Белорусский государственный университет

Вычислительные методы алгебры

Лабораторная работа 1

Выполнил учащийся 2 курса 9 группы специальности «Прикладная математика»

Крученков Евгений

Преподаватель:

Ассистент кафедры вычислительной математики ФПМИ

Горбачёва Юлия Николаевна

Минск

2021

**Содержание:**

Постановка задачи ------------------------------------------------------------------ 3

Краткие теоретические сведения ------------------------------------------------ 4-5

Листинг программы ---------------------------------------------------------------- 6-13

Результаты --------------------------------------------------------------------------- 14-16

Выводы ------------------------------------------------------------------------------- 16

Постановка задачи

Написать и отладить программу, реализующую метод Гаусса с выбором главного элемента по столбцу для численного решения систем линейных алгебраических уравнений *Ax = f*, вычисления обратной матрицы *А-1*. Предусмотреть сообщения, предупреждающие о невозможности решения указанных задач с заданной матрицей *A* порядка *n*.

Выполнить следующие пункты задания.

1. Найти решение системы линейных алгебраических уравнений с матрицей *А* порядка *n* = 10.  
   Для заполнения матрицы *А* использовать случайные числа из диапазона от -100 до 100. В качестве точного решения взять вектор *x* = *(1,2,…,10)T*. Правую часть *f* задать умножением матрицы А на вектор x: *f = Ax*.  
   В результатах необходимо привести следующую информацию: матрица *А* (построчно), вектор *f*, точное решение x, полученное приближенное решение , относительная погрешность вида .
2. Вычислить для матрицы *А* из пункта 1 обратную матрицу *А-1*.  
   В результатах необходимо привести следующую информацию: обратная матрица *А-1* (построчно), матрица *А-1А* (построчно).
3. Оценить точность решения системы линейных алгебраических уравнений. Для этого провести эксперимент, который охватывает матрицы порядка от 6 до 106 (через 10 порядков). Для этого сгенерировать случайные матрицы *А*, задать точное решение *x* и образовать правые части *f = Ax*. Провести анализ точности вычисленного решения от порядка матрицы. Результаты представить в виде таблицы.  
   Для заполнения матрицы *А* использовать случайные числа из диапазона от -100 до 100. В качестве точного решения взять вектор *x* *= (1,2,…,n)T* , где *n* – порядок матрицы.

Краткие теоретические сведения

Метод Гаусса – классический метод решения системы линейных алгебраических уравнений (СЛАУ). Назван в честь немецкого математика Карла Фридриха Гаусса.

Это метод последовательного исключения переменных, когда с помощью элементарных преобразований система уравнений приводится к равносильной системе треугольного вида, из которой последовательно, начиная с последних (по номеру), находятся все переменные системы.

Алгоритм решения СЛАУ методом Гаусса подразделяется на два этапа.

* На первом этапе осуществляется так называемый прямой ход, когда путём элементарных преобразований над строками систему приводят к ступенчатой или треугольной форме, либо устанавливают, что система несовместна.   
  Для этого среди элементов первого столбца матрицы выбирают ненулевой, перемещают содержащую его строку в крайнее верхнее положение, делая эту строку первой. Далее ненулевые элементы первого столбца всех нижележащих строк обнуляются путём вычитания из каждой строки первой строки, домноженной на отношение первого элемента этих строк к первому элементу первой строки. После того, как указанные преобразования были совершены, первую строку и первый столбец мысленно вычёркивают и продолжают, пока не останется матрица нулевого размера. Если на какой-то из итераций среди элементов первого столбца не нашёлся ненулевой, то переходят к следующему столбцу и проделывают аналогичную операцию.

Формулы прямого хода:

= , k, j = 1, n

= , k = 1, n

k = 1, n-1, j = k+1, n

=

= , i = k+1,n

=

= , i = k+1,n

=

* На втором этапе осуществляется так называемый обратный ход, суть которого заключается в том, чтобы выразить все получившиеся базисные переменные через небазисные и построить фундаментальную систему решений, либо, если все переменные являются базисными, то выразить в численном виде единственное решение системы линейных уравнений.   
  Эта процедура начинается с последнего уравнения, из которого выражают соответствующую базисную переменную (а она там всего одна) и подставляют в предыдущие уравнения, и так далее, поднимаясь по «ступенькам» наверх. Каждой строчке соответствует ровно одна базисная переменная, поэтому на каждом шаге, кроме последнего (самого верхнего), ситуация в точности повторяет случай последней строки.

Формулы обратного хода:

=

= – , k = n-1, … ,1

Метод Гаусса требует *O({\displaystyle O(n^{3})})*арифметических операций.

Достоинства метода:

* Для матриц ограниченного размера менее трудоёмкий по сравнению с другими методами.
* Позволяет однозначно установить, совместна система или нет, и, если совместна, найти её решение.
* Позволяет найти максимальное число линейно независимых уравнений — ранг матрицы системы.

Листинг программы:

Header.h:

#pragma once

#include <iostream>

#include <stdio.h>

#include <iomanip>

#include <fstream>

#include <string>

#include <time.h>

using namespace std;

class Matr

{

friend class Sist;

double\*\* M;

int n;

public:

Matr()

{

this->n = 1;

M = new double\* [n];

for (int i = 0; i < n; i++)

{

M[i] = new double[n];

}

srand(time(0));

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

M[i][j] = (((double)rand() / RAND\_MAX) \* 2 - 1) \* 100;

}

}

}

Matr(int n)

{

this->n = n;

M = new double\* [n];

for (int i = 0; i < n; i++)

{

M[i] = new double[n];

}

srand(time(0));

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

M[i][j] = (((double)rand() / RAND\_MAX) \* 2 - 1) \* 100;

}

}

}

Matr(const Matr& B)

{

this->n = B.n;

M = new double\* [n];

for (int i = 0; i < n; i++)

{

M[i] = new double[n];

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

M[i][j] = B.M[i][j];

}

}

}

~Matr()

{

for (int i = 0; i < n; i++)

{

delete[] M[i];

}

delete[] M;

}

friend ostream& operator <<(ostream& os, Matr& A)

{

os << fixed << showpoint << setprecision(3);

for (int i = 0; i < A.n; i++)

{

for (int j = 0; j <A.n; j++)

{

os << A.M[i][j] << " ";

}

os << endl;

}

return os;

}

Matr& operator = (const Matr& B)

{

if (this == &B)

return \*this;

for (int i = 0; i < n; i++)

{

delete[] M[i];

}

n = B.n;

delete[] M;

M = new double\* [n];

for (int i = 0; i < n; i++)

{

M[i] = new double[n];

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

M[i][j] = B.M[i][j];

}

}

return \*this;

}

Matr operator \* (const Matr& B)

{

if (n != B.n) throw "Матрицы имеют разные размеры";

Matr P(n);

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

P.M[i][j] = 0;

for (int k = 0; k < n; k++)

{

P.M[i][j] += M[i][k] \* B.M[k][j];

}

}

}

return P;

}

Matr Obrat()

{

Matr E(n), Obr(n), O(\*this);

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (i == j) E.M[i][j] = 1;

else E.M[i][j] = 0;

}

}

for (int i = 0; i < n; i++)

{

//выбор главного элемента по столбцу

double max = abs(O.M[i][i]);

int gl = i;

for (int j = i; j < n; j++)

{

if (abs(O.M[j][i]) > max)

{

max = O.M[j][i];

gl = j;

}

}

if (max == 0) throw "Все главные элементы нулевые";

//исключение в случае, когда все главные элементы нулевые

for (int j = i; j < n; j++)

{

swap(O.M[i][j], O.M[gl][j]);

}

for (int j = 0; j < n; j++)

{

swap(E.M[i][j], E.M[gl][j]);

}

//прямой ход метода Гаусса

for (int j = i + 1; j < n; j++)

{

O.M[i][j] /= O.M[i][i];

}

for (int j = 0; j < n; j++)

{

E.M[i][j] /= O.M[i][i];

}

O.M[i][i] /= O.M[i][i];

for (int j = i + 1; j < n; j++)

{

for (int k = i + 1; k < n; k++)

{

O.M[j][k] -= O.M[i][k] \* O.M[j][i];

}

for (int k = 0; k < n; k++)

{

E.M[j][k] -= E.M[i][k] \* O.M[j][i];

}

O.M[j][i] -= O.M[i][i] \* O.M[j][i];

}

}

//обратный ход метода Гаусса

for (int k = 0; k < n; k++)

{

for (int i = (n - 1); i >= 0; i--)

{

Obr.M[i][k] = E.M[i][k];

for (int j = n - 1; j > i; j--)

{

Obr.M[i][k] -= O.M[i][j] \* Obr.M[j][k];

}

}

}

return Obr;

}

};

class Sist

{

friend class Matr;

private:

Matr A;

double\* X;

double\* F;

double\* X\_;

double pogr;

public:

int size()

{

return A.n;

}

void ShowPogr()

{

cout <<scientific << pogr << fixed << showpoint << setprecision(3);

}

Sist(const Matr& B)

{

A = B;

X = new double[B.n];

for (int i = 0; i < B.n; i++)

{

X[i] = 1 + i;

}

F = new double[B.n];

for (int i = 0; i < size(); i++)

{

F[i] = 0;

for (int j = 0; j < size(); j++)

{

F[i] += A.M[i][j] \* X[j];

}

}

X\_ = new double[B.n];

for (int i = 0; i < size(); i++)

{

X\_[i] = 0;

}

}

Sist(const Sist& B)

{

A = B.A;

X = new double[size()];

F = new double[size()];

X\_ = new double[size()];

for (int i = 0; i < size(); i++)

{

for (int j = 0; j < size(); j++)

{

A.M[i][j] = B.A.M[i][j];

}

}

for (int i = 0; i < size(); i++)

{

X[i] = B.X[i];

}

for (int i = 0; i < size(); i++)

{

F[i] = B.F[i];

}

for (int i = 0; i < size(); i++)

{

X\_[i] = B.X\_[i];

}

}

~Sist()

{

delete[] X;

delete[] X\_;

delete[] F;

}

void Gauss()

{

//выбор главного элемента по столбцу

for (int i = 0; i < size(); i++)

{

double max = abs(A.M[i][i]);

int gl = i;

for (int j = i; j < size(); j++)

{

if (abs(A.M[j][i]) > max)

{

max = A.M[j][i];

gl = j;

}

}

if (max == 0) throw "Все главные элементы нулевые";

//исключение в случае, когда все главные элементы нулевые

for (int j = i; j < size(); j++)

{

swap(A.M[i][j], A.M[gl][j]);

}

swap(F[i], F[gl]);

//прямой ход метода Гаусса

for (int j = i+1; j < size(); j++)

{

A.M[i][j] /= A.M[i][i];

}

F[i] /= A.M[i][i];

A.M[i][i] /= A.M[i][i];

for (int j = i+1; j < size(); j++)

{

for (int k = i+1; k < size(); k++)

{

A.M[j][k] -= A.M[i][k] \* A.M[j][i];

}

F[j] -= F[i] \* A.M[j][i];

A.M[j][i] -= A.M[i][i] \* A.M[j][i];

}

}

//обратный ход метода Гаусса

for (int i = (size()-1); i >= 0; i--)

{

X\_[i] = F[i];

for (int j = size()-1; j > i; j--)

{

X\_[i] -= A.M[i][j] \* X\_[j];

}

}

//вычисление относительной погрешности

double maxX = abs(X[0]), max\_X = abs(X[0]-X\_[0]);

for (int i = 0; i < size(); i++)

{

if (max\_X < abs(X[i] - X\_[i]))

{

max\_X = abs(X[i] - X\_[i]);

}

if (maxX < abs(X[i]))

{

maxX = abs(X[i]);

}

}

pogr = max\_X/maxX ;

}

friend ostream& operator <<(ostream& os, Sist& B)

{

os << "Матрица системы:" << endl << fixed << showpoint << setprecision(3);

os << B.A << endl;

os << "Столбец свободных членов:" << endl;

for (int i = 0; i < B.size(); i++)

{

os << B.F[i] << " ";

os << endl;

}

os << "Точное решение X:" << endl;

for (int i = 0; i < B.size(); i++)

{

os << B.X[i] << " ";

os << endl;

}

os << "Приближенное решение X\_:" << endl;

for (int i = 0; i < B.size(); i++)

{

os << fixed << showpoint << setprecision(20) << B.X\_[i] << " ";

os << endl;

}

os << "Относительная погрешность: " << fixed << showpoint << setprecision(20) << B.pogr<< endl << fixed << showpoint << setprecision(3);

return os;

}

Sist& operator = (const Sist& B)

{

if (this == &B)

return \*this;

A = B.A;

delete[] X;

delete[] X\_;

delete[] F;

X = new double[A.n];

F = new double[A.n];

X\_ = new double[A.n];

for (int i = 0; i < A.n; i++)

{

X[i] = B.X[i];

}

for (int i = 0; i < A.n; i++)

{

F[i] = B.F[i];

}

for (int i = 0; i < A.n; i++)

{

X\_[i] = B.X\_[i];

}

return \*this;

}

};

Main.cpp:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include "Header.h"

#include <fstream>

#include<time.h>

using namespace std;

ifstream fin;

ofstream fout;

int main()

{

srand(time(0));

setlocale(LC\_ALL, "ru");

Matr M(10), N(M), L(N);

Sist A(M), B(A);

//1

cout << "Матрица А системы: " << endl << M << endl;

A.Gauss();

cout << "Система АX = f после решения методом Гаусса: " << endl;

cout << A<<endl;

cout << endl;

//2

cout << "Матрица М:"<<endl;

cout << M << endl;

N = M.Obrat();

cout << "Матрица N, обратная М:" << endl;

cout << N << endl;

cout << "Матрица M\*N:" << endl;

L = M \* N;

cout << L << endl;

cout << endl;

cout << "Порядок матрицы: Относительная погрешность:" << endl;

for (int i = 6; i <= 106; i += 10)

{

Sist A1(i);

A1.Gauss();

cout << i << " ";

A1.ShowPogr();

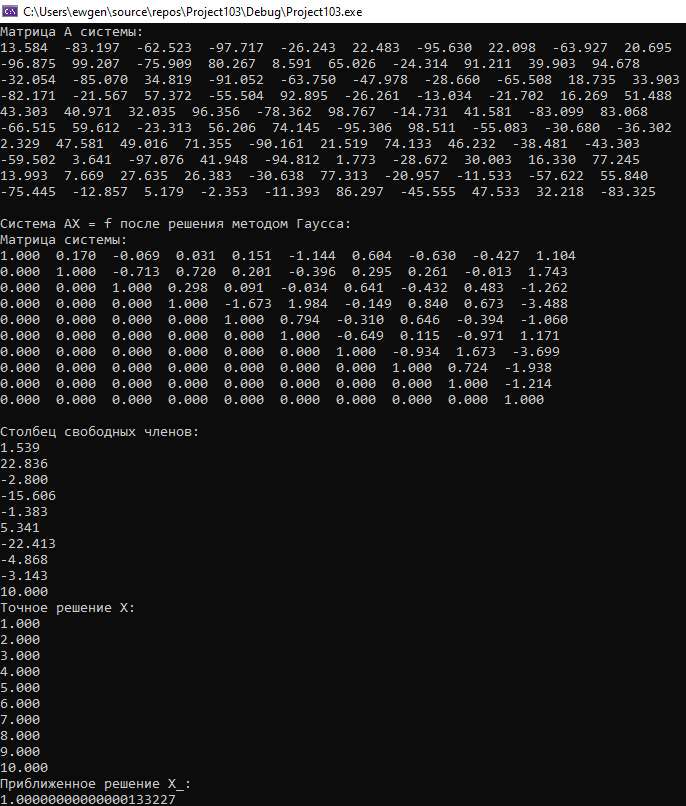
cout << endl;

}

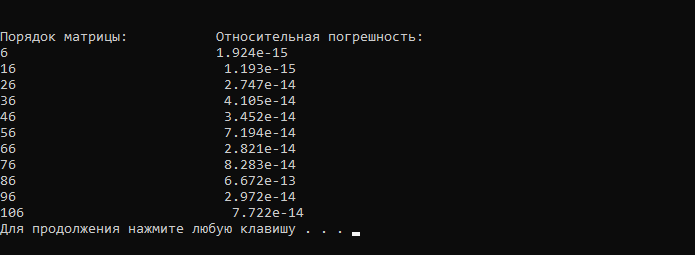
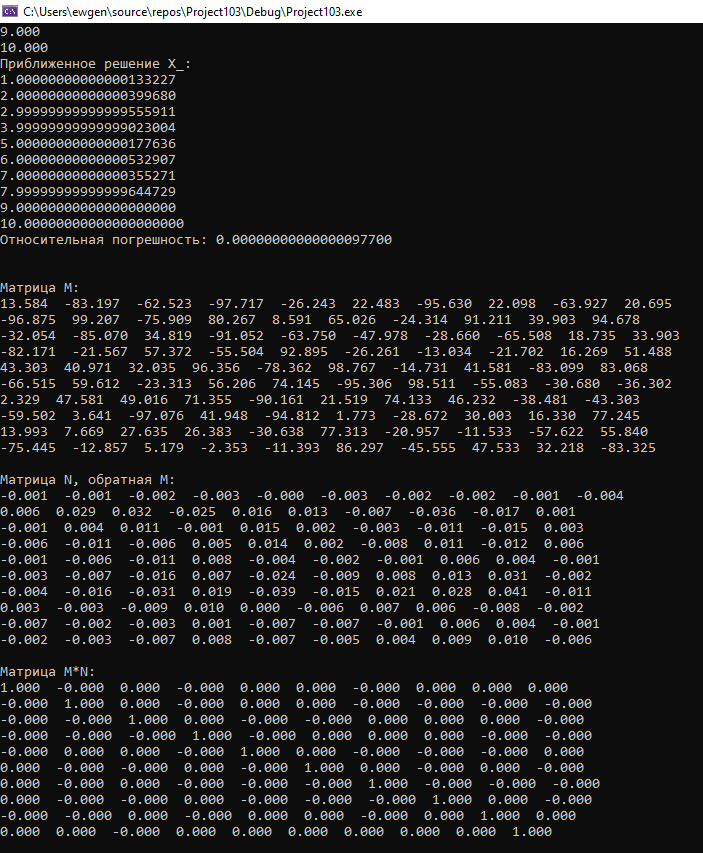
system("pause");

return 0;

}



Результаты



Выводы

Метод Гаусса решения линейных неоднородных систем с выбором главного элемента по столбцу позволяет достаточно точно (с погрешностью меньше 10-13 для элементов матрицы в диапазоне от -100 до 100) вычислить решение системы.

Алгоритмическая сложность данного метода равна *O (n3).*

Из таблицы погрешностей в задании 3, можно заметить, что в общем случае при возрастании порядка матрицы растёт относительная погрешность её решения.