Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

**Отчет по дисциплине «Методы вычислений»**

студента 3 курса 341 группы  
направления 010500.62 Математическое обеспечение и администрирование информационных систем (профиль Параллельное программирование)

факультета компьютерных наук и информационных технологий

Акимова Артемия Андреевича

Преподаватель: Поплавский Д.В.

Саратов 2015

Оглавление

[1. Интерполяции. 4](#_Toc438126524)

[1.1 Метод кусочно-линейной интерполяции 4](#_Toc438126525)

[1.2 Интерполяционный многочлен в форме Лагранжа 6](#_Toc438126526)

[1.3 Интерполяционный многочлен в форме Ньютона 8](#_Toc438126527)

[1.4 Интерполяционный кубический сплайн 11](#_Toc438126528)

[2. СЛАУ 13](#_Toc438126529)

[2.1 Метод прогонки. 19](#_Toc438126530)

[2.2 Метод Гаусса 19](#_Toc438126531)

[2.3 Метод Зейделя 20](#_Toc438126532)

[3. Нелинейные алгебраические уравнения 20](#_Toc438126533)

[4. СНУ 23](#_Toc438126534)

[4.1 Метод простой итерации 23](#_Toc438126535)

[4.2 Метод Ньютона 26](#_Toc438126536)

# 1 Интерполяции.

## Метод кусочно-линейной интерполяции

Задание требуется вычислить значения функции кусочно-линейной интерполяции в заданных точках .

**Входные данные:**

2 4 6

2 6 2

**Выходные данные:**

2 4 6

2 6 2

2 3 4 5 6

2 4 6 4 2

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace \_17.\_09.chislmet

{

class Program

{

static void Main(string[] args)

{

using (StreamWriter outfile = new StreamWriter("output.txt"))

{

string file = File.ReadAllText("input.txt");

int[] indata = file.Split(new char[] { ' ', '\n' }, StringSplitOptions.RemoveEmptyEntries).Select(n => int.Parse(n)).ToArray();

int size = indata.Length / 2;

int k = size;

double[,] table = new double[2, size];

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

{

table[0, i] = indata[i];

}

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

{

table[1, j] = indata[k];

k++;

}

double[,] result = new double[2, size \* 2 - 1];

int k1 = 0;

for (int i = 0; i < size \* 2 - 1; i++)

{

if (i % 2 == 0)

{

result[0, i] = indata[k1];

k1++;

}

}

for (int i = 1; i < size \* 2 - 1; i++)

{

if (i % 2 != 0)

{

result[0, i] = (result[0, i - 1] + result[0, i + 1]) / 2;

}

}

int k2 = 0;

for (int i = 0; i < size \* 2 - 1; i++)

{

if (i % 2 == 0)

{

result[1, i] = table[1, k2];

k2++;

}

}

for (int i = 1; i < size \* 2 - 1; i++)

{

if (i % 2 != 0)

{

result[1, i] = (result[1, i - 1] + result[1, i + 1]) / 2;

}

}

for (int i = 0; i < table.GetLength(0); i++)

{

for (int j = 0; j < table.GetLength(1); j++)

{

if (table[i, j] < 0)

{

outfile.Write("{0} ", table[i, j]);

}

else outfile.Write("{0} ", table[i, j]);

}

outfile.WriteLine();

}

for (int i = 0; i < result.GetLength(0); i++)

{

for (int j = 0; j < result.GetLength(1); j++)

{

if (result[i, j] < 0)

{

outfile.Write("{0} ", result[i, j]);

}

else outfile.Write("{0} ", result[i, j]);

}

outfile.WriteLine();

}

}

}

}

}

## Интерполяционный многочлен в форме Лагранжа

Задание: Ваша задача — построить интерполяционный многочлен по формуле Лагранжа и вычислить его значения в заданных точках .

**Входные данные:**

2 5

0 0.5 1

0 1 0

0 0.2 0.4 0.6 0.8 1

**Выходные данные:**

0 0.64 0.96 0.96 0.64 0

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace Example

{

class Program

{

static void Main()

{

System.Threading.Thread.CurrentThread.CurrentCulture = new System.Globalization.CultureInfo("en-US");

using (StreamReader inFile = new StreamReader("input.txt"))

{

using (StreamWriter outFile = new StreamWriter("output.txt", false))

{

string S = inFile.ReadLine();

string[] nN = S.Split(' ');

int n = int.Parse(nN[0]);

int N = int.Parse(nN[1]);

string line = inFile.ReadLine();

string[] mas = line.Split(' ');

double[] x = new double[n + 1];

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

{

x[i] = double.Parse(mas[i]);

}

line = inFile.ReadLine();

mas = line.Split(' ');

double[] y = new double[n + 1];

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

{

y[i] = double.Parse(mas[i]);

}

line = inFile.ReadLine();

mas = line.Split(' ');

double[] s = new double[mas.Length];

for (int i = 0; i < mas.Length; i++)

{

s[i] = double.Parse(mas[i]);

}

for (int i = 0; i < mas.Length; i++)

{

outFile.Write(Sum(s[i], x, y) + " ");

}

}

}

}

static double LNK(double X, int k, double[] x)

{

double result = 1;

for (int i = 0; i < x.Length; i++)

{

if (i != k)

result \*= ((X - x[i]) / (x[k] - x[i]));

}

return result;

}

static double Sum(double X, double[] x, double[] y)

{

double result = 0;

for (int i = 0; i < x.Length; i++)

{

result += (y[i] \* LNK(X, i, x));

}

return result;

}

}

## Интерполяционный многочлен в форме Ньютона

Задание: Ваша задача — построить интерполяционный многочлен по формуле Ньютона и вычислить его значения в заданных точках .

Входные данные:

2 5

0 0.5 1

0 1 0

0 0.2 0.4 0.6 0.8 1

Выходные данные:

0 0.64 0.96 0.96 0.64 0

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Text;

**using** System.IO;

**namespace** Example

{

**class** Program

    {

**static** **void** Main()

        {

            System.Threading.Thread.CurrentThread.CurrentCulture = new System.Globalization.CultureInfo("en-US");

**using** (StreamReader inFile = new StreamReader("input.txt"))

            {

**using** (StreamWriter outFile = new StreamWriter("output.txt", **false**))

                {

**string** S = inFile.ReadLine();

**string**[] nN = S.Split(' ');

**int** n = **int**.Parse(nN[0]);

**int** N = **int**.Parse(nN[1]);

**string** line = inFile.ReadLine();

**string**[] mas = line.Split(' ');

**double**[] x = new **double**[n + 1];

**for** (**int** i = 0; i <= n; i++)

                    {

                        x[i] = **double**.Parse(mas[i]);

                    }

                    line = inFile.ReadLine();

                    mas = line.Split(' ');

**double**[] y = new **double**[n + 1];

**for** (**int** i = 0; i <= n; i++)

                    {

                        y[i] = **double**.Parse(mas[i]);

                    }

                    line = inFile.ReadLine();

                    mas = line.Split(' ');

**double**[] s = new **double**[mas.Length];

**for** (**int** i = 0; i < mas.Length; i++)

                    {

                        s[i] = **double**.Parse(mas[i]);

                    }

**for** (**int** i = 0; i < mas.Length; i++)

                    {

                        outFile.Write(NT(s[i], x, y) + " ");

                    }

                }

            }

        }

**static** **double** F(**int** l, **int** r, **double**[] x, **double**[] y)

        {

**if** (l == r)

            {

**return** y[l];

            }

**else**

            {

**return** (F(l, r - 1, x, y) - F(l + 1, r, x, y)) / (x[l] - x[r]);

            }

        }

**static** **double** NT(**double** Si, **double**[] x, **double**[] y)

        {

**double** result = 0;

**for** (**int** k = 0; k < x.Length; k++)

            {

**double** S = 1;

**for** (**int** i = 0; i < k; i++)

                    S \*= (Si - x[i]);

                result += F(0, k, x, y) \* S;

            }

**return** result;

        }

    }

}

## Интерполяционный кубический сплайн

Задание: требуется вычислить значения интерполяционного кубического сплайна  в заданных точках .

**Входные данные:**

2 4 0 2

0 3 1

0 0.6 1 1.1 2

**Выходные данные:**

0.000000 2.280000 3.000000 3.013750 1.000000

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

namespace Spline

{

class Program

{

static void Main()

{

System.Threading.Thread.CurrentThread.CurrentCulture = new System.Globalization.CultureInfo("en-US");

using (StreamReader inFile = new StreamReader("input.txt"))

{

using (StreamWriter outFile = new StreamWriter("output.txt", false))

{

///////////////////////////// СЧИТЫВАНИЕ ДАННЫХ

string S = inFile.ReadLine();

string[] nN = S.Split(' ');

int n = int.Parse(nN[0]);

int N = int.Parse(nN[1]);

double A = double.Parse(nN[2]);

double B = double.Parse(nN[3]);

string line = inFile.ReadLine();

string[] mas = line.Split(' ');

double[] y = new double[n + 1];

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

{

y[i] = double.Parse(mas[i]);

}

line = inFile.ReadLine();

mas = line.Split(' ');

double[] s = new double[N + 1];

for (int i = 0; i < N + 1; i++)

{

s[i] = double.Parse(mas[i]);

}

////////////////////////////////////////////////////

double[] Ci = new double[n + 1];

double[] Alph = new double[n + 1];

double[] Beta = new double[n + 1];

double h = (B - A) / n;

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

{

double Di = 6 / h / h \* (y[i - 1] - 2 \* y[i] + y[i + 1]);

Alph[i] = -1 / (Alph[i - 1] + 4);

Beta[i] = (Di - Beta[i - 1]) / (Alph[i - 1] + 4);

}

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

{

Ci[i] = Alph[i] \* Ci[i + 1] + Beta[i];

}

for (int j = 0; j < N + 1; j++)

{

if (s[j] == A)

{

outFile.Write("{0:0.000000} ", y[0]);

continue;

}

if (s[j] == B)

{

outFile.Write("{0:0.000000} ", y[n]);

continue;

}

int i = (int)Math.Ceiling((double)n \* (s[j] - A) / (B - A));

double Xi = A + h \* i;

double Ai = y[i];

double Di = (Ci[i] - Ci[i - 1]) / h;

double Bi = (y[i] - y[i - 1]) / h + h / 2 \* Ci[i] - (h \* h) / 6 \* Di;

outFile.Write("{0:0.000000} ", Ai + Bi \* (s[j] - Xi) + Ci[i] / 2 \* (s[j] - Xi) \* (s[j] - Xi) + Di / 6 \* (s[j] - Xi) \* (s[j] - Xi) \* (s[j] - Xi));

}

}

}

}

}

# СЛАУ

Задание: данной программе реализовать 3 метода решения систем линейных алгебраических уравнений (СЛАУ представлена в скриншотах): метод прогонки, Гауса и Зейделя.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Progonka

{

class Program

{

public double eps = 1e-7;

public double st = Math.Pow(10, -6);

public static int n = 5;

public double[] x = new double[n];

public double[] p = new double[n];

public double[] f = new double[n];

public double[,] matrix = new double[n, n];

public int step=0;

public bool converge(double[] xk, double[] xkp)

{

double norm = 0;

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

{

norm += (xk[i] - xkp[i]) \* (xk[i] - xkp[i]);

}

if (Math.Sqrt(norm) >= eps)

return false;

return true;

}

public bool Seidel()

{

Console.WriteLine();

Console.WriteLine("<------------------------------Seidel------------------------------>");

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

{

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

{

matrix[i, j] = 0;

}

}

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

{

matrix[i, i] = 1 + i \* 2;

if (i == 0)

{

matrix[i, i + 1] = matrix[i, i] \* st;

continue;

}

if (i == n - 1)

{

matrix[i, i - 1] = matrix[i, i] \* st;

continue;

}

matrix[i, i + 1] = matrix[i, i] \* st;

matrix[i, i - 1] = matrix[i, i] \* st;

}

double[] x = new double[n];

double[] f = new double[n];

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

{

f[i] = 2;

}

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

{

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

{

Console.Write("\t" + matrix[i, j]);

}

Console.Write("\t" + "f" + "[" + i + "] = " + f[i]);

Console.WriteLine();

}

do

{

step++;

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

p[i] = x[i];

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

{

double var = 0;

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

var += (matrix[i, j] \* x[j]);

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

var += (matrix[i, j] \* p[j]);

x[i] = (f[i] - var) / matrix[i, i];

}

}

while (!converge(x, p));

Console.WriteLine(step);

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

{

Console.WriteLine("x" + "[" + i + "]" + "=" + "\t" + x[i] + " ");

}

Console.WriteLine("\n" + "<------------------------------ProverkaSeidel------------------------------>");

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

{

f[i] = 0;

}

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

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

{

f[i] = f[i] + x[j] \* matrix[i, j];

}

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

{

Console.WriteLine("f" + "[" + i + "]" + "=" + "\t" + f[i] + " ");

}

return true;

}

public bool Gauss()

{

Console.WriteLine();

Console.WriteLine("<------------------------------Gauss------------------------------>");

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

{

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

{

matrix[i, j] = 0;

}

}

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

{

matrix[i, i] = 1 + i \* 2;

if (i == 0)

{

matrix[i, i + 1] = matrix[i, i] \* st;

continue;

}

if (i == n - 1)

{

matrix[i, i - 1] = matrix[i, i] \* st;

continue;

}

matrix[i, i + 1] = matrix[i, i] \* st;

matrix[i, i - 1] = matrix[i, i] \* st;

}

double[,] copymatrix = new double[n, n];

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

{

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

{

copymatrix[i, j] = matrix[i, j];

}

}

double[] x = new double[n];

double[] f = new double[n];

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

{

f[i] = 2;

}

double[] copyF = new double[n];

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

{

copyF[i] = f[i];

}

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

{

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

{

Console.Write("\t" + matrix[i, j]);

}

Console.Write("\t" + "f" + "[" + i + "] = " + f[i]);

Console.WriteLine();

}

int k = 0;

while (k < n)

{

double max = Math.Abs(matrix[k, k]);

int index = k;

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

{

if (Math.Abs(matrix[i, k]) > max)

{

max = Math.Abs(matrix[i, k]);

index = i;

}

}

if (max < eps)

return false;

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

{

double temple = matrix[k, j];

matrix[k, j] = matrix[index, j];

matrix[index, j] = temple;

}

double temp = f[k];

f[k] = f[index];

f[index] = temp;

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

{

double temple = matrix[i, k];

if (Math.Abs(temple) < eps) continue;

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

{

matrix[i, j] = matrix[i, j] / temple;

}

f[i] = f[i] / temple;

if (i == k) continue;

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

{

matrix[i, j] = matrix[i, j] - matrix[k, j];

}

f[i] = f[i] - f[k];

}

k++;

}

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

{

x[k] = f[k];

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

{

f[i] = f[i] - matrix[i, k] \* x[k];

}

}

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

{

Console.WriteLine("x" + "[" + i + "]" + "=" + "\t" + x[i] + " ");

}

Console.WriteLine("\n" + "<------------------------------ProverkaGauss------------------------------>");

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

{

copyF[i] = 0;

}

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

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

{

copyF[i] = copyF[i] + x[j] \* copymatrix[i, j];

}

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

{

Console.WriteLine("f" + "[" + i + "]" + "=" + "\t" + copyF[i] + " ");

}

return true;

}

public bool Progonka()

{

Console.WriteLine("<------------------------------Progonka------------------------------>");

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

{

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

{

matrix[i, j] = 0;

}

}

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

{

matrix[i, i] = 1 + i \* 2;

if (i == 0)

{

matrix[i, i + 1] = matrix[i, i] \* st;

continue;

}

if (i == n - 1)

{

matrix[i, i - 1] = matrix[i, i] \* st;

continue;

}

matrix[i, i + 1] = matrix[i, i] \* st;

matrix[i, i - 1] = matrix[i, i] \* st;

}

double[] x = new double[n];

double m;

double[] f = new double[n];

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

{

f[i] = 2;

}

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

{

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

{

Console.Write("\t" + matrix[i, j]);

}

Console.Write("\t" + "f" + "[" + i + "] = " + f[i]);

Console.WriteLine();

}

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

{

m = matrix[i, i + 1] / matrix[i, i];

matrix[i, i] = matrix[i, i] - m \* matrix[i - 1, i + 1];

f[i] = f[i] - m \* f[i - 1];

}

x[n - 1] = f[n - 1] / matrix[n - 1, n - 1];

for (int i = n - 2; i >= 0; i--)

x[i] = (f[i] - matrix[i + 1, i] \* x[i + 1]) / matrix[i, i];

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

{

Console.WriteLine("x" + "[" + i + "]" + "=" + "\t" + x[i] + " ");

}

Console.WriteLine("\n" + "<------------------------------ProverkaProgonka------------------------------>");

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

{

f[i] = 0;

}

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

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

{

f[i] = f[i] + x[j] \* matrix[i, j];

}

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

{

Console.WriteLine("f" + "[" + i + "]" + "=" + "\t" + f[i] + " ");

}

return true;

}

public void Multi()

{

//Console.WriteLine("Progonka ");

//Progonka();

//Gauss();

Seidel();

}

static void Main(string[] args)

{

Program res = new Program();

res.Multi();

}

}

}

## 2.1 Метод прогонки.C:\Users\Артем\Dropbox\Скриншоты\Скриншот 2015-12-13 16.23.26.png

## 2.2 Метод ГауссаC:\Users\Артем\Dropbox\Скриншоты\Скриншот 2015-12-13 16.24.12.png

## 2.3 Метод ЗейделяC:\Users\Артем\Dropbox\Скриншоты\Скриншот 2015-12-13 16.26.10.png

# 3. Нелинейные алгебраические уравнения

Задание: решить нелинейное уравнение

1. Методом деления отрезка пополам
2. Методом простой итерации
3. Методом Ньютона
4. Модифицированным методом Ньютона
5. Методом секущих.

Известное решение: 1,10714871789577

using System;

namespace AllMethods

{

class Program

{

const double EPS = 1e-9;

static double F(double x)

{

**return Math.Sin(x) - 2 \* Math.Cos(x);**

}

static double Psi(double x)

{

**return Math.Acos(Math.Sin(x) / 2);**

}

static double dF(double x)

{

**return Math.Cos(x) + 2 \* Math.Sin(x);**

}

#region Метод деления отрезка пополам

static double DevideByTwoMethod(double a, double b, out int counter)

{

counter = 0;

double c = 0.0;

while (Math.Abs(a - b) >= EPS)

{

c = a + (b - a) / 2;

if (F(a) \* F(c) < 0)

b = c;

else

a = c;

counter++;

}

return c;

}

#endregion

#region Метод приближений

static double SimpleIterationMethod(double x, out int counter)

{

counter = 0;

double xPrev = 0.0;

do

{

xPrev = x;

x = Psi(xPrev);

counter++;

}

while (Math.Abs(x - xPrev) >= EPS);

return x;

}

#endregion

#region Метод Ньютона

static double NewtonMethod(double x, out int counter)

{

double xPrev = 0.0;

counter = 0;

do

{

xPrev = x;

x = xPrev - F(xPrev) / dF(xPrev);

counter++;

}

while (Math.Abs(x - xPrev) >= EPS);

return x;

}

#endregion

#region Модифицированный метод Ньютона

static double ModifyNewtonMethod(double x, out int counter)

{

counter = 0;

double xPrev = 0.0;

double d = dF(x);

do

{

xPrev = x;

x = xPrev - F(xPrev) / d;

counter++;

}

while (Math.Abs(x - xPrev) >= EPS);

return x;

}

#endregion

#region Метод секущих

static double SecantMethod(double a, double b, out int counter)

{

counter = 0;

double temp = 0.0;

do

{

temp = b;

b = b - (F(b) \* (b - a)) / (F(b) - F(a));

a = temp;

counter++;

}

while (Math.Abs(b - a) >= EPS);

return b;

}

#endregion

static void Main(string[] args)

{

int counter;

double a = 0.0;

double b = Math.PI / 2;

Console.WriteLine("Метод деления отрезка пополам: {0}; {1} итераций", DevideByTwoMethod(a, b, out counter), counter);

Console.WriteLine("Метод итераций: {0}; {1} итераций", SimpleIterationMethod((a + b) / 2, out counter), counter);

Console.WriteLine("Метод Ньютона: {0}; {1} итераций", NewtonMethod(b, out counter), counter);

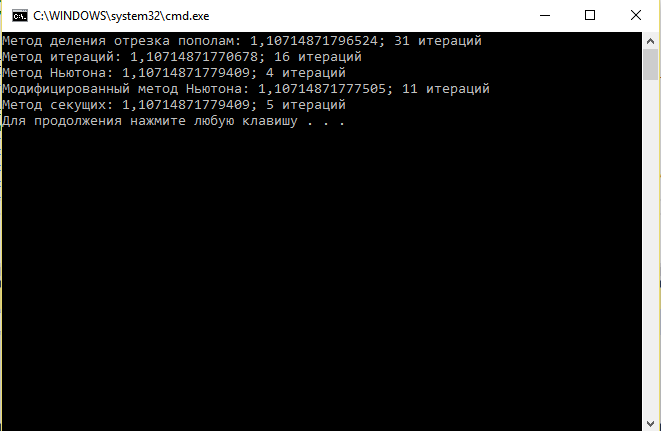
Console.WriteLine("Модифицированный метод Ньютона: {0}; {1} итераций", ModifyNewtonMethod(b, out counter), counter);

Console.WriteLine("Метод секущих: {0}; {1} итераций", SecantMethod(a, b, out counter), counter);

}

}

}



Результат работы программы.

# 4.СНУ

## 4.1 Метод простой итерации

Задание: решить систему нелинейных уравнений:

*,*

c точностью

Известное решение системы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace NewtonSnau

{

class Program

{

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

public static double fi1(double x1\_0, double x2\_0)

{

return 0.3-0.1\* x1\_0\* x1\_0- 0.2 \* x2\_0 \* x2\_0;

}

public static double fi2(double x1\_0, double x2\_0)

{

return 0.7- 0.2 \* x1\_0 \* x1\_0 +0.1\*x1\_0\*x2\_0;

}

static void Main(string[] args)

{

double q = 0.5;

double x1\_k = 0;

double x2\_k = 0;

double eps = Math.Pow(10, -6);

double x1\_0 = 0.25;

double x2\_0 = 0.75;

double x1\_simple = x1\_0;

double x2\_simple = x2\_0;

int step = 0;

Console.WriteLine("f1 = 0.1\*x1^2 + x1 + 0.2\*x2^2 - 0.3");

Console.WriteLine("f2 = 0.2\*x1^2 + x2 - 0.1\*x1\*x2 - 0.7");

while (q / (1 - q) \* Math.Max(Math.Abs(x1\_k - x1\_simple), Math.Abs(x2\_k - x2\_simple)) > eps)

{

step++;

x1\_simple = x1\_k;

x2\_simple = x2\_k;

x1\_k = fi1(x1\_0, x2\_0);

x2\_k = fi2(x1\_0, x2\_0);

x1\_0 = x1\_k;

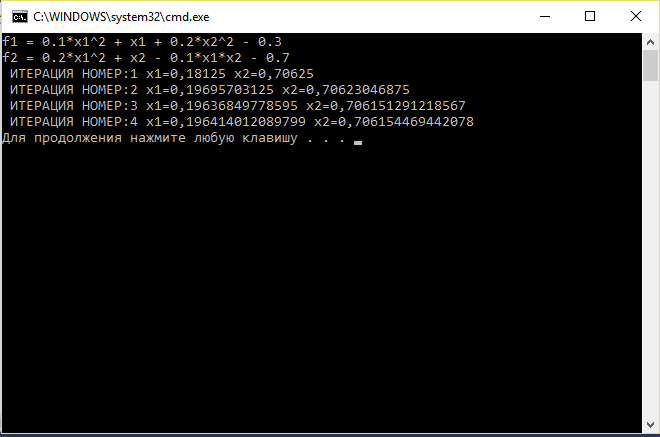
x2\_0 = x2\_k;

Console.WriteLine(" ИТЕРАЦИЯ НОМЕР:{2} x1={0} x2={1}", x1\_k, x2\_k, step);

}

}

}



Результат работы программы.

## 4.2 Метод Ньютона

Задание: решить систему нелинейных уравнений:

*,*

c точностью

Известное решение системы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace NewtonSnau

{

class Program

{

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

public static double f1(double x1\_k, double x2\_k)

{

return 0.1 \* x1\_k \* x1\_k + x1\_k + 0.2 \* x2\_k \* x2\_k - 0.3;

}

public static double f2(double x1\_k, double x2\_k)

{

return 0.2 \* x1\_k \* x1\_k + x2\_k - 0.1 \* x1\_k \* x2\_k - 0.7;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

public static double df1\_dx1(double x1\_k, double x2\_k)

{

return 0.2 \* x1\_k + 1;

}

public static double df1\_dx2(double x1\_k, double x2\_k)

{

return 0.4 \* x2\_k;

}

public static double df2\_dx1(double x1\_k, double x2\_k)

{

return 0.4 \* x1\_k - 0.1 \* x2\_k;

}

public static double df2\_dx2(double x1\_k, double x2\_k)

{

return 1 - 0.1 \* x1\_k;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

public static double J\_k(double x1\_k, double x2\_k)

{

return df1\_dx1(x1\_k, x2\_k) \* df2\_dx2(x1\_k, x2\_k) - df1\_dx2(x1\_k, x2\_k) \* df2\_dx2(x1\_k, x2\_k);

}

public static double A1\_k(double x1\_k, double x2\_k)

{

return f1(x1\_k, x2\_k) \* df2\_dx2(x1\_k, x2\_k) - f2(x1\_k, x2\_k) \* df2\_dx1(x1\_k, x2\_k);

}

public static double A2\_k(double x1\_k, double x2\_k)

{

return f2(x1\_k, x2\_k) \* df1\_dx1(x1\_k, x2\_k) - f1(x1\_k, x2\_k) \* df2\_dx1(x1\_k, x2\_k);

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

static void Main(string[] args)

{

double x1\_k = double.MaxValue;

double x2\_k = double.MaxValue;

double eps = Math.Pow(10, -4);

double x1\_0 = 0.25;

double x2\_0 = 0.75;

double x1\_simple = x1\_0;

double x2\_simple = x2\_0;

int step = 0;

Console.WriteLine("f1 = 0.1\*x1^2 + x1 + 0.2\*x2^2 - 0.3");

Console.WriteLine("f2 = 0.2\*x1^2 + x2 - 0.1\*x1\*x2 - 0.7");

while (Math.Max(Math.Abs(x1\_k - x1\_simple), Math.Abs(x2\_k - x2\_simple)) > eps)

{

step++;

x1\_simple = x1\_k;

x2\_simple = x2\_k;

x1\_k = x1\_0 - A1\_k(x1\_0, x2\_0) / J\_k(x1\_0, x2\_0);

x2\_k = x2\_0 - A2\_k(x1\_0, x2\_0) / J\_k(x1\_0, x2\_0);

x1\_0=x1\_k;

x2\_0=x2\_k;

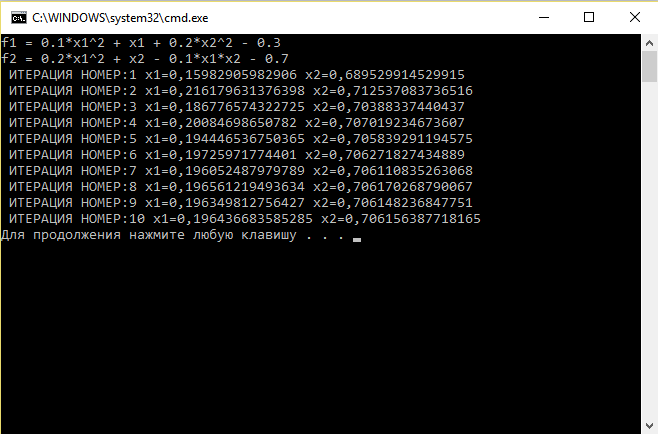
Console.WriteLine(" ИТЕРАЦИЯ НОМЕР:{2} x1={0} x2={1}", x1\_k,x2\_k,step);

}

}

}

}



Результат работы программы.