**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»**

Физико-технический факультет

Кафедра компьютерных технологий

Дисциплина «Вычислительная математика»

Отчет к лабораторной работе № 2

«Решение системы линейных уравнений методом Гаусса с выбором главного элемента»

Выполнил:

студент группы ИВТ-6

Давыденко Дмитрий Александрович

Принял:

асс. Пшеничный Кирилл Анатольевич

Донецк 2022

**ЗАДАНИЕ**

Для тех, кто программирует, необходимо написать программу метода Гаусса для решения системы *n* линейных уравнений *Ax*=*f*. Продемонстрировать её работоспособность на примере решения системы линейных уравнений с матрицей Гильберта *A*, элементы которой *aij* =1/(*i*+*j*-1), *i*,*j*=1.. *n*. Свободный член системы уравнений задайте в виде *fi* = *n*/*i2*. Проведите расчёты без выбора и с выбором главного элемента матрицы. Можно взять в Internet сторонние коды метода Гаусса и модифицировать их под поставленную задачу.

Для тех, кто не программирует, сделайте соответствующее расчётное задание с преподавателем в лаборатории.

Всем, кто программирует и не программирует, необходимо найти число обусловленности рассматриваемой матрицы – Cond(*A*)=||*A*||·||*A*-1||. Это можно сделать в сторонних приложениях, например, Python, MathCad…

Размерность *n* для системы уравнений возьмите в соответствии с вашим номером в списке группы и добавьте к нему цифру 3.

Сделайте выводы из всех полученных результатов.

**ХОД РАБОТЫ**

Репозиторий проекта:

<https://github.com/uncleDimasik/computational-mathematics-22-23-laba-2>

#include <iostream>

using namespace std;

//вывод системы уравнений

void sysout(double\*\* a, double\* y, int n)

{

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

{

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

{

cout << a[i][j] << "\*x" << j + 1;

if (j < n - 1)

cout << " + ";

}

cout << " = " << y[i] << endl;

}

return;

}

double\* gauss(double\*\* a, double\* y, int n)

{

double\* x, max;

int k, index;

const double eps = 0.00001; //точность

x = new double[n];

k = 0;

while (k < n)

{

//поиск главного элемента

max = abs(a[k][k]);

index = k;

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

{

if (abs(a[i][k]) > max)

{

max = abs(a[i][k]);

index = i;

}

}

//перестановка строк

if (max < eps)

{

//нет ненулевых диагональных элементов

cout << "Решение получить невозможно из-за нулевого столбца ";

cout << index << " матрицы A" << endl;

return 0;

}

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

{

double temp = a[k][j];

a[k][j] = a[index][j];

a[index][j] = temp;

}

double temp = y[k];

y[k] = y[index];

y[index] = temp;

//нормализация уравнений

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

{

double temp = a[i][k];

if (abs(temp) < eps) continue; //для нулевого коэффициента пропустить

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

a[i][j] = a[i][j] / temp;

y[i] = y[i] / temp;

if (i == k) continue; //уравнение не вычитать само из себя

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

a[i][j] = a[i][j] - a[k][j];

y[i] = y[i] - y[k];

}

k++;

}

//обратная подстановка

for (k = n - 1; k >= 0; k--)

{

x[k] = y[k];

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

y[i] = y[i] - a[i][k] \* x[k];

}

return x;

}

double\* gauss2(double\*\* a, double\* y, int n) //без выбора главного элемента

{

double\* x;

int k;

const double eps = 0.00001;

x = new double[n];

k = 0;

while (k < n)

{

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

{

double temp = a[i][k];

if (abs(temp) < eps) continue;

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

a[i][j] = a[i][j] / temp;

y[i] = y[i] / temp;

if (i == k) continue;

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

a[i][j] = a[i][j] - a[k][j];

y[i] = y[i] - y[k];

}

k++;

}

for (k = n - 1; k >= 0; k--)

{

x[k] = y[k];

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

y[i] = y[i] - a[i][k] \* x[k];

}

return x;

}

int main()

{

setlocale(LC\_ALL, "Russian");

double\*\* a, \* y, \* x;

int n = 12+3;// номер по списку + 3. Я 12-й

a = new double\* [n];//rowCount

y = new double[n];

for (int i = 0; i < n; i++)//инициализация матрицы гильберта

{

a[i] = new double[n]; //made a new column

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

{

a[i][j] = 1.0 / ((i + 1) + (j + 1) - 1.0);

}

}

for (int i = 0; i < n; i++)// Свободный член системы уравнений задайте в виде fi = n/i^2

{

y[i] = ((double)n / ((i + 1) \* (i + 1)));

}

sysout(a, y, n);

cout << "\nС выбором" << endl;

x = gauss(a, y, n);

cout << endl;

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

cout << "x[" << i + 1 << "]=" << x[i] << endl;

cout << "\nБез выбора" << endl;

x = gauss2(a, y, n);

cout << endl;

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

cout << "x[" << i + 1 << "]=" << x[i] << endl;

return 0;

}

**Число обусловленности рассматриваемой матрицы**

**import** **numpy** **as** **np**

**from** **numpy** **import** linalg **as** LA

**import** **math**

**def** printMatrix(n):

H = [[**None** **for** \_ **in** range(n)] **for** \_\_ **in** range(n)]

**for** i **in** range(n):

**for** j **in** range(n):

*# using the formula to generate*

*# hilbert matrix*

H[i][j] = 1 / ((i + 1) + (j + 1) - 1)

**return** H

*# 12+3*

print(LA.cond(printMatrix(12 + 3)))

**ВЫВОДЫ**

Заметим, что в методе последовательного исключения Гаусса вычисления возможны, если ведущие элементы системы  . Добиться выполнения этого условия можно, переставляя элементы строк и столбцов матрицы. Но среди ведущих элементов могут оказаться очень маленькие по абсолютной величине. При делении на такие ведущие элементы получается большая погрешность округления (вычислительная погрешность). Чтобы избежать сильного влияния вычислительной погрешности на решение, применяется метод Гаусса с выбором главного элемента

**Контрольные вопросы**

1. Метод Гаусса O(n^3)

Удаление первого столбца потребует n сложений и n умножений для n−1 строк. Следовательно, количество операций для первого столбца равно 2n(n−1). Для второго столбца у нас есть n-1 сложений и n-1 умножений, и мы делаем это для (n-2) строк, что дает нам 2(n-1)(n-2). Следовательно,

общее количество операций, необходимых для полной декомпозиции, можно записать как

 отбрасываем коэф. и получаем O(n^3) сложность по времени.

***Крамер*** очень неэффективен, временная сложность O (n! × n) с наивным алгоритмом поиска определителя.

Для выполнения **прогонки** в трёхдиагональной СЛАУ из n уравнений с n неизвестными в параллельном варианте требуется последовательно выполнить следующие ряды:

* n рядов делений (в каждом из рядов, кроме одного, по 2 деления),
* по 2n−2 рядов умножений и сложений/вычитаний (в n−1 рядов по 2 операции, в n−1 - по одной).

Таким образом, прогонка относится к алгоритмам со сложностью O(n).

2. Среди элементов матрицы  выберем наибольший по модулю, называемый главным, элемент. Например, пусть им будет элемент . Строка с номером , содержащая главный элемент, называется главной строкой.

3. Необходимо дописать справа единичную матрицу и с помощью элементарных преобразований

4. Если матрица системы «почти вырождена», то ошибки округления могут привести к неверному результату. В таком случае нужно уметь заранее определить меру «близости» матрицы системы к множеству вырожденных матриц.

Мерой близости матрицы А к вырожденной является величина cond(A) − ­число обусловленности, которое вычисляется по формуле cond(A)=, где − норма матрицы А, а − норма обратной матрицы. (Порядок нахождения нормы матрицы приведен в Причем, чем больше величина cond(A), тем ближе матрица А к вырожденной.

Пределы обусловленности >1

5. Метод прогонки является частным случаем метода Гаусса и используется для решения систем линейных уравнений вида Ax = B, где A — трёхдиагональная матрица. Трёхдиагональной матрицей называется матрица такого вида, где во всех остальных местах, кроме главной диагонали и двух соседних с ней, стоят нули.

Метод прогонки состоит из двух этапов: прямой прогонки и обратной прогонки. На первом этапе определяются прогоночные коэффициенты, а на втором – находят неизвестные x.

6. ***Метод итерации*** — это численный и приближенный метод решения СЛАУ.

**Суть:** нахождение по приближённому значению величины следующего приближения, которое является более точным. Метод позволяет получить значения корней системы с заданной точностью в виде предела последовательности некоторых векторов (итерационный процесс).