Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий
Кафедра прикладной математики

Отчет защищен с оценкой _____

Преподаватель _____
(подпись)
«___» _____ 2022 г.

Отчет
по лабораторной работе № 1

"Решить систему линейных уравнений методом Гаусса"

по дисциплине «Вычислительные алгоритмы»

Студент группы ПИ-92 Шинтяпин И. И.

Преподаватель доцент, к.ф.-м.н. Проскурин А. В.

Барнаул 2022

Задание

- Составить программу для решения системы линейных алгебраических уравнений методом Гаусса с выбором главного элемента, нахождения определителя матрицы системы и вычисления обратной матриц. Исходные данные – матрица системы уравнений и столбец свободных членов должны читаться из файла, а результаты расчетов помещаться в файл. В случае, когда матрица системы вырождена, выдать об этом сообщение. В противном случае вывести решение системы, невязки, величину определителя, обратную матрицу. Подобрать тестовые примеры, предусматриваемые различные ситуации (матрица вырожденная, невырожденная) и провести вычисления.

- Решить систему

$$\begin{pmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{pmatrix} X = b,$$

  Выбирая b равным

$$\begin{pmatrix} 23 \\ 32 \\ 33 \\ 31 \end{pmatrix}, \quad \begin{pmatrix} 23.01 \\ 31.99 \\ 32.99 \\ 31.01 \end{pmatrix}, \quad \begin{pmatrix} 23.1 \\ 31.9 \\ 32.9 \\ 31.1 \end{pmatrix}$$

  Пояснить полученные результаты.

- Вычислить для матриц **A**, выбранных Вами систем, там, где это возможно, $\|A\| * \|A^{-1}\|$, в некоторой выбранной Вами норме матрицы.

Краткое описание алгоритма

В рамках реализации данного задания был создан двумерный массив, для хранения расширенной матрицы. Максимальная размерность массива 100*101 элемент. Чтение расширенной матрицы происходит из файла input.dat построчно. Вся логика программы реализуется в функциях:

- `void read(double matrix[max_count][max_count], double *n, ifstream *fin)` - функция чтения матрицы из файла

- `void print(double matrix[max_count][max_count], double *n, FILE *file)` - функция вывода расширенной матрицы

- `void print(double matrix[max_count][max_count], int *n, FILE *file)` - вывод матрицы в консоль и в файл

- `int search_decision(double matrix[max_count][max_count], double *n, FILE *file)` - решение системы линейных уравнений, вычисление определителя

- `void norm_matrix(double matrix[max_count][max_count], double inversia_matrix[max_count][max_count], int *n, FILE *file)` – функция нахождения произведение норм m матриц: ||A||*||A^-1||

Изначально происходит заполнение расширенной матрицы данными из файла. Далее копирование содержимого матрицы в двумерный массив temp_matrix для дальнейшего поиска нормы матрицы. Затем в глобальном цикле по столбцам матрицы происходит поиск главного элемента каждого столбца, перестановка строки с этим элементом на главную диагональ и сразу же зануление элементов ниже главной диагонали, в цикле: пусть самый большой элемент - первый в текущем столбце, запоминание его номера и значения, если он равен нулю, установка соответствующего флага. Если в текущем столбце найден элемент больше первого, и он не нулевой - запоминание его значения и номера. Затем, если самый большой элемент найден, и он не нулевой – перестановка строк местами, если остались только нулевые элементы, в том числе на главной диагонали – вывод сообщения о том что матрица вырождена. В этом же цикле, происходит поиск коэффициента умножения каждого элемента строки, чтобы занулить нижние элементы: элемент столбца, который необходимо занулить делиться на элемент главной диагонали текущего столбца и затем полученное значение для экономии памяти запоминается на месте элемента, которые надо было занулить, а вся текущая строка поэлементно умножается на данные коэффициент, и затем полученное значение вычитается из элементов текущей строки, результат запоминается в данном элементе строки. Тут же происходит накопительное вычисление определителя: перед циклом значение определителя равно 1, каждый новый полученный элемент главной диагонали умножается на переменную определителя. После цикла происходит подсчет кол-ва перестановок: если не четно определитель домножатеся на -1. Таким образом, будет получено значение определителя, если матрица невырождена.

Корни находятся в следующем цикле методом обратной прогонки: сначала вычисляется последний корень, затем предпоследний и так далее до первого корня. Затем происходит вывод корней, определителя. Далее вычисляется невязка для каждого корня: внешний цикл – прогонка по строкам, начиная с последней, обнуление значения невязки для каждого корня. Вложенный цикл – прогонка по столбцам в текущей строке: каждый элемент текущей строки домножается на соответствующ корень и значения складываются. Затем во внешнем цикле невязка = полученной значение во вложенном цикле минус значение, соответствующей строки матрицы b, полученной после зануления элементов.

Обратная матрица находится путем решения n систем линейных уравнений, из единичной матрицы, поочередно подставляя каждый столбец данной матрицы на место последнего столбца расширенной матрицы.

Поиск нормы m матриц ||A||*||A^-1|| происходит путем поочередного сложения элементов строк двух матриц, поиска наибольшего элемента и умножения этих элементов.

Расчетные формулы

- Поиск коэффициента домножения:

$$\sum_{j=k}^{m} a_{ij}^{(k)} x_j = b_i^{(k)}, \quad k \le i \le m. \quad C_{pk} = \frac{a_{pk}^{(k)}}{a_{kk}^{(k)}}, \ p > k$$

- Остальные элементы строки изменяются по формуле:

$$a_{pl}^{(k+1)} = a_{pl}^{(k)} - C_{pk}a_{kl}^{(k)}, \;\; b_{p}^{(k+1)} = b_{p}^{(k)} - C_{pk}b_{k}^{(k)}, \;\; k < p, l \leq m.$$

- Поиск корней, методом обратной прогонки:

$$x_k = \frac{b_k^{(k)} - \sum\limits_{j=k+1}^{m} a_{kj}^{(k)} x_j}{a_{kk}^{(k)}}, \;\; k = m, m-1, \ldots, 1.$$

- Вычисление невязки:

$$r_k = b_k - \sum_{i=1}^{m} a_{ki}x_i, \quad 1 \leq k \leq m.$$

- Вычисление определителя у матрицы треугольного (ступенчатого) вида:

$$\det \mathbf{A} = \pm \prod_{k=1}^{m} a_{kk}^{(k)}.$$

  Оценки погрешностей в данном задании происходят по невязки каждого из корней.

Текст программы с комментариями

```
#include "stdafx.h"
#include <iostream>
#include <Conio.h>
#include <Windows.h>
#include <fstream>
using namespace std;
const int max_count = 100;// максимальная размерность матрицы
void read(double matrix[max_count][max_count], double *n, ifstream *fin);// функция чтения
матрицы из файла
void print(double matrix[max_count][max_count], double *n, FILE *file);// функция вывода
расширенной матрицы
void print(double matrix[max_count][max_count], int *n, FILE *file);// вывод матрицы в консоль
и в файл
int search_decision(double matrix[max_count][max_count], double *n, FILE *file);// решение
системы линейных уравнений, вычисление определителя
void norm_matrix(double matrix[max_count][max_count], double
inversia_matrix[max_count][max_count], int *n, FILE *file);// ||A||*||A^-1||
// функция чтения расширенной матрицы из файла
void read(double matrix[max_count][max_count+1], int *n, ifstream *fin)
{
 *fin >> *n;
 for (int i = 0; i < *n; i++)
 {
       for (int j = 0; j < *n+1; j++)
       {
             *fin >> matrix[i][j];
       }
 }
}
```

```
// функция вывода расширенной матрицы в консоль и в файл
void print(double matrix[max_count][max_count + 1], int *n, FILE *file)
{
 for (int i = 0; i < *n; i++)
 {
	for (int j = 0; j < *n+1; j++)
	{
		printf("%10lf ", matrix[i][j]);
		fprintf(file, "%10lf ", matrix[i][j]);
	}
	printf("\n\n");
	fprintf(file, "\n\n");
 }
 cout << endl;
 fprintf(file, "\n");
}
// вывод матрицы в консоль и в файл
void print(double matrix[max_count][max_count], int *n, FILE *file)
{
 for (int i = 0; i < *n; i++)
 {
	for (int j = 0; j < *n; j++)
	{
		printf("%10lf ", matrix[i][j]);
		fprintf(file, "%10lf ", matrix[i][j]);
	}
	printf("\n\n");
	fprintf(file, "\n\n");
 }
 cout << endl;
 fprintf(file, "\n");
}
// решение системы линейных уравнений, вычисление определителя, невязки, обратной матрицы
int search_decision(double matrix[max_count][max_count + 1], int *n, FILE *file)
{
 double *X=new double[max_count];// корни уравнения
 double temp_matrix[max_count][max_count];// временная матрица
 double inversia_matrix[max_count][max_count];// обратная матрица
 int *current_line=new int;// номер строки, с главным элементом
 double *big_number=new double;// значение главного элемента в столбце
 double *opredelitel = new double;// определитель матрицы
 *opredelitel = 1;// начальное значение
 bool flag = false;// флаг на подсчет перестановок
 bool flag_zero;// флаг проверки, если элемент нулевой
 // копирование матрицы
 for (int i = 0; i < *n; i++)
 {
	for (int j = 0; j < *n; j++)
	{
		temp_matrix[i][j] = matrix[i][j];
	}
 }
 // заполнение элементов обртаной матрицы как единичной
 for (int i = 0; i < *n; i++)
 {
	for (int j = 0; j < *n; j++)
	{
		if (i == j)
		{
			inversia_matrix[i][j] = 1;
		}
```

```
                else
                {
                        inversia_matrix[i][j] = 0;
                }
        }

 }
 // цикл по столбцам, в котором происходит зануление элементов ниже гл диагонали и поиск главного
главного эл-та каждого столбца
 for (int j = 0; j < *n; j++)
 {
        flag_zero=false;
        *current_line = j;
        *big_number = matrix[j][j];// пусть самый большой - первый элемент столбца, на гл диаганали
        if (!*big_number)// если на главной диагонали нулевой элемент
        {
                flag_zero = true;
        }
        for (int i = j + 1; i < *n; i++)// цикл по строкам
        {
                // если найден ненулевой элемент больше текущего или элемент на гл диаганали равен
нулю, но текущий элемент нулю не равен
                if ((matrix[i][j]>*big_number||flag_zero)&&matrix[i][j] != 0)
                {
                        if (flag_zero)
                        {
                                flag_zero = false;
                        }
                        *current_line= i;// запоминание его номера
                        *big_number = matrix[i][j];// запоминание значения
                }
        }
        if (flag_zero)// если все элементы оказались нулевые, матрица вырождена
        {
                cout << "ошибка" << endl;
                cout << "определитель матрицы равен 0, завершение работы";
                fprintf(file, "ошибка\nопределитель матрицы равен 0, завершение работы");
                _getch();
                return 1;
        }
        if (*big_number != matrix[j][j])// если найден самый большой элемент
        {
                if (!flag)// подсчет количества перестановок
                {
                        flag = true;
                }
                else
                {
                        flag = false;
                }
                for (int i = 0; i < *n+1; i++)// перестановка строк
                {
                        *big_number = matrix[*current_line][i];
                        matrix[*current_line][i] = matrix[j][i];
                        matrix[j][i] = *big_number;
                }
                for (int i = 0; i < *n; i++)// перестановка строк обратной матрицы
                {
                        *big_number = inversia_matrix[*current_line][i];
                        inversia_matrix[*current_line][i] = inversia_matrix[j][i];
                        inversia_matrix[j][i] = *big_number;
```

```
			}

		}
		//зануление элементов ниже гл диагонали
		for (int i = j + 1; i < *n; i++)
		{
			matrix[i][j] = matrix[i][j] / matrix[j][j];// вычисление коэф. на который будет
происходить умножение
			for (int k = j + 1; k < *n + 1; k++)
			{

				matrix[i][k] = matrix[i][k] - matrix[i][j] * matrix[j][k];
			}
		}
		*opredelitel *= matrix[j][j];// вычисление определителя
	}
	if (flag)// если кол-во перестановок нечетное
	{
		*opredelitel *= (-1);
	}
	//нахождение корней в обратном порядке
	double *temp = new double;
	X[*n - 1] = matrix[*n - 1][*n] / matrix[*n - 1][*n - 1];// вычисление первого с конца корня
	for (int i = *n - 2; i >= 0; i--)
	{
		*temp = matrix[i][*n];
		for (int j = *n - 1; j > i; j--)
		{
			*temp = *temp - matrix[i][j] * X[j];
		}
		X[i] = *temp / matrix[i][i];// вычисление следующих корней в обратном порядке
	}
	// вывод решеения системы
	cout << "Решение системы:" << endl;
	fprintf(file, "Решение системы:\n");
	for (int i = 0; i < *n; i++)
	{
		printf(" X%d = %lf\n", i + 1, X[i]);
		fprintf(file, " X%d = %lf\n", i + 1, X[i]);
	}
	cout << endl;
	//вывод определителя
	printf("Определитель равен: %lf\n", *opredelitel);
	fprintf(file, "Определитель равен: %lf\n", *opredelitel);
	// нахождение и вывод невязки
	double *nevyzka = new double;// значение невязки
	cout << "Невязка равна:" << endl;
	fprintf(file, "Невязка равна:\n");
	for (int i = *n - 1; i >= 0; i--)
	{
		*nevyzka = 0;
		for (int j = i; j < *n; j++)
		{
			*nevyzka += matrix[i][j] * X[j];
		}
		*nevyzka -= matrix[i][*n];
		cout << "X" << i + 1 << " = " << *nevyzka << endl;
		fprintf(file, "X%d = %lf\n", i + 1, *nevyzka);
	}
	// решение n уравнений для поиска обратной матрицы
	for (int j = 0; j < *n; j++)// цикл по столбцам обратной матрицы
```

```
 {
	for (int k = 0; k < *n - 1; k++)
	{
		for (int i = k + 1; i < *n; i++)
		{
			inversia_matrix[i][j] = inversia_matrix[i][j] - matrix[i][k] *
inversia_matrix[k][j];// домножение элементов обратной матрицы на коэф.
		}
	}
	inversia_matrix[*n - 1][j] = inversia_matrix[*n - 1][j] / matrix[*n - 1][*n - 1];
	for (int i = *n - 2; i >= 0; i--)
	{
		*temp = inversia_matrix[i][j];
		for (int k = *n - 1; k > i; k--)
		{
			*temp -= matrix[i][k] * inversia_matrix[k][j];
		}
		inversia_matrix[i][j] = *temp / matrix[i][i];// вычисление элемента обратной
матрицы
	}
 }
 // вывод обратной матрицы
 cout << "\nОбратная матрица:" << endl;
 fprintf(file, "\nОбратная матрица:\n");
 print(inversia_matrix, n, file);
 norm_matrix(temp_matrix, inversia_matrix, n, file);// ||A||*||A^-1||
 delete[] X;
 delete current_line, big_number, opredelitel, temp, nevyzka;
 return 0;
}
// ||A||*||A^-1||
void norm_matrix(double matrix[max_count][max_count], double
inversia_matrix[max_count][max_count], int *n, FILE *file)
{
 double norm_matrix1 = 0;// норма матрицы
 double temp1;// временная переменная для суммы по столбцам
 double norm_matrix2 = 0;// норма обратной матрицы
 double temp2;// временная переменная для суммы по столбцам
 // поиск нормы двух матриц
 for (int i = 0; i < *n; i++)
 {
	temp1 = 0;
	temp2 = 0;
	for (int j = 0; j < *n; j++)// сложение по столбцам
	{
		temp1 += matrix[i][j];
		temp2 += inversia_matrix[i][j];
	}
	if (!i)
	{
		norm_matrix1 = temp1;
		norm_matrix2 = temp2;
	}
	else// поиск наибольшей суммы
	{
		if (norm_matrix1 < temp1)
		{
			norm_matrix1 = temp1;
		}
		if (norm_matrix2 < temp2)
		{
```

```
                    norm_matrix2 = temp2;
                }
        }
 }
 norm_matrix1 *= norm_matrix2;// умножение норм двух матриц
 printf("\n||A||*||A^-1||: %lf\n", norm_matrix1);
 fprintf(file, "\n||A||*||A^-1||: %lf\n", norm_matrix1);
}
int _tmain(int argc, _TCHAR* argv[])
{
 SetConsoleCP(1251);// подключение русскоязычного ввода/вывода
 SetConsoleOutputCP(1251);
 ifstream fin("input.dat");
 if (fin.is_open() == false)
 {
        cout << "Ошибка открытия файла input.dat\nзавершение работы" << endl;
        _getch();
        return 1;
 }
 double matrix[max_count][max_count+1];// расширенная матрица
 int n;// размерность матрицы

 read(matrix, &n, &fin);
 fin.close();
 FILE *file = fopen("output.dat", "w");
 if (file == NULL)
 {
        cout << "Ошибка открытия файла output.dat\nзавершение работы" << endl;
        _getch();
        return 1;
 }
 cout << "Расширенная матрица:" << endl;
 fprintf(file, "Расширенная матрица:\n");
 print(matrix, &n, file);
 if (search_decision(matrix, &n, file))
 {
        return 1;
 }
 fclose(file);
 _getch();
 return 0;
}
```

Тестовые примеры

1.

```
Расширенная матрица:
  1.000000   3.000000   8.000000  25.000000

  5.000000   4.000000   2.000000  10.000000

 10.000000   5.000000   1.000000  15.000000


Решение системы:
 X1 = 2.098765
 X2 = -1.913580
 X3 = 3.580247

Определитель равен: -81.000000
Невязка равна:
X3 = 0.000000
X2 = 0.000000
X1 = 0.000000

Обратная матрица:
  0.074074  -0.456790   0.320988

 -0.185185   0.975309  -0.469136

  0.185185  -0.308642   0.135802



||A||*||A^-1||: 5.135802
```

2.

```
Расширенная матрица:
  1.000000   8.000000   4.000000   5.000000   6.000000  10.000000

  8.000000   5.000000   4.000000  12.000000   7.000000  11.000000

  3.000000  15.000000   4.000000   5.000000   4.000000  20.000000

  8.000000   4.000000   7.000000   1.000000   3.000000  25.000000

  4.000000   8.000000   7.000000   2.000000   4.000000   5.000000


Решение системы:
 X1 = 8.093610
 X2 = 2.379128
 X3 = -10.709205
 X4 = -8.562634
 X5 = 11.420559

Определитель равен: 5117.000000
Невязка равна:
X5 = 0
X4 = 0
X3 = 0
X2 = 3.55271e-015
X1 = 0

Обратная матрица:
  0.084034  -0.034200   0.092437   0.309361  -0.390659

  0.008403  -0.038304   0.109244   0.037913  -0.083252

 -0.319328   0.106703  -0.151261  -0.370920   0.721712

 -0.310924   0.184679  -0.042017  -0.356263   0.452414

  0.613445  -0.168263  -0.025210   0.442056  -0.682040



||A||*||A^-1||: 6.479578
```

3.

```
Расширенная матрица:
  4.000000   2.000000  -3.000000   1.000000

 -8.000000  -7.000000   1.000000   4.000000

  4.000000   2.000000  -3.000000  10.000000


ошибка
определитель матрицы равен 0, завершение работы
```

4.

```
Расширенная матрица:
  1.000000   2.000000   3.000000   5.000000   1.000000  15.000000

  1.000000   2.000000   3.000000   5.000000   1.000000  10.000000

  4.000000   5.000000   8.000000   1.000000   4.000000   2.000000

  4.000000   8.000000   7.000000   2.000000   7.000000   1.000000

  8.000000   4.000000   3.000000   1.000000   4.000000   1.000000


ошибка
определитель матрицы равен 0, завершение работы
```

5.

```
Расширенная матрица:
  1.000000   5.000000   4.000000  10.000000

  2.000000   8.000000   4.000000  20.000000

  4.000000   9.000000   1.000000  15.000000


Решение системы:
 X1 = -11.428571
 X2 = 7.142857
 X3 = -3.571429

Определитель равен: -14.000000
Невязка равна:
X3 = 0
X2 = 0
X1 = -1.77636e-015

Обратная матрица:
  2.000000  -2.214286   0.857143

 -1.000000   1.071429  -0.285714

  1.000000  -0.785714   0.142857



||A||*||A^-1||: 9.000000
```

Выполнение других пунктов задания

Решить систему

$$\begin{pmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{pmatrix} \mathbf{x} = \mathbf{b},$$

выбирая $\mathbf{b}$ равным

$$\begin{pmatrix} 23 \\ 32 \\ 33 \\ 31 \end{pmatrix}, \quad \begin{pmatrix} 23.01 \\ 31.99 \\ 32.99 \\ 31.01 \end{pmatrix}, \quad \begin{pmatrix} 23.1 \\ 31.9 \\ 32.9 \\ 31.1 \end{pmatrix}$$

Пояснить полученные результаты.

1.

```
Расширенная матрица:
  5.000000   7.000000   6.000000   5.000000  23.000000

  7.000000  10.000000   8.000000   7.000000  32.000000

  6.000000   8.000000  10.000000   9.000000  33.000000

  5.000000   7.000000   9.000000  10.000000  31.000000


Решение системы:
 X1 = 1.000000
 X2 = 1.000000
 X3 = 1.000000
 X4 = 1.000000

Определитель равен: 1.000000
Невязка равна:
X4 = 0
X3 = 0
X2 = 0
X1 = 0

Обратная матрица:
 68.000000 -41.000000 -17.000000  10.000000

-41.000000  25.000000  10.000000  -6.000000

-17.000000  10.000000   5.000000  -3.000000

 10.000000  -6.000000  -3.000000   2.000000


||A||*||A^-1||: 660.000000
```

2.

```
Расширенная матрица:
  5.000000   7.000000   6.000000   5.000000  23.010000

  7.000000  10.000000   8.000000   7.000000  31.990000

  6.000000   8.000000  10.000000   9.000000  32.990000

  5.000000   7.000000   9.000000  10.000000  31.010000


Решение системы:
 X1 = 2.360000
 X2 = 0.180000
 X3 = 0.650000
 X4 = 1.210000

Определитель равен: 1.000000
Невязка равна:
X4 = 0
X3 = 0
X2 = 0
X1 = 0

Обратная матрица:
 68.000000 -41.000000 -17.000000  10.000000

-41.000000  25.000000  10.000000  -6.000000

-17.000000  10.000000   5.000000  -3.000000

 10.000000  -6.000000  -3.000000   2.000000



||A||*||A^-1||: 660.000000
```

3.

```
Расширенная матрица:
  5.000000   7.000000   6.000000   5.000000  23.100000

  7.000000  10.000000   8.000000   7.000000  31.900000

  6.000000   8.000000  10.000000   9.000000  32.900000

  5.000000   7.000000   9.000000  10.000000  31.100000


Решение системы:
 X1 = 14.600000
 X2 = -7.200000
 X3 = -2.500000
 X4 = 3.100000

Определитель равен: 1.000000
Невязка равна:
X4 = 0
X3 = 0
X2 = 1.11022e-016
X1 = 7.10543e-015

Обратная матрица:
 68.000000 -41.000000 -17.000000  10.000000

-41.000000  25.000000  10.000000  -6.000000

-17.000000  10.000000   5.000000  -3.000000

 10.000000  -6.000000  -3.000000   2.000000



||A||*||A^-1||: 660.000000
```

Стоит отметить, что с учетом погрешности, результат вычислений оказался верным, так как если в данные системы линейных уравнений вместо неизвестных подставить полученные корни, то результат совпадет со столбцом свободных элементов.
По полученным результатам можно сказать, что даже при незначительном изменении значений столбца свободных элементов (буквально на одну сотую или десятую), получаются совершенно другие корни уравнений. Этот пример показывает, насколько важно учитывать погрешность вычислений, даже незначительную.

Выводы

Данная работа оказалась очень полезной в плане понимания программной реализации решения систем линейных уравнений методом Гаусса. В теории было доказано, что программное вычисление определителя классическим способом матрицы 30-го порядка может занять до 10^9 лет, поэтому очень важно было найти подходящий алгоритм решение, который бы удовлетворял временным и количественным требованиям решения задачи. Также очень важно вычислять и оценивать погрешность вычислений, так как даже небольшие погрешности могут привести к большим вычислительным ошибкам. Уверен, что полученные знания обязательно пригодятся мне в будущем.