МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

**Компьютерный практикум по курсу**

**«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»**

**ЗАДАНИЕ № 1 (1)**

**«РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ МЕТОДОМ ГАУССА И МЕТОДОМ ГАУССА С ВЫБОРОМ ГЛАВНОГО ЭЛЕМЕНТА»**

**ОТЧЕТ**

**о выполненном задании**

студента 203 учебной группы факультета ВМК МГУ

Дорожкина Дениса Сергеевича

Москва, 2015 г.

**Вариант задания**

Подвариант 1: приложение 1-11, приложение 2 (п. 1-5)

**Цель работы**

Изучить метод Гаусса для решения системы линейных алгебраических уравнений.

**Постановка задачи**

Дана система Ax= b, где A – невырожденная матрица размера n \* n, а b и x – вектора порядка n.

x = (x1, x2, …, xn). Написать программу, которая решает систему линейных алгебраических уравнений для произвольного размера матрицы n методом Гаусса и методом Гаусса с выбором главного элемента.

**Метод решения**

Программа поддерживает 2 метода ввода матрицы A и вектора b:

1. Из файла

Файл имеет следующую структуру:

В первой строке находится единственное число n – порядок задаваемой матрицы.

Дальше идет n \* n элементов матрицы A.

После этого поступает n элементов – элементов вектора b.

1. Задание матрицы через рекуррентную формулу

Рекуррентная формула соответствует (п. 1-5) задания

Далее находится решение системы линейных алгебраических уравнений сначала методом Гаусса, а потом методом Гаусса с выделением главного элемента. Параллельно находится ее определитель, число обусловленности и обратная матрица.

**Алгоритм решения**

* На вход в программу сначала подается число (1 или 2), которое выбирает метод ввода матрицы A и вектора b в программе. (1 - Из файла, 2 - Через рекуррентную формулу).
* В зависимости от метода получения матрицы, заполняется матрица matrix и вектор vector.
* Запускается метод Гаусса по нахождению решений системы линейных алгебраических уравнений.
* Метод Гаусса заключается в сведении расширенной матрицы A|b к треугольной путем линейных преобразований строк и получения вектора решения путем решения линейных уравнений с одной переменной. Разделяют прямой и обратный ход метода Гаусса.

1)Прямой: Итерируясь по всем строкам на i-ом шаге линейными преобразованиями обнуляем все элементы в i-ом столбце ниже i-ой строки. Если на i-ом шаге элемент на позиции i, i нулевой, то мы меняем i-ую строку местами с первой строкой, в которой i-ый элемент не равен нулю. Мы обязательно найдем такую строку, так как по условию - матрица невырожденная.

2)Обратный: Получаем решение системы обратным ходом метода Гаусса. Проходим по треугольной матрице A с ее конца и на i-ом шаге получаем (n - i + 1)-ое решение системы уравнений из линейного уравнения. Потом составляем вектор решений.

* Подсчет определителя треугольной матрицы А сводится к перемножению диагональных элементов матрицы, также важно помнить и о знаке, который меняется на противоположный, каждый раз, когда меняются строки матрицы.
* Обратная матрица находится путем сведения левой части расширенной матрицы A|I к единичной, путем преобразования строк.
* Число обусловленности находится, как произведение норм обратной матрицы и исходной. Норму матрицы можно определить как корень из суммы квадратов ее элементов.

**Исходный код**

#include **<iostream>**#include **<fstream>**#include **<stdio.h>**#include **<stdlib.h>**#include **<math.h>**#include **<limits.h>**#include **<algorithm>  
  
using namespace** std;  
  
**void** print\_matrix(**double** \*\*matrix, **int** n) *// вывод матрицы*{  
 **for** (**int** i = 0; i < n; i++) {  
 **for** (**int** k = 0; k < n; k++) {  
 cout << matrix[i][k] << **" "**;  
 }  
 cout << **"\n"**;  
 }  
   
}  
  
**void** print\_vector(**double** \*vector, **int** n) *// вывод вектора*{  
 **for** (**int** i = 0; i < n; i++) {  
 cout << vector[i] << **" "**;  
 }  
 cout << **"\n"**;  
}  
  
**double** \*\*create\_matrix(**int** n) *//выделение памяти для матрицы*{  
 **double** \*\*newmatrix = (**double** \*\*)calloc(n, **sizeof**(\*newmatrix));  
 **for** (**int** i = 0; i < n; i++) {  
 newmatrix[i] = (**double** \*)calloc(n, **sizeof**(\*\*newmatrix));  
 }  
 **return** newmatrix;  
}  
  
**double** \*\*duplicate\_matrix(**double** \*\*matrix, **int** n) *// копирование матрицы*{  
 **double** \*\*newmatrix = create\_matrix(n);  
 **for** (**int** i = 0; i < n; i++) {   
 **for** (**int** k = 0; k < n; k++) {  
 newmatrix[i][k] = matrix[i][k];  
 }  
   
 }  
 **return** newmatrix;  
}  
  
**double** \*duplicate\_vector(**double** \*vector, **int** n) *// копирование вектора*{  
 **double** \*newvector = (**double** \*)calloc(n, **sizeof**(\*vector));  
 **for** (**int** i = 0; i < n; i++) {   
 newvector[i] = vector[i];  
 }  
 **return** newvector;  
}  
  
**void** free\_matrix(**double** \*\*matrix, **int** n) *// освобождение памяти*{  
 **for**(**int** i = 0; i < n; i++) {  
 free(matrix[i]);  
 }  
 free(matrix);  
}  
  
**void** gauss(**double** \*\*matrix, **double** \*vector, **int** n)  
{  
 *// метод Гаусса* **int** flag; *// флаг, который хранит, нужно ли переставлять строки в матрице* **double** dev; *// сохраняемый элемент матрицы, для упрощения вычислений* **double** \*\*invert = create\_matrix(n); *// обратная матрица, которая параллельно вычисляется.* **double** det = 1; *// детерминант, который параллельно вычисляется* **double** normmatr = 0; *// норма исходной матрицы* **double** norminv = 0; *// норма обратной матрицы* **for**(**int** i = 0; i < n; i++)  
 **for** (**int** k = 0; k < n; k++)   
 normmatr += matrix[i][k] \* matrix[i][k];  
 normmatr = sqrt(normmatr);  
 **for** (**int** i = 0; i < n; i++) invert[i][i] = 1; *// сначала обратная матрица - единичная* cout << **"Исходная матрица\n"**;  
 print\_matrix(matrix, n); *//вывод исходной матрицы* **for** (**int** i = 0; i < n - 1; i++) { *// проход прямым ходом метода Гаусса* flag = 0;  
 **for** (**int** k = i; k < n; k++) { *// поиск первого ненулевого элемента в i-ом столбце* **if** (fabs(matrix[k][i]) < 0.000001) { *// считаем ноль с опредленной точностью* flag = 1;  
 **if** (k == n - 1) { *// обработка ошибки при вырожденности матрицы* cout << **"матрица вырождена!\n"**;  
 exit(0);  
 }  
 } **else** { *// если, найденный первый ненулевой элемент не i-ый,   
 //то нужно поменять его строку местами с i-ым  
 // это учитывается и в обратной матрице* **if** (flag) {  
 swap(matrix[k], matrix[i]);  
 swap(invert[k], invert[i]);  
 swap(vector[k], vector[i]);  
 det \*= -1; *// перестановка строк отражается на знаке определителя* }  
 **break**;  
 }  
 }  
 *// Преобразовывание матрицы: обнуление всех элементов i-го столбца после i-той строки  
 // с помощью линейных преобразований строк расширенной матрицы  
 // также преобразуется и обратная матрица* **for** (**int** k = i + 1; k < n; k++) {  
 dev = 1.0 / matrix[i][i] \* matrix[k][i];  
 **for** (**int** j = i; j < n; j++) {  
 matrix[k][j] -= matrix[i][j] \* dev;   
 invert[k][j] -= invert[i][j] \* dev;   
 }  
 vector[k] -= vector[i] \* dev;  
 }  
 }  
 *// проход обратным ходом метода Гаусса  
 // вычисление значения вектора ответа по полученной треугольной матрице* **for** (**int** i = n - 1; i >= 0; i--) {  
 **for** (**int** k = n - 1; k > i; k--) {  
 vector[i] -= matrix[i][k] \* vector[k];  
 }  
 vector[i] = vector[i] / matrix[i][i];  
 }  
 cout << **"Метода Гаусса - ответ:\n"**;  
 print\_vector(vector, n);  
   
   
 *// Вычисление определителя, Обратной матрицы и Числа обусловленности* cout << **"Определитель:\n"**;  
 **for**(**int** i = 0; i < n; i++) { *// Оставшаяся после метода Гаусса   
 // треугольная матрица упрощает вычисление определителя   
 // ее определить - произведение элементов на диагонали* det \*= matrix[i][i];  
 }  
 cout << det << **"\n"**;  
 cout << **"Обратная:\n"**;  
 *// Преобразовывание оставшейся после метода Гаусса треугольной матрицы   
 // в единичную линейными преобразованиями. Параллельное преобразование и обратной.* **for** (**int** i = n - 1; i > 0; i--) {  
 dev = matrix[i][i];  
 **for** (**int** k = n - 1; k >= 0; k--) {  
 matrix[i][k] /= dev;  
 invert[i][k] /= dev;  
 }  
 dev = matrix[i - 1][i];  
 **for** (**int** k = n - 1; k >= 0; k--) {  
 invert[i - 1][k] -= invert[i][k] \* dev;  
 matrix[i - 1][k] -= matrix[i][k] \* dev;  
 }  
 }  
 dev = matrix[0][0];  
 **for** (**int** k = n - 1; k >= 0; k--) {  
 matrix[0][k] /= dev;  
 invert[0][k] /= dev;  
 }  
 print\_matrix(invert, n);  
 cout << **"Число обусловленности:\n"**;  
 *// Вычисление нормы обратной матрицы.* **for**(**int** i = 0; i < n; i++)  
 **for** (**int** k = 0; k < n; k++)   
 norminv += invert[i][k] \* invert[i][k];  
 norminv = sqrt(norminv);  
 *// Число обусловленности есть произведение нормы исходной матрицы и нормы обратной к ней* cout << norminv \* normmatr << **"\n"**;  
}  
  
**void** gauss\_full\_pivot(**double** \*\*matrix, **double** \*vector, **int** n)   
{  
 *// метод Гаусса с выделением главного элемента* **double** max\_elem = 0; *// главный элемент* **int** max\_index = 0; *// индекс главного элемента* **double** dev; *// сохраняемый элемент матрицы, для упрощения вычислений* **double** \*swap\_vector = (**double** \*)calloc(n, **sizeof**(\*swap\_vector));*// вектор перестановок столбцов в матрице* **for**(**int** i = 0; i < n; i++) swap\_vector[i] = i;  
 **for** (**int** i = 0; i < n - 1; i++) { *// проход прямым ходом метода Гаусса* **for** (**int** k = i; k < n; k++) { *// поиск индекса максимального по модулю элемента в i строке* **if** (fabs(max\_elem) < fabs(matrix[i][k])) {  
 max\_elem = matrix[i][k];  
 max\_index = k;  
 }  
 }  
 *// меняются местами столбцы - выбирается главный элемент  
 // учитывается это и в векторе перестановок* **for** (**int** k = 0; k < n; k++) {  
 swap(matrix[k][max\_index], matrix[k][i]);  
 }  
 swap(swap\_vector[i], swap\_vector[max\_index]);  
 *// Преобразовывание матрицы: обнуление всех элементов i-го столбца после i-той строки  
 // с помощью линейных преобразований строк расширенной матрицы* **for** (**int** k = i + 1; k < n; k++) {  
 dev = 1.0 / matrix[i][i] \* matrix[k][i];  
 **for** (**int** j = i; j < n; j++) {  
 matrix[k][j] -= matrix[i][j] \* dev;   
 }  
 vector[k] -= vector[i] \* dev;  
 }  
 }  
 *// проход обратным ходом метода Гаусса  
 // вычисление значения вектора ответа по полученной треугольной матрице* **for** (**int** i = n - 1; i >= 0; i--) {  
 **for** (**int** k = n - 1; k > i; k--) {  
 vector[i] -= matrix[i][k] \* vector[k];  
 }  
 vector[i] = vector[i] / matrix[i][i];  
 }  
 *// преобразовывание ответа с помощью накопленного вектора перестановок  
 // сортировка вектора перестановок и вектора ответов* **for**(**int** i = 0; i < n - 1; i++) {  
 **for**(**int** k = i; k < n; k++) {  
 **if** (swap\_vector[i] > swap\_vector[k]) {  
 swap(swap\_vector[i], swap\_vector[k]);  
 swap(vector[i], vector[k]);  
 }  
 }  
 }  
   
 cout << **"Метода Гаусса c выбором главного элемента - ответ:\n"**;  
 print\_vector(vector, n);  
}  
  
**int** main(**void**)  
{  
 **int** mod; *// Метод ввода матрицы* **double** n; *// Порядок матрицы* **double** elem; *// Элемент матрицы* **double** \*\*matrix; *// Исходная матрица* **double** \*vector; *// Исходный вектор* cout << **"Пожалуйста выберите метод ввода матрицы:(1) - из файла, (2) - рекурентной формулой\n"**;  
 cin >> mod;  
 **if** (mod == 1) {  
 *// Ввод матрицы и вектора из файла* **char** filename[PATH\_MAX] = {};  
 cout << **"Введите имя файла, в котором содержится матрица\n"**;  
 cin >> filename;  
 ifstream in;  
 in.open(filename);  
 in >> n;  
 matrix = create\_matrix(n);  
 **for** (**int** i = 0; i < n; i++) {  
 **for** (**int** k = 0; k < n; k++) {  
 in >> elem;  
 matrix[i][k] = elem;  
 }  
 }  
 vector = (**double** \*)calloc(n, **sizeof**(\*vector));  
 **for** (**int** i = 0; i < n; i++) {  
 in >> elem;  
 vector[i] = elem;  
 }  
 in.close();  
 } **else if** (mod == 2) {  
 *// Генерерация данных по формуле  
 // Пример 1 Варинант 5* n = 30;  
 **double** m = 20;   
 matrix = create\_matrix(n);  
 vector = (**double** \*)calloc(n, **sizeof**(\*vector));  
 **for** (**int** i = 0; i < n; i++) {  
 **for** (**int** k = 0; k < n; k++) {  
 **if** (i == k) {  
 matrix[i][k] = n + m \* m + (k + 1) / m + (i + 1)/ **double**(n);  
 } **else** {  
 matrix[i][k] = (i + k + 2) / (m + n);  
 }  
 }  
 vector[i] = m \* (i + 1) + n;  
 }  
 } **else** {  
 *// Ввод неправильного режима ввода* cout << **"Введите верный режим ввода марицы (1 или 2)\n"**;  
 exit(0);  
 }  
 *// Копирование матрицы и вектора* **double** \*\*work\_matrix = duplicate\_matrix(matrix, n);  
 **double** \*work\_vector = duplicate\_vector(vector, n);  
   
 *// метод Гаусса, определитель, число обусловленности, обратная матрица* gauss(work\_matrix, work\_vector, n);  
 *// освобождение выделенной памяти* free\_matrix(work\_matrix, n);  
 free(work\_vector);  
   
 *// метод Гаусса с выбором главного элемента* gauss\_full\_pivot(matrix, vector, n);  
 *// освобождение выделенной памяти* free\_matrix(matrix, n);  
 free(vector);  
 **return** 0;  
}

**Тестирование**

Сверка ответов осуществлялась при помощи сервиса:  
http://www.wolframalpha.com/

В ходе тестирования программе были поданы на ввод тесты из приложения 1-11 задания.

**Wolframalpha:**

x1 = 2, x2 = 1, x3 = -3, x4 = 1;

Определитель:

135

Обратная матрица:

-0.8125 2.19 -1.29 0.4

-1.41666666 2.92 -1.72 0.53333333333

0.898148 -2.0666666 0.95555555 -0.296296296

-0.0185185185185 0.133333333 -0.244444444 0.037037037

**Программа:**

Исходная матрица

4 -3 1 5

1 -2 -2 -3

3 -1 2 0

2 3 2 -8

Метода Гаусса - ответ:

2 1 -3 1

Определитель:

135

Обратная:

-0.8125 2.19 -1.29 0.4

-1.41667 2.92 -1.72 0.533333

0.898148 -2.06667 0.955556 -0.296296

-0.0185185 0.133333 -0.244444 0.037037

Метода Гаусса c выбором главного элемента - ответ:

2 1 -3 1

**Тест пройден**

**Wolframalfa:**

Показал, что матрица вырождена.

**Программа:**

Показала, что матрица вырождена.

**Тест пройден**

**Wolframalfa:**

Показал, что матрица вырождена.

**Программа:**

Показала, что матрица вырождена.

**Тест пройден**

Также были проведены тесты на таких исходных данных придуманных самостоятельно.

**Wolframalpha:**

x1 = 0, x2 = -6/7, x3 = 0, x4 = 1;

Определитель:

3754800

Обратная:

-0.0397221 0.00510751 -0.0033801 -0.00278413

0.678609 -0.078072 0.0516672 0.0425575

-0.0377134 0.00722347 -0.00283371 -0.00233408

-0.0175626 -0.1044 0.53132 0.0626398

**Программа:**

Исходная матрица

214 14 30 40

36 21 345 61

484 28 60 82

60 35 75 99

Метода Гаусса - ответ:

1.51364e-17 -0.857143 1.04506e-17 1

Определитель:

3.7548e+06

Обратная:

-0.0397221 0.00510751 -0.0033801 -0.00278413

0.678609 -0.078072 0.0516672 0.0425575

-0.0377134 0.00722347 -0.00283371 -0.00233408

-0.0175626 -0.1044 0.53132 0.0626398

Метода Гаусса c выбором главного элемента - ответ:

-8.30073e-18 -0.857143 0 1

**Тест пройден**

**Wolframalpha:**

x1 = -3.01436, x2 = -3.66256, x3 = 43.0429, x4 = -2.1483;

Определитель:

-2.5

Обратная:

0.314827 -0.0904815 -0.0043846 0.126309

0.0480702 0.560985 0.0271845 -0.783117

-1.32916 -1.21533 -0.299247 8.62055

0.0340003 0.0303334 0.335014 -0.421603

**Программа:**

Исходная матрица

3.1 0.5 0.3 0.25

0.5 2.3 0.25 0.2

0.3 0.25 0.2 3.16

0.25 0.2 0.16 0.14

Метода Гаусса - ответ:

-3.01436 -3.66256 43.0429 -2.1483

Определитель:

-2.49109

Обратная:

0.314827 -0.0904815 -0.0043846 0.126309

0.0480702 0.560985 0.0271845 -0.783117

-1.32916 -1.21533 -0.299247 8.62055

0.0340003 0.0303334 0.335014 -0.421603

Метода Гаусса c выбором главного элемента - ответ:

-3.01436 -3.66256 43.0429 -2.1483

**Тест пройден**

**Wolframalpha:**

x1 = 0.58, x2 = -130.921, x3 = -77.2, x4 = -17.54;

Определитель:

0.7

Обратная:

-662.441 -157.307 -52.9571 65.7952

-1107.4 -262.178 -88.2619 109.659

-205.805 -49.161 -16.5 20.5

-44.7143 -10.6463 -3.64286 4.47143

**Программа:**

Исходная матрица

0.5 -0.3 1.2 -3.1

4.1 -1.2 3.1 -4.7

2 1 -2 1

5 0 0 0

Метода Гаусса - ответ:

0.58 -130.92 -77.2 -17.54

Определитель:

0.7

Обратная:

-662.441 -157.307 -52.9571 65.7952

-1107.4 -262.178 -88.2619 109.659

-205.805 -49.161 -16.5 20.5

-44.7143 -10.6463 -3.64286 4.47143

Метода Гаусса c выбором главного элемента - ответ:

0.58 -130.92 -77.2 -17.54

**Тест пройден**

Также была протестирована матрица, сгенерированная рекуррентной формулой.

В ходе тестирования было получено, что ответы программы, для Метода Гаусса с выбором главного элемента совпадают (с точностью их вывода на экран), с ответами, которые показывали сервисы тестирования. Ответы программы для метода Гаусса в некоторых примерах были с погрешностью.

**Выводы:**

Показано, что для большинства матриц, даже обычный метода Гаусса дает ответ с минимальной погрешностью. Но для некоторых матриц, например, [матриц Гильберта](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_%D0%93%D0%B8%D0%BB%D1%8C%D0%B1%D0%B5%D1%80%D1%82%D0%B0), метод приводит к погрешностям. При тестировании было выявлено, что метод Гаусса дает больше погрешности, чем метод Гаусса с выбором главного элемента. Метод Гаусс с выделением главного элемента является более точным.

**Литература:**

Костомаров Д.П., Фаворский А.П.. Вводные лекции по численным методам.

<http://statistica.ru/branches-maths/chislennye-metody-sistemy-lineynykh-uravneniy/>

https://ru.wikipedia.org/wiki/Метод\_Гаусса

Костомаров Д.П. Введение в численные методы. Методическое пособие для 2 курса.