|  |  |
| --- | --- |
|  |  |
| 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 №3** | |
| **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 3

Теоретическая часть 3

Практическая часть 4

Приложения 6

# Задание

Решить систему нелинейных уравнений с точностью до 0.001 методами:

1) Ньютона(модифицированный)

2) Градиентный

## Теоретическая часть

Пусть дана система ***n***нелинейныхуравнений с ***n***неизвестными. Общий вид системы:

или где - нелинейные функции.

Требуется найти такой вектор который при подстановке в систему превращает каждое уравнение в верное числовое равенство.

Все функции непрерывны и дифференцируемы по всем своим неизвестным в некоторой выпуклой области существования неизвестной. Под выпуклой областью понимается такая область, в которой производная по каждой неизвестной не меняет свои знаки.

**Модифицированный метод Ньютона**

Формула для нахождения решения:

где -матрица Якоби в начальном приближении **x(0)**

причем а - обратная матрица Якоби в начальном приближении **x(0)** соответственно.

Матрица не изменяется от итерации к итерации.

Окончание итерационного процесса произойдет при достижении точности и при условии:

**Метод «градиентного спуска»**

Формула для нахождения решения: .

Где

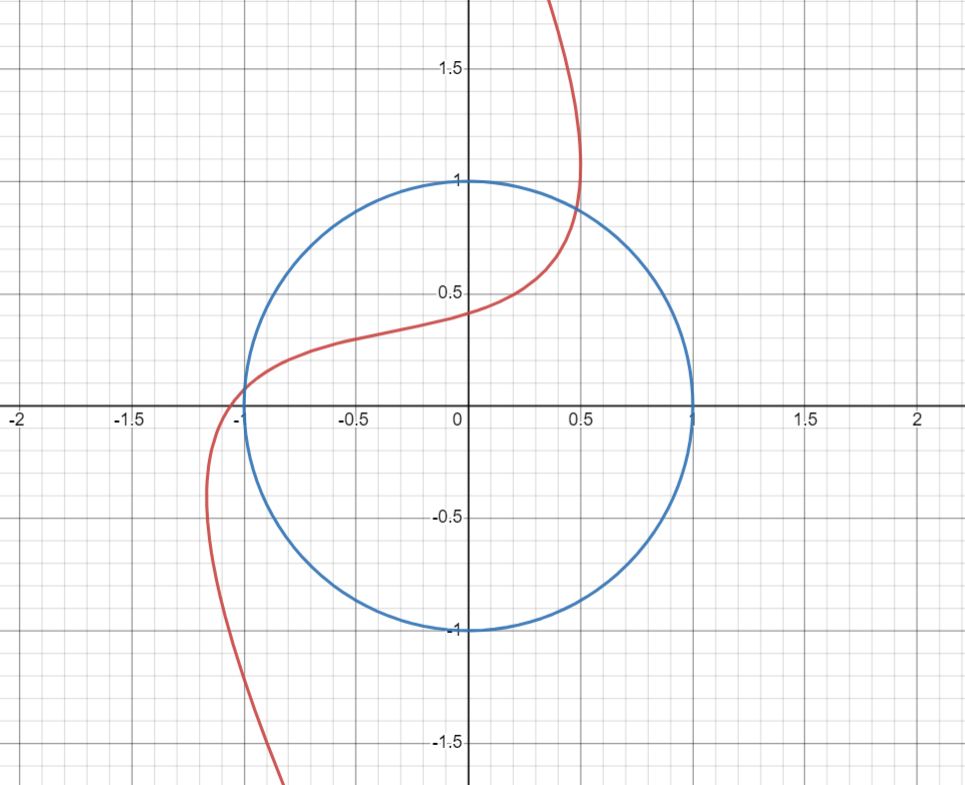
– транспонированная матрица Якоби, а = – вектор результатов функций в приближении .

**Практическая часть**

Для решения нелинейных систем уравнений методами «модифицированный методом Ньютона» и «метод градиентного спуска» были разработаны программы «lab3.1» и «lab3.2» соответственно. Программа написана на языке С++ в операционной системе «Windows» с использованием IDE «Qt Creator».

**1)** Решим систему

модифицированным методом Ньютона.

Графически уточним корни: 

Вычислим матрицу Якоби для нашей системы:

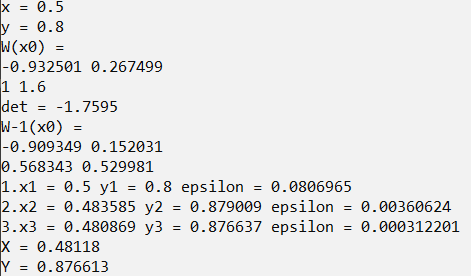
В качестве начального приближения возьмем:

Определитель отличен от нуля, значит существует обратная матрица.

Теперь запишем обратную матрицу Якоби в начальном приближении :

Организуем итерационный процесс:

Вывод программы:



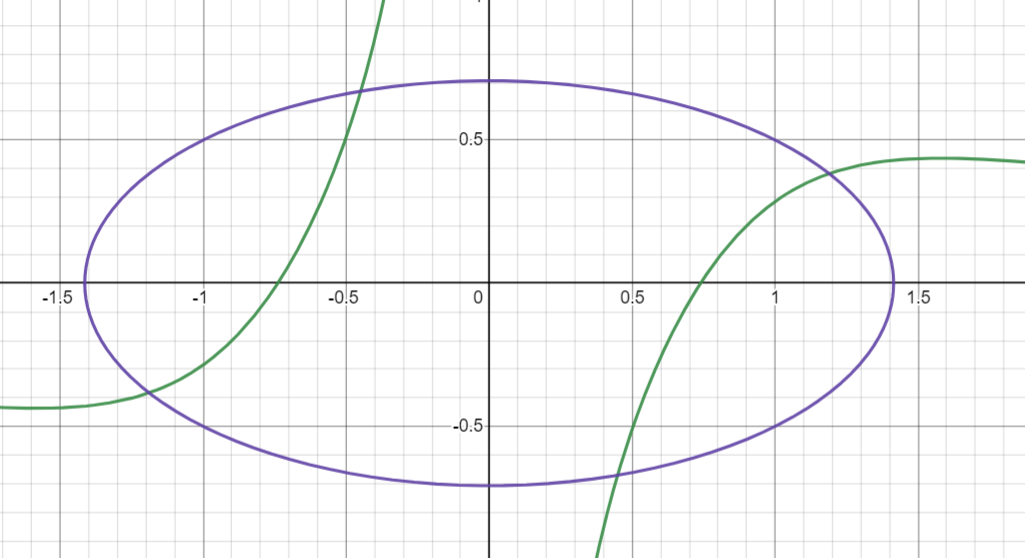
**Ответ:**

**2)** Решим систему

Методом градиентного спуска.

Вычислим матрицу Якоби для нашей системы:

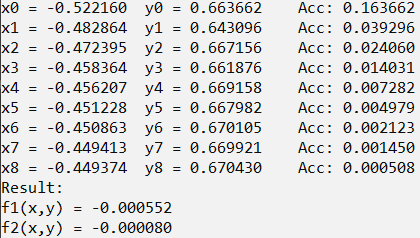
Графически уточним корни:



В качестве начального приближения возьмем:

Организуем итерационный процесс .

Вывод программы:



**Ответ:**

**Приложения**

*Код программы «lab3.1»*

#include <iostream>

#include <math.h>

*using* *namespace* std;

#define eps 0.001

#define x1 0.5

#define y1 0.8

double **function1**(double x, double y)

{

*return* sin(x + y) - 1.2\*x - 0.4;

}

double **function2**(double x, double y)

{

*return* x\*x + y\*y - 1;

}

double **func11**(double x, double y)

{

*return* cos(x + y) - 1.2;

}

double **func12**(double x, double y)

{

*return* cos(x + y);

}

double **func21**(double x, double y)

{

*return* 2\*x;

}

double **func22**(double x, double y)

{

*return* 2\*y;

}

void **ober\_matr**(double a[2][2])

{

double det, aa;

det = a[0][0]\*a[1][1] - a[0][1]\*a[1][0];

std::cout << "det = " << det << std::endl;

aa = a[0][0];

a[0][0] = a[1][1]/det;

a[1][1] = aa/det;

aa = a[0][1];

a[0][1] = -a[0][1]/det;

a[1][0] = -a[1][0]/det;

}

void **nuton**(double x, double y)

{

int i = 1;

double a[2][2], dx, dy, b[2], norm;

a[0][0] = func11(x1, y1);

a[0][1] = func12(x1, y1);

a[1][0] = func21(x1, y1);

a[1][1] = func22(x1, y1);

std::cout << "W(x0) = " << std::endl;

*for*(int i = 0; i < 2; i++)

{

*for*(int j = 0; j < 2; j++)

{

std::cout << a[i][j] << " ";

}

std::cout << std::endl;

}

ober\_matr(a);

std::cout << "W-1(x0) = " << std::endl;

*for*(int i = 0; i < 2; i++)

{

*for*(int j = 0; j < 2; j++)

{

std::cout << a[i][j] << " ";

}

std::cout << std::endl;

}

*do*

{

std::cout << i << ".x" << i << " = " << x << " y" << i << " = " << y;

dx = -a[0][0]\*function1(x, y) + -a[0][1]\*function2(x, y);

dy = -a[1][0]\*function1(x, y) + -a[1][1]\*function2(x, y);

x = x + dx;

y = y + dy;

b[0] = function1(x, y);

b[1] = function2(x, y);

norm = sqrt(dx\*dx+dy\*dy);

i++;

std::cout <<" epsilon = " << norm << std::endl;

}

*while* (norm >= eps);

cout << "X = " << x << endl << "Y = " << y << endl;

}

int **main**()

{

double x, y;

cout << "x = ";

cin >> x ;

cout << "y = ";

cin >> y;

nuton(x, y);

cout << endl;

system("PAUSE");

*return* 0;

}

*Код программы «lab3.2»*

#include <iostream>

#include <cmath>

#include <math.h>

#include <stdlib.h>

#include <iomanip>

*using* *namespace* std;

float res[2] = {-0.5, 0.5};

float resn[2];

float m = 0;

float eps = 0.001;

float **f1**(float x, float y)

{

float res;

res = tan(x\*y + 0.5)-x\*x;

*return* res;

}

float **f2**(float x, float y)

{

float res;

res = 0.5\*x\*x+2\*y\*y-1;;

*return* res;

}

float **f1dx**(float x, float y)

{

float res;

res = y/(cos(x\*y+0.5)\*cos(x\*y+0.5)) - 2\*x;

*return* res;

}

float **f1dy**(float x, float y)

{

float res;

res = x/(cos(x\*y+0.5)\*cos(x\*y+0.5));

*return* res;

}

float **f2dx**(float x, float y)

{

float res;

res = 0.5\*x;

*return* res;

}

float **f2dy**(float x, float y)

{

float res;

res = 4\*y;

*return* res;

}

void **count\_m**(float x, float y)

{

float wf1;

float wf2;

wf1 = (f1dx(x,y)\*f1dx(x,y)+f1dy(x,y)\*f1dy(x,y))\*f1(x,y)+

(f1dx(x,y)\*f2dx(x,y)+f1dy(x,y)\*f2dy(x,y))\*f2(x,y);

wf2 = (f1dx(x,y)\*f2dx(x,y)+f1dy(x,y)\*f2dy(x,y))\*f1(x,y) +

(f2dx(x,y)\*f2dx(x,y)+f2dy(x,y)\*f2dy(x,y))\*f2(x,y);

m = (f1(x,y)\*wf1+f2(x,y)\*wf2)/(wf1\*wf1+wf2\*wf2);

}

int **main**()

{

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

{

count\_m(res[0], res[1]);

resn[0] = res[0]-m\*(f1dx(res[0],res[1])\*f1(res[0],res[1])+f2dx(res[0],res[1])\*f2(res[0],res[1]));

resn[1] = res[1]-m\*(f1dy(res[0],res[1])\*f1(res[0],res[1])+f2dy(res[0],res[1])\*f2(res[0],res[1]));

float max\_eps = (fabs(res[0]-resn[0])>fabs(res[1]-resn[1]))?fabs(res[0]-resn[0]):fabs(res[1]-resn[1]);

res[0]=resn[0];

res[1]=resn[1];

printf("x%d = %f\ty%d = %f\t Acc: %f\n", i, res[0], i, res[1], max\_eps);