения СЛАУ методом

Гаусса с выбором ведущего элемента по столбцу.

Для заполнения матрицы A использовать случайные числа из диапазона от −100 до 100. Правую часть b задать умножением матрицы A на вектор

x=(m, m+1, ... , n+m–1): b=Ax. Для вычислений выбрать параметры: m – номер в списке студенческой группы; n – одно из чисел в пределах от 15 до 20 (12 для сдачи в конце семестра). Программно реализовать вычисления для рассматриваемого примера методом Гаусса с выбором ведущего элемента и методом Гаусса без выбора ведущего элемента (система уравнений в обоих случаях одна и та же). Для вычислений использовать тип float.

Входные данные:

19.01 -1 -3 -2 0 -1 0 -2 -2 -2 0 -1 -1 -1 -3 -144

-1 21 -3 -3 -2 -3 0 -3 0 -2 -1 0 -2 -1 0 -103

-2 -3 33 -3 -2 -3 -2 -1 0 -3 -3 -3 -2 -3 -3 -183

-2 -1 -1 19 0 -1 -3 -1 0 -3 0 -2 -1 -1 -3 -92

-3 -1 -1 -2 18 0 -3 -2 -2 0 -2 0 -2 0 0 -29

-2 -1 -2 -1 -2 20 -2 -2 -1 -1 0 -1 -3 0 -2 -34

0 0 -2 0 -3 0 20 0 -3 -2 -3 -1 -2 -2 -2 -57

-1 -1 -1 -3 -3 -2 0 21 -2 -1 -2 -1 -2 -1 -1 6

0 0 -2 -1 -1 -1 -1 0 17 -2 -2 -1 0 -3 -3 -16

-3 -2 -3 0 -2 -2 -1 0 -2 27 -2 -3 -3 -3 -1 53

-1 -3 -1 -3 -1 -1 -1 -2 -1 0 22 -2 -2 -2 -2 69

-1 0 0 -3 -2 -3 -2 -3 0 -1 0 22 -3 -3 -1 79

-2 -2 -3 0 -2 0 0 -1 0 -3 0 0 16 -1 -2 101

-1 -3 -2 -2 0 0 -3 -3 0 -1 0 -3 -3 22 -1 142

-2 -1 -1 -2 -1 0 -3 -3 0 -1 -3 0 -2 -3 22 154

Матрица после первого шага алгоритма:

95 76 67 -61 -26 31 56 1 -28 60 -6 84 55 89 -93 -85 93 3145

0 159 -7.053 -85.58 73.74 80.74 97.11 -74.11 -115.1 15.68 -94.37 -16.84 126.2 140.6 -158.2 -41.05 51.21 377.9

0 99.2 128 -80.25 -8.042 64.36 75.94 76.46 -101 81.79 94.22 42.91 106.5 89.22 -78.07 38.63 42.07 6679

0 -38.6 -7.463 97.71 -65.72 34.08 -50.07 -84.13 93.54 -67.58 -20.24 24.39 -100.9 60.76 9.747 5.737 7.253 -822.3

0 94.6 108.8 30.77 77.29 17.49 46.44 -90.24 -90.22 90.47 70.45 -1.337 39.68 109.5 28.52 3.579 -17.52 4339

0 37.6 -37.64 -4.863 26.19 93.39 25.28 2.916 24.36 -39.05 49.51 -83.07 -103.6 -54.49 84.83 -84.84 -79.83 -3485

-3.815e-06 -40 45.32 30.47 -7.421 76.58 95.37 33.63 -18.68 15.89 -87.79 88.05 31.74 -17.79 -46.74 -86.68 -31.26 -756.7

0 132.6 96.25 35.98 -97.34 88.86 11.49 93.71 16.25 27.32 40.77 -12.76 29.79 156.8 -8.589 -158.9 118.6 4786

0 -100 -52.53 1.211 89.37 21.37 2.053 -18.05 53.47 62.84 16.32 32.58 -64.89 39.32 5.895 1.474 -33.89 1071

0 -12.2 -28.29 4.358 115.7 19.33 3.589 -77.99 74.71 25.63 -77.06 -111.1 -15.42 -141.1 177 80.11 -107 -1376

0 13.8 -6.032 -2.747 -41.96 60.64 8.063 23.54 -6.032 -27.79 61.78 37.09 -28.47 134.8 -46.93 -18.63 78.93 3286

0 89 122.2 -117.7 84.05 81.05 36.58 30.42 -105.8 87.26 39.47 100.4 5.158 -10.53 27.84 -114.8 34.16 2509

0 108.6 99.41 -148.3 -107.5 106.7 -33.82 81.02 0.4105 -15.74 -52.13 45.77 93.16 135.9 -77.96 -61.79 -5.042 1296

0 -40.2 21.44 30.46 -7.589 41.01 50.12 6.484 -100.6 99.05 -64.91 34.67 -53.37 -12.91 20.97 -7.158 -28.97 -674.2

0 -75.2 -93.66 -0.6947 24.88 40.48 -38.67 -24.73 28.34 -94.58 35.36 -117 -101.9 -156.6 164.5 12.74 -118.5 -5618

0 -18.4 98.31 18.56 13.93 40.13 -29.61 34.81 -80.69 -40.37 -25.86 120.1 -11.42 67.14 -157.4 27.11 67.38 874.1

0 -28.4 57.73 -43.81 -77.97 -45.77 -105.5 -37.35 45.73 -95.84 -28.92 -14.18 -25.11 -101.9 33.31 46.53 62.69 -2597

Вектор приближенного решения:

1.995 2.995 3.995 4.995 5.995 6.995 7.995 8.995 9.995 10.99 11.99 12.99 13.99 14.99 15.99

Относительная погрешность:

R = 3.016e-07

Выходные данные:

1 -0.0526 -0.1578 -0.1052 0 -0.0526 0 -0.1052 -0.1052 -0.1052 0 -0.0526 -0.0526 -0.0526 -0.1578 -7.574

0 1 -0.1507 -0.1482 -0.09548 -0.1457 0 -0.1482 -0.005022 -0.1005 -0.04774 -0.002511 -0.09799 -0.05025 -0.007534 -5.279

0 -2.384e-07 1 -0.1139 -0.07128 -0.1104 -0.06208 -0.05186 -0.007015 -0.1093 -0.09772 -0.09663 -0.07479 -0.1012 -0.1036 -6.659

0 0 0 1 -0.01144 -0.07748 -0.1675 -0.07862 -0.01226 -0.1887 -0.01071 -0.122 -0.07175 -0.07102 -0.1884 -6.656

0 0 0 0 1 -0.04034 -0.2001 -0.1569 -0.1333 -0.06296 -0.1265 -0.03643 -0.1458 -0.03229 -0.06521 -4.882

0 0 0 0 0 1 -0.1503 -0.1556 -0.08112 -0.1066 -0.03141 -0.08492 -0.2003 -0.0314 -0.1582 -4.874

0 0 0 0 0 0 1 -0.03508 -0.181 -0.1305 -0.1883 -0.07135 -0.1408 -0.1217 -0.1312 -4.645

0 0 0 0 0 2.384e-07 0 1 -0.161 -0.1401 -0.1563 -0.1067 -0.1938 -0.09688 -0.145 -3.902

0 0 0 0 0 0 1.192e-07 0 1 -0.1837 -0.174 -0.1042 -0.06481 -0.2241 -0.2492 -3.719

0 0 0 0 0 0 0 0 0 1 -0.1773 -0.1976 -0.234 -0.2155 -0.1755 -3.013

0 0 0 0 0 0 0 0 0 0 1 -0.2195 -0.2676 -0.2342 -0.2786 -2.57

0 0 0 2.384e-07 0 0 -2.384e-07 0 0 -2.384e-07 0 1 -0.3866 -0.3267 -0.2866 -1.898

0 0 0 -5.96e-08 2.384e-07 0 0 0 0 0 0 0 1 -0.4375 -0.5622 -1.5580

0 0 0 0 0 0 0 0 -2.384e-07 0 0 0 1 -0.9996 -0.993

0 0 0 0 0 0 0 0 0 0 -4.768e-07 2.384e-07 -4.768e-07 0 1 15.99

**Листинг программы:**

#include <iostream>

#include <vector>

#include <cmath>

#include <iomanip>

#include <fstream>

float CountInfelicity(std::vector<float> x\_de\_ure, std::vector<float> x\_de\_facto) {

float sum\_x2 = 0, sum\_x = 0;

for (int i = 0; i < x\_de\_facto.size(); i++) {

sum\_x2 += x\_de\_facto[i] \* x\_de\_facto[i];

sum\_x += (x\_de\_facto[i] - x\_de\_ure[i]) \* (x\_de\_facto[i] - x\_de\_ure[i]);

}

std::ofstream fout("errors.txt");

fout << sum\_x / sum\_x2 << std::endl;

fout.close();

return sum\_x / sum\_x2;

}

class Matrix {

public:

Matrix() { // создание матрицы для 1 случая

line\_length = (15 + rand() % 5);

matrix\_.clear();

matrix\_.resize(line\_length);

for (auto& vec : matrix\_) {

vec.resize(line\_length);

}

for (auto& vec : matrix\_) {

for (auto& x : vec) {

x = rand() % 200 - 100;

}

}

}

Matrix(const int& n1) { // создание матрицы для 2 случая

line\_length = n1;

matrix\_.resize(line\_length);

for (int i = 0; i < line\_length; ++i) {

matrix\_[i].resize(line\_length);

}

for (int i = 0; i < line\_length; ++i) {

for (int j = 0; j < line\_length; ++j) {

if (i != j) {

matrix\_[i][j] = (-1) \* (rand() % 4);

}

else {

matrix\_[i][j] = 0;

if (i == 0) {

matrix\_[i][j] = pow(10, -2);

}

}

}

}

for (int i = 0; i < line\_length; ++i) {

for (int j = 0; j < line\_length; ++j) {

if (i != j) {

matrix\_[i][i] += -matrix\_[i][j];

}

}

}

PrintMatr();

std::cout << std::endl << std::endl;

}

~Matrix() {

matrix\_.clear();

}

std::vector<float> GaussWithoutMainElement() {

// заполнение матрицы

std::vector<float> vec(line\_length);

int m = 2;

for (int i = 0; i < line\_length; ++i) {

vec[i] = m;

++m;

}

std::vector<float> vec1(line\_length);

for (int i = 0; i < line\_length; ++i) {

for (int j = 0; j < line\_length; ++j) {

vec1[i] += matrix\_[i][j] \* vec[j];

}

}

// заполняем матрицу гаусса

std::vector<std::vector<float>> matrix\_transform(line\_length);

for (int i = 0; i < line\_length; ++i) {

matrix\_transform[i].resize(line\_length + 1);

}

for (int i = 0; i < line\_length; ++i) {

for (int j = 0; j < line\_length + 1; ++j) {

if (j != line\_length) {

matrix\_transform[i][j] = matrix\_[i][j];

}

else {

matrix\_transform[i][j] = vec1[i];

}

}

}

// прямой ход гаусса

for (int i = 0; i < line\_length; ++i) {

for (int s = i + 1; s < line\_length; ++s) {

float h = matrix\_transform[s][i] / matrix\_transform[i][i];

for (int j = i; j < line\_length + 1; ++j) {

matrix\_transform[s][j] -= matrix\_transform[i][j] \* h;

}

}

}

// выводим на экран

for (int i = 0; i < line\_length; ++i) {

for (int j = 0; j < line\_length; ++j) {

std::cout << std::setw(7) << std::setprecision(2) << std::left << matrix\_transform[i][j] << " ";

}

std::cout << " " << matrix\_transform[i][line\_length] << std::endl;

}

// вычисляем ответ

std::vector<float> vals(line\_length);

float x = 0;

for (int i = line\_length - 1; i >= 0; --i) {

if (i > line\_length) {

break;

}

x = matrix\_transform[i][line\_length];

for (int j = line\_length - 1; j > i; --j) {

if (i == line\_length - 1 && j == line\_length - 1) {

vals[i] = matrix\_transform[i][j];

continue;

}

else {

x -= matrix\_transform[i][j] \* vals[j];

}

}

vals[i] = x / matrix\_transform[i][i];

}

std::cout << "Error :" << CountInfelicity(vec, vals) << std::endl;

return vals;

}

std::vector<float> GaussWithMainElement() {

std::vector<float> vals(line\_length), vec(line\_length), vec1(line\_length);

int m = 2;

for (int i = 0; i < line\_length; ++i) {

vec[i] = m;

++m;

}

for (int i = 0; i < line\_length; ++i) {

for (int j = 0; j < line\_length; ++j) {

vec1[i] += matrix\_[i][j] \* vec[j];

}

}

std::vector<std::vector<float>> matrix\_transform(line\_length);

for (int i = 0; i < line\_length; ++i) {

matrix\_transform[i].resize(line\_length + 1);

}

for (int i = 0; i < line\_length; ++i) {

for (int j = 0; j < line\_length + 1; ++j) {

if (j != line\_length) {

matrix\_transform[i][j] = matrix\_[i][j];

} else {

matrix\_transform[i][j] = vec1[i];

}

}

}

for (int i = 0; i < line\_length; ++i) {

for (int j = 0; j < line\_length; ++j) {

std::cout << std::setw(6) << std::setprecision(4) << std::left << matrix\_transform[i][j] << " ";

}

std::cout << " " << matrix\_transform[i][line\_length] << std::endl;

}

int max\_index = 0;

float max = 0;

for (int k = 0; k < line\_length; ++k) {

max\_index = k;

max = 0;

for (int i = k; i < line\_length; ++i) {

if (matrix\_transform[i][k] > max) {

max = matrix\_transform[i][k];

max\_index = i;

}

}

swap(matrix\_transform[k], matrix\_transform[max\_index]);

}

for (int i = 0; i < line\_length; ++i) {

for (int s = i + 1; s < line\_length; ++s) {

float h = matrix\_transform[s][i] / matrix\_transform[i][i];

for (int j = i; j < line\_length + 1; ++j) {

matrix\_transform[s][j] -= matrix\_transform[i][j] \* h;

}

}

}

for (int i = 0; i < line\_length; ++i) {

float h = matrix\_transform[i][i];

for (int j = i; j < line\_length + 1; ++j) {

matrix\_transform[i][j] /= h;

}

}

std::cout << std::endl << std::endl;

for (int i = 0; i < line\_length; ++i) {

for (int j = 0; j < line\_length; ++j) {

std::cout << std::setw(6) << std::setprecision(4) << std::left << matrix\_transform[i][j] << " ";

}

std::cout << " " << matrix\_transform[i][line\_length] << std::endl;

}

float x = 0;

for (int i = line\_length - 1; i >= 0; --i) {

if (i > line\_length) {

break;

}

x = matrix\_transform[i][line\_length];

for (int j = line\_length - 1; j > i; --j) {

if (i == line\_length - 1 && j == line\_length - 1) {

vals[i] = matrix\_transform[i][j];

continue;

}

else {

x -= matrix\_transform[i][j] \* vals[j];

}

}

vals[i] = x;

}

std::cout << "Error :" << CountInfelicity(vec, vals) << std::endl;

return vals;

}

void PrintMatr() {

for (const auto& vec : matrix\_) {

for (const auto& x : vec) {

std::cout << std::left << std::setw(7) << x << " ";

}

std::cout << std::endl;

}

}

private:

int line\_length;

std::vector<std::vector<float>> matrix\_;

};

int main() {

Matrix m(15);

std::vector<float> vec = m.GaussWithoutMainElement();

std::cout << std::endl << std::endl;

for (const auto& x : vec) {

std::cout << x << " ";

}

std::cout << std::endl << std::endl;

//Matrix x;

std::vector<float> vec2 = m.GaussWithMainElement();

for (const auto& x : vec2) {

std::cout << x << " ";

}

return 0;

}

**Вывод:**

Данный алгоритм решает СЛАУ. Кроме того, он позволяет выполнить это с достаточно малой погрешностью, т.е. наше решение отличается от точного совсем на маленькие значения.