|  |  |
| --- | --- |
|  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| Институт Искусственного Интеллекта | | |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры | | |

|  |  |
| --- | --- |
| **Отчет по лабораторной работе №2** | |
| **по дисциплине** | |
| **«** Численные методы **»** | |
| **Вариант 34** | |
| Студент 3-го курса  группы КМБО-02-21 | Бредихин В.А. |
| Преподаватель | Крыжановский Ю.М. |
| Рецензент |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_2023 г. |  |
|  |  |  |
| «Допущен к защите» | «\_\_»\_\_\_\_\_\_\_2023 г. |  |

Москва 2023

Содержание

[Задание № 1. 3](#_Toc27854021)

[Теоретическая часть. 3](#_Toc27854022)

[Практическая часть. 3](#_Toc27854023)

[Задание № 2. 4](#_Toc27854024)

[Теоретическая часть. 4](#_Toc27854025)

[Практическая часть. 4](#_Toc27854026)

[Задание № 3. 5](#_Toc27854027)

[Практическая часть 5](#_Toc27854028)

[Приложения. 6](#_Toc27854029)

Задание № 1.

Разработать программу решения системы линейных уравнений (СЛУ) методом Крамера.

Теоретическая часть.

Метод Крамера(правило Крамера) — способ решения [систем линейных алгебраических уравнений](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BB%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D1%85_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D1%85_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B9) с числом уравнений равным числу неизвестных с ненулевым главным [определителем](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C) [матрицы коэффициентов системы](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)).

Имеется СЛУ AX=B, где , , . Тогда, где определитель матрицы А, определитель матрицы A, в которой i-й столбец заменен на столбец B.

Практическая часть.

Для реализации метода Крамера была написана программа lab2.1.cpp на языке C в среде разработки Visual Studio. Для отладки и тестирования использовались данные решенных систем линейных уравнений. Код программы находится в пункте «Приложения».

Задание № 2.

Разработать программу решения СЛУ методом итераций .

Теоретическая часть.

Метод итераций – метод решения [системы линейных алгебраических уравнений](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BB%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D1%85_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D1%85_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B9). Суть метода заключается в нахождении по приближённому значению величины следующего приближения, являющегося более точным.

Имеется СЛУ AX=B, где , , . Тогда её можно привести к виду:

где

и .

Введя матрицы , систему можно записать в матричном виде:

Получившуюся систему будем решать методом последовательных приближений. За нулевое решение берется столбец свободных членов Затем строится последовательность Если существует предел последовательности , то этот предел и будет являться решением.

Процесс итерации будет сходиться, если какая-нибудь каноническая форма матрицы будет меньше единицы, т.е.

Практическая часть.

Для реализации метода итерации была написана программа lab2.2.cpp на языке C++ в среде разработки Qt Creator. Для отладки и тестирования использовались данные решенных систем линейных уравнений. Код программы находится в пункте «Приложения».

Задание № 3.

3.1. Исследовать сходимость итерационных процессов для конкретной СЛУ и решить ее точным и приближенным методом, разработанным в заданиях 1 и 2.

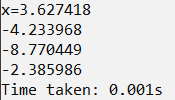
3.2. Сопоставить время расчетов и точность результатов.

Столбец свободных членов:

Практическая часть

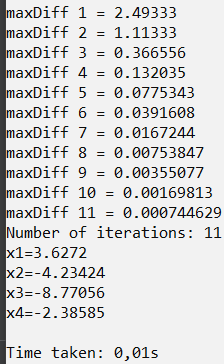
Решая систему методом Крамера, программа нашла ответ – вектор

Время выполнения программы около 1 тысячной секунды.



Решая систему методом итерации, программа выдала ответ – вектор

Время выполнения программы около 1 тысячной секунды.



**Вывод:** в данном случае метод Крамера и метод итерация дали решения отличающиеся не более чем на 0.001. В силу малости числа уравнений в системе, время выполнения программ было около 0.01 секунд, из-за чего трудно судить о разнице во времени работы этих алгоритмов.

Однако можно прийти к следующим выводам, изучая работу этих двух методов.

Метод Крамера:

а) Данное правило разработано и применимо лишь для решения систем линейных алгебраических уравнений с невырожденной матрицей коэффициентов.

б) Реализация данного метода в виде вычислительной процедуры требует выполнения значительного количества арифметических операций и соответственно больших затрат машинного времени.

в) Данный метод дает менее точные результаты, чем другие методы решения систем линейных алгебраических уравнений.

Метод итерации:

а) Имеет простую вычислительную процедуру.

б) Самоисправление ошибок.

в) Не всегда может решить систему уравнений (требуется выполнение условий сходимости).

г) Сходимость итерационных процессов может быть медленной.

д) Корни системы могут быть определены только приближенно с точностью ε.

Приложения.

**Исходный код**

**laba2.1.cpp**

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <iostream>

#include <time.h>

*using* *namespace* std;

*//Выделение* *памяти*

double\*\* **Init**(int rows,int cols){

double\*\*mas;

mas=(double\*\*)malloc(rows\**sizeof*(double\*));

*for*(int i=0;i<rows;i++){

mas[i]=(double \*)malloc(cols\**sizeof*(double));

}

*return* mas;

}

*//Ввод* *матриц*

double\*\* **Input**(int rows,int cols){

double\*\*mas;

mas=Init(rows,cols);

*for*(int i=0;i<rows;i++){

*for*(int j=0;j<cols;j++){

printf("mat[%d][%d]=",i+1,j+1);

scanf("%lf",&mas[i][j]);

}

}

*return* mas;

}

*//Вывод* *матриц*

void **Output**(double\*\* mas,int rows,int cols){

*for*(int i=0;i<rows;i++){

*for*(int j=0;j<cols;j++){

printf("%f ",mas[i][j]);

}

printf("\n");

}

}

*//Умножение*

double\*\***Multiply**(double\*\* a,double\*\* b,int rowsa,int colsa,int rowsb,int colsb){

*if* (colsa!=rowsb){

printf("Invalid matrix sizes\n");

*return* 0;

}

*else*

{

double\*\*c;

c=Init(rowsa,colsb);

*for*(int i=0;i<rowsa;i++){

*for*(int j=0;j<colsb;j++){

c[i][j]=0;

*for* (int k=0;k<colsa;k++){

c[i][j]=a[i][k]\*b[k][j];

}

}

}

printf("a\*b=\n");

Output(c,rowsa,colsb);

*return* 0;

}

}

*//Транспонирование*

double\*\* **Transpore**(double\*\* mas,int rows,int cols){

double\*\*x;

x=Init(cols,rows);

*for*(int i=0;i<rows;i++){

*for*(int j=0;j<cols;j++){

x[j][i]=mas[i][j];

}

}

*return* x;

}

*//Доп* *матрица* *составленая* *из* *миноров*

double\*\* **GetMatr**(double\*\* mas,int rows,int cols,int kr,int kc){

int di,dj;

double\*\*p=Init(rows-1,cols-1);

di=0;

*for*(int i=0;i<rows;i++){

*if* (i==kr) {di=1; *continue*;}

dj=0;

*for*(int j=0;j<cols;j++){

*if*(j==kc) {dj=1; *continue*;}

p[i-di][j-dj]=mas[i][j];

}

}

*return* p;

}

*//Определитель*

double **Det**(double\*\* mas,int size)

{

int k=1;

double d=0;

*if* (size<1) {printf("невозможно");

*return* 0;}

*if* (size==1) {*return* mas[0][0];}

*if* (size==2) { *return* (mas[0][0]\*mas[1][1]-mas[1][0]\*mas[0][1]);

}

*if* (size>2){

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

d=d+k\*mas[0][i]\*Det(GetMatr(mas,size,size,0,i),size-1);

k\*= -1;

}

}

*return* d;

}

*//Обратная* *матрица*

double\*\* **Mreverse**(double\*\* mas,int size){

double\*\*rez;

rez=Init(size,size);

double det=Det(mas,size);

rez=Transpore(mas,size,size);

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

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

rez[i][j]=rez[i][j]/det;

}

}

*return* rez;

}

*//Метод* *Крамера*

double\*\***Kramer**(double\*\* a,double\*\* b,int size){

double det=Det(a,size);

double q=0;

double\*\* aq=Init(size,size);

double\*\* x=Init(size,1);

*for*(int k=0;k<size;k++){

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

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

aq[i][j]=a[i][j];

}

}

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

aq[i][k]=b[i][0];

}

x[k][0]=Det(aq,size)/det;

}

*return* x;

}

int **main**(){

double\*\* a;

double\*\* b;

int rowsa,colsa;

int rowsb,colsb;

printf("rowsa=");

scanf("%d",&rowsa);

printf("\n");

printf("colsa=");

scanf("%d",&colsa);

printf("\n");

printf("rowsb=");

scanf("%d",&rowsb);

printf("\n");

printf("colsb=");

scanf("%d",&colsb);

printf("\n");

a=Input(rowsa,colsa);

printf("a=\n");

Output(a,rowsa,colsa);

b=Input(rowsb,colsb);

printf("b=\n");

Output(b,rowsb,colsb);

double\*\* g;

clock\_t tStart = clock();

g=Kramer(a,b,rowsa);

printf("x=");

Output(g,rowsa,1);

printf("Time taken: %.3fs\n", (double)(clock() - tStart)/CLOCKS\_PER\_SEC);

getchar();

*return* 0;

}

**laba2.2.cpp**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <time.h>

#define e 0.005

*using* *namespace* std;

*class* Matr

{

*private*:

int size;

double \*\*mas;

double \*mas1;

*public*:

Matr()

{

size = 0;

mas = NULL;

mas1 = NULL;

}

Matr(int l)

{

size = l;

mas = *new* double\*[l];

*for* (int i = 0; i < l; i++)

mas[i] = *new* double[l];

mas1 = *new* double[l];

}

void **Add**()

{

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

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

cin >> mas[i][j];

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

{

cin >> mas1[i];

}

}

void **Print**()

{

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

{

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

{

cout << setw(4) << mas[i][j] << " ";

}

cout << " " << mas1[i] << endl;

}

}

void **Preob**()

{

double temp = 0;

*for* (int k = 0; k < size; k++)

{

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

{

temp = mas[i][i]\*(-1);

mas1[i] /= temp;

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

{

mas[i][j] /= temp;

}

}

}

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

{

mas1[i] \*= -1;

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

mas[i][i] = 0;

}

}

double **Norma**(double \*\*mas)

{

double sum = 0, max = 0;

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

{

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

{

sum += fabs(mas[i][j]);

*if* (sum > max) max = sum;

}

sum = 0;

}

*return* max;

}

double **Pogr**()

{

double eps = 0;

double norm = Norma(mas);

eps = ((1 - norm) / norm)\*e;

*return* 0.001;

}

void **Itera**()

{

double \*x = *new* double[size];

double \*x0 = *new* double[size];

double \*E = *new* double[size];

double max = 0, per = 0;

per = Pogr();

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

x0[i] = mas1[i];

int counter = 0;

int k = 0;

*do*

{

k++;

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

{

x[i] = 0;

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

{

x[i] += mas[i][j] \* x0[j];

}

x[i] += mas1[i];

E[i] = fabs(x[i] - x0[i]);

}

max = 0;

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

{

*if* (max < E[i]) max = E[i];

x0[i] = x[i];

}

cout<<endl << "maxDiff " << k << " = " << max;

counter++;

} *while* (max > per);

cout << endl << "Number of iterations: " << counter << endl;

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

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

*delete*[] x;

*delete*[] x0;

*delete*[] E;

}

};

int **main**()

{

setlocale(LC\_ALL, "rus");

int n;

cout << "Введите количество уравнений: ";

cin >> n;

Matr a(n);

cout << "Введите матрицу коэффициентов, потом столбец свободных членов:" << endl;

a.Add();

cout << endl << "Расширенная матрица:" << endl;

a.Print();

a.Preob();

cout << endl << "Преображенная матрица" << endl;

a.Print();

cout << endl;

clock\_t tStart = clock();

a.Itera();

cout << endl;

printf("Time taken: %.2fs\n", (double)(clock() - tStart)/CLOCKS\_PER\_SEC);

system("pause");

*return* 0;

}