1. **Завдання**

α = 0.259 ; β= -1.064.

a = 0.113 ; b = 0.601 ; c = -1.929 ; d = 0.668.

**4.1.** Методом простої ітерації із точністю (норма вектора нев’язки) і за умови відмінності значень норми вектора-розв’язку на сусідніх ітераціях на те саме значення, знайти приблизний розв’язок системи рівнянь:

**4.2.** Методом Ньютона із точністю (норма вектора нев’язки) і за умови відмінності значень норми вектора-розв’язку на сусідніх ітераціях на те саме значення, знайти приблизний розв’язок системи рівнянь:

1. **Математичний розв’язок**

Для (4.1) очевидним є явне вираження з заданої системи вектора (х, у)

Таким чином, маємо формулу для ітераційного процесу . В даному випадку вектор-функція очевидно, буде стискаючим відображенням.

Графічно (за допомогою прикладного ПЗ) встановлюємо, що система матиме один корінь, до якого збігатиметься від довільної точки на площині:



Для (4.2), згідно з методом Ньютона, обчислимо необхідні елементи для побудови ітераційного процесу

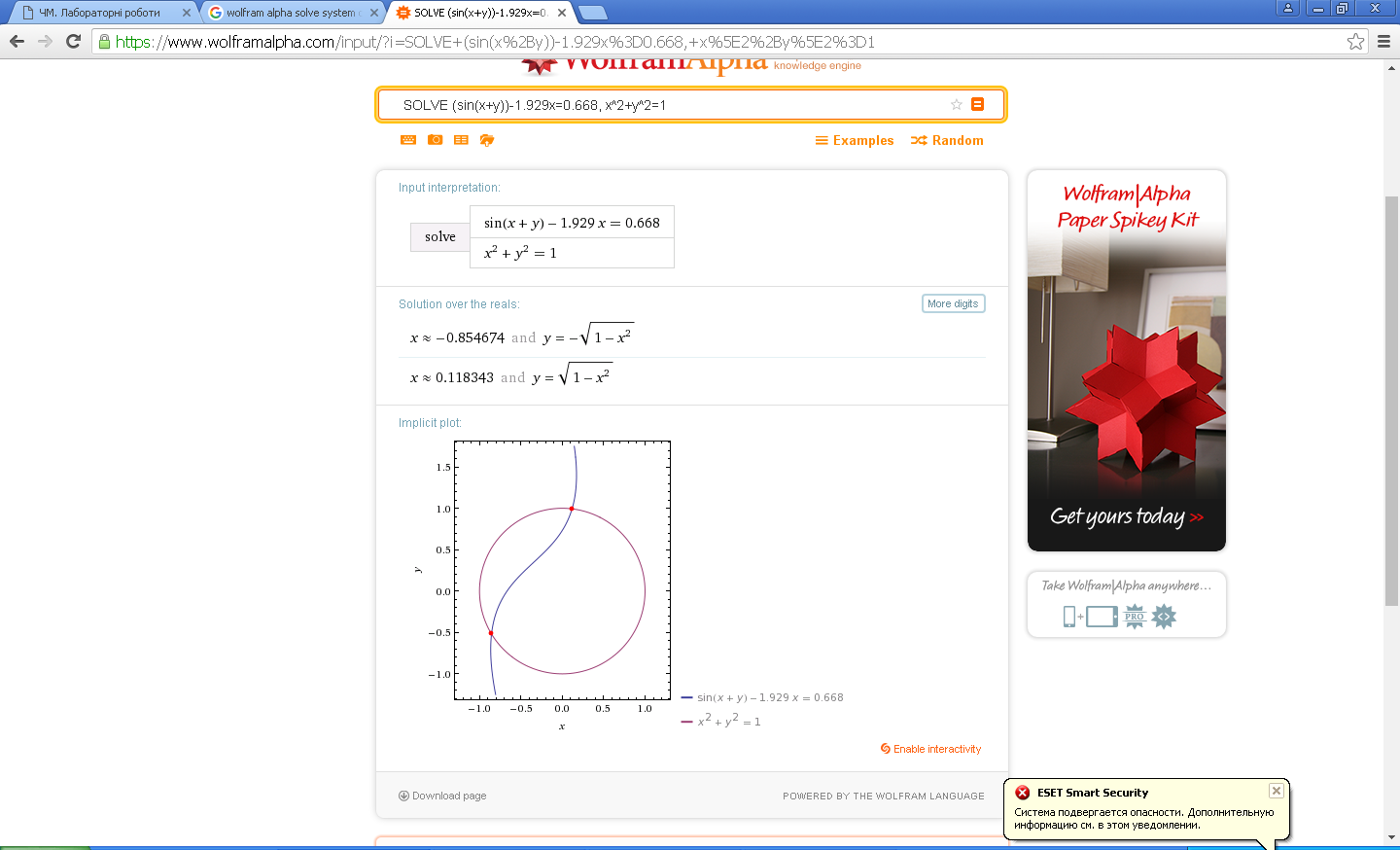
Вектор-функція – по суті, задана лівою частиною умови:

Для неї знаходимо Якобіан:

Обернена матриця матиме вигляд:

Враховуючи , маємо готові формули для обох координат в ітераційному процесі:

Графічно встановлюємо, що система матиме два корені:



1. **Лістинг**

#include <iostream>

#include <cmath>

#define eps 0.00001

using namespace std;

double fx(double x, double y) {

return -sin(y-1.064)+0.668;

}

double fy(double x, double y) {

return (-cos(x+0.259)-1.929)/(0.601);

}

double f2(double x, double y) {

return sin(x+y)-1.929\*x-0.668;

}

double dxf2(double x, double y) {

return cos(x+y)-1.929;

}

double dyf2(double x, double y) {

return cos(x+y);

}

double g2(double x, double y) {

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

}

double dxg2(double x, double y) {

return 2\*x;

}

double dyg2(double x, double y) {

return 2\*y;

}

double det(double \*\*matr) {

return matr[0][0]\*matr[1][1]-matr[0][1]\*matr[1][0];

}

double simple\_iteration(double &x, double &y) {

int i=0;

double a, b, norm1;

do {

a=x; b=y;

x=fx(x, y);

y=fy(x, y);

norm1=sqrt((x-a)\*(x-a)+(y-b)\*(y-b));

i++;

if(i>10000) return 0;

} while(norm1>eps);

return 1;

}

double newton(double &x, double &y) {

int i=0;

double a, b, norm1, norm2;

double \*\*matrd=new double\*[2];

for(i=0; i<2; i++) matrd[i]=new double[2];

double detf;

do {

a=x; b=y;

matrd[0][0]=dxf2(x,y);

matrd[0][1]=dyf2(x,y);

matrd[1][0]=dxg2(x,y);

matrd[1][1]=dyg2(x,y);

detf=det(matrd);

if (detf==0) return 0; //check det for zero

//formulas for x & y

x=x-(2\*y\*f2(x,y)-cos(x+y)\*g2(x,y))/detf;

y=y-(-2\*x\*f2(x,y)+(cos(x+y)-1.929)\*g2(x,y))/detf;

//residual norm calc

norm1=sqrt((x-a)\*(x-a)+(y-b)\*(y-b));

norm2=sqrt(f2(x,y)\*f2(x,y)+g2(x,y)\*g2(x,y));

i++;

if(i>10000) return 0; //check for num of iter

} while((norm1>eps) and (norm2>eps));

return 1;

}

int main() {

double x1, y1, x2, y2;

int k;

cout << "Simple iteration method\n";

cout << "\nOK interval is:\n x in (0.9,1.1) and y in (-1.2, -1)\n";

cout << "\nInput 1st iteration point (x,y): \n" << "Enter x: ";

do {

k=scanf("%lf", &x1);

fflush(stdin);

if (k!=1) cout << "Error! Input again: ";

} while(k!=1);

cout<<"Input y: ";

do {

k=scanf("%lf", &y1);

fflush(stdin);

if (k!=1) cout << "Error! Input again: ";

} while(k!=1);

if(simple\_iteration(x1, y1)==1) cout << "\nRoot: ("<<x1<<","<<y1<<")\n";

else cout<<"\nError!";

cout<<"\nNewton method\n";

cout<<"\nOK intervals are:\n x in (-0.9,-0.8) and y in (0.5, 0.6)\n or\n x in (-0.7, -0.9) and y in (-0.3, -0.2)\n";

cout<<"\nInput 1st iteration point (x,y): \n" << "Input x: ";

do {

k = scanf("%lf", &x2);

fflush(stdin);

if (k != 1) cout << "Error! Input again: ";

} while(k != 1);

cout << "Input y: ";

do {

k=scanf("%lf", &y2);

fflush(stdin);

if (k!=1) cout<<"Error! Input again: ";

} while(k!=1);

if(newton(x2, y2)== 1) cout<<"\nRoot1: ("<<x2<<","<<y2<<")\n";

else cout<<"\nError!";

cout<<"\nInput 1st iteration point (x,y): \n" << "Enter x: ";

do {

k = scanf("%lf", &x2);

fflush(stdin);

if (k != 1) cout << "Error! Input again: ";

} while(k != 1);

cout << "Input y: ";

do {

k=scanf("%lf", &y2);

fflush(stdin);

if (k!=1) cout<<"Error! Input again: ";

} while(k!=1);

if(newton(x2, y2)== 1) cout<<"\nRoot1: ("<<x2<<","<<y2<<")\n";

else cout<<"\nError!";

}

1. **Результати роботи програми**

