**МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ**

**НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС**

**«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»**

**НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

**КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ**

**Лабораторна робота №4**

**з курсу «Чисельні методи»**

**тема: «МЕТОДИ РОЗВ'ЯЗАННЯ СИСТЕМ НЕЛІНІЙНИХ РІВНЯНЬ»**

**Виконав: студент 2 курсу**

**групи КА-23**

**Деундяк О.В.**

**Прийняв: Коновалюк М. М.**

**Київ – 2014р.**

**Варіант 9**

Система рівнянь (1):

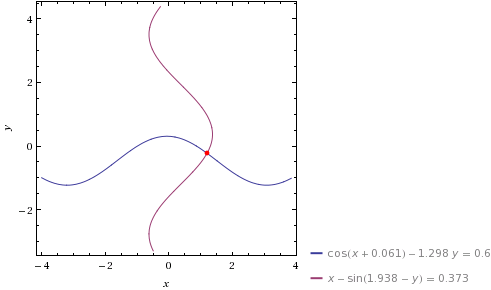
Система рівнянь (2):

**Умова:** розв’язати систему рівнянь (1) методом простих ітерацій та систему (2) методом Н’ютона.

**Допрограмовий етап**

Система рівнянь (1) еквівалентна:

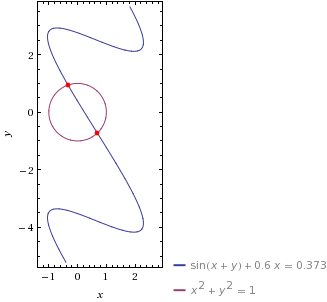
Тоді:



З графіку знайдемо початкове наближення:

Система рівнянь (2) еквівалентна:

Тоді:



З графіку знайдемо початкове наближення для 1 і 2 коренів:

**Текст програми:**

**Matrix.h**

#pragma once

#include <vector>

#include <string>

#include <fstream>

using std::vector;

using std::string;

using std::ifstream;

typedef double Item;

class Matrix

{

vector <vector <Item>> A;

size\_t n;

public:

Matrix(const vector<vector<Item>> &v);

Matrix(size\_t n, Item fill);

Matrix Inverse2() const;

Matrix operator\*(const Matrix &other) const;

vector<Item> operator\*(const vector<Item> &vec) const;

};

**Matrix.cpp**

#include "Matrix.h"

#include <iostream>

#include <iomanip>

#include <fstream>

#include <exception>

#include <algorithm>

const double eps = 1E-5;

const long MAXIT = 1000;

using std::cin;

//using std::cout;

using std::endl;

using std::ifstream;

extern std::ofstream cout;

Matrix::Matrix(const vector<vector<Item>> &v) :

n(v.size())

{

for (size\_t i = 0; i < n; i++)

{

vector<Item> T;

for (size\_t j = 0; j < n; j++)

T.push\_back(v[i][j]);

A.push\_back(T);

}

}

Matrix::Matrix(size\_t n\_, Item fill) : n(n\_)

{

A.resize(n);

for (size\_t i = 0; i < n; i++)

A[i].push\_back(fill);

}

Matrix Matrix::operator\*(const Matrix &other) const

{

Matrix P(n, 0);

for (size\_t row = 0; row < n; row++)

for (size\_t col = 0; col < n; col++)

for (size\_t inner = 0; inner < n; inner++)

P.A[row][col] += A[row][inner] \* other.A[inner][col];

return P;

}

Matrix Matrix::Inverse2() const

{

if (n != 2)

throw std::logic\_error("n!=2");

Matrix m\_inv(n, 0);

Item det = A[0][0] \* A[1][1] - A[0][1] \* A[1][0];

vector<Item> b, r;

b.resize(n);

r.resize(2);

r[0] = A[1][1] / det;

r[1] = -A[1][0] / det;

m\_inv.A[0] = r;

r[0] = -A[0][1] / det;

r[1] = A[0][0] / det;

m\_inv.A[1] = r;

return m\_inv;

}

vector<Item> Matrix::operator\*(const vector<Item> &vec) const

{

vector<Item> res;

for (size\_t i = 0; i < n; i++)

{

Item r = 0;

for (size\_t j = 0; j < n; j++)

r += A[i][j] \* vec[j];

res.push\_back(r);

}

return res;

}

Item operator\*(const vector<Item> &v1, const vector<Item> &v2)

{

size\_t l = std::min(v1.size(), v2.size());

Item r = 0;

for (size\_t i = 0; i < l; i++)

r += v1[i] \* v2[i];

return r;

}

**SONE.h**

#pragma once

#include <vector>

using std::vector;

typedef double Item;

class SONE

{

const size\_t n;

vector<Item> F(const vector<Item> &arg) const;

vector<Item> f(const vector<Item> &arg) const;

vector<Item> F2(const vector<Item> &arg) const;

vector<Item> f2(const vector<Item> &arg) const;

void ShowV(const vector<double> &v) const;

Item Norm(const vector<Item> &v) const;

vector<Item> R\_;

SONE() : n(0) {};

public:

SONE(Item x0, Item y0) : n(2)

{

R\_.push\_back(x0);

R\_.push\_back(y0);

}

vector<Item> SolveSimple();

vector<Item> SolveNewton();

};

**SONE.cpp**

#include "SONE.h"

#include <fstream>

#include <algorithm>

#include "Matrix.h"

const double eps = 1E-5;

const long MAXIT = 1000;

extern std::ofstream cout;

using std::endl;

Item SONE::Norm(const vector<Item> &v) const

{

Item r = Item(0);

for (size\_t i = 0; i < v.size(); i++)

r += v[i] \* v[i];

return sqrt(r);

}

void SONE::ShowV(const vector<double> &v) const

{

cout << "{";

for (size\_t i = 0; i < v.size(); i++)

cout << v[i] << " ";

cout << "}" << endl;

}

vector<Item> SONE::F(const vector<Item> &arg) const

{

vector<Item> r;

r.push\_back(0.373 - sin(arg[1] - 1.938));

r.push\_back((-0.6 + cos(arg[0] + 0.061))/1.298);

return r;

}

vector<Item> operator-(const vector<Item> &v1, const vector<Item> &v2)

{

size\_t l = std::min(v1.size(), v2.size());

vector<Item> r;

r.resize(l);

for (size\_t i = 0; i < l; i++)

r[i] = v1[i] - v2[i];

return r;

}

vector<Item> SONE::f(const vector<Item> &arg) const

{

vector<Item> r;

r.push\_back(-0.6 + cos(arg[0] + 0.061) - 1.298\*arg[1]);

r.push\_back(-0.373 + sin(arg[1] - 1.938) + arg[0]);

return r;

}

vector<Item> SONE::F2(const vector<Item> &arg) const

{

vector<vector<Item>> Fder = { { cos(arg[0] + arg[1]) + 0.6, cos(arg[0] + arg[1]) }, { 2 \* arg[0], 2 \* arg[1] } };

Matrix Fd(Fder);

Fd = Fd.Inverse2();

vector<Item> r1 = Fd\*f2(arg);

vector<Item> r = arg - r1;

return r;

}

vector<Item> SONE::f2(const vector<Item> &arg) const

{

vector<Item> r;

r.push\_back(-0.373 + sin(arg[0] + arg[1]) + 0.6\*arg[0]);

r.push\_back(arg[0] \* arg[0] + arg[1] \* arg[1] - 1);

return r;

}

vector<Item> SONE::SolveSimple()

{

vector<Item> R = R\_, residual = f(R), R0 = { 0, 0 };

Item residualNorm = Norm(residual);

long it = 1;

while (residualNorm > eps || Norm(R - R0) > eps)

{

cout << "Iteration " << it << endl << "(x,y)=";

ShowV(R);

cout << "Residual vector=";

ShowV(residual);

cout << "Residual vector's norm="

<< residualNorm << endl << endl;

R0 = R;

R = F(R);

residual = f(R);

residualNorm = Norm(residual);

++it;

if (it > MAXIT)

throw std::exception("Too many iterations");

}

cout << "Result in " << it << " iterations:" << endl << "(x,y)=";

ShowV(R);

cout << "Residual vector=";

ShowV(residual);

cout << "Residual vector's norm="

<< residualNorm << endl;

return R;

}

vector<Item> SONE::SolveNewton()

{

vector<Item> R = R\_, residual = f2(R), R0 = { 0, 0 };

Item residualNorm = Norm(residual);

long it = 1;

while (residualNorm > eps || Norm(R - R0) > eps)

{

cout << "Iteration " << it << endl << "(x,y)=";

ShowV(R);

cout << "Residual vector=";

ShowV(residual);

cout << "Residual vector's norm="

<< residualNorm << endl << endl;

R0 = R;

R = F2(R);

residual = f2(R);

residualNorm = Norm(residual);

++it;

if (it > MAXIT)

throw std::exception("Too many iterations");

}

cout << "Result in " << it << " iterations:" << endl << "(x,y)=";

ShowV(R);

cout << "Residual vector=";

ShowV(residual);

cout << "Residual vector's norm="

<< residualNorm << endl;

return R;

}

**main.cpp**

#include "SONE.h"

#include <fstream>

#include <iostream>

std::ofstream cout("output.txt");

int main()

{

SONE S(1.2, 0), S1(0, 1), S2(1,-0.7);

try

{

S2.SolveNewton();

}

catch (std::exception &e)

{

std::cout << e.what() << std::endl;

}

std::cin.get();

}

**Результати роботи програми**

**Для системи (1):**

Iteration 1

(x,y)={1.2 0 }

Residual vector={-0.295135 -0.106335 }

Residual vector's norm=0.313707

Iteration 2

(x,y)={1.30633 -0.227377 }

Residual vector={-0.102804 0.104951 }

Residual vector's norm=0.146913

Iteration 3

(x,y)={1.20138 -0.306579 }

Residual vector={0.101486 0.0469163 }

Residual vector's norm=0.111806

Iteration 4

(x,y)={1.15447 -0.228392 }

Residual vector={0.0443523 -0.0463472 }

Residual vector's norm=0.0641497

Iteration 5

(x,y)={1.20081 -0.194223 }

Residual vector={-0.04381 -0.0186823 }

Residual vector's norm=0.0476271

…

Iteration 20

(x,y)={1.20094 -0.228097 }

Residual vector={3.59286e-005 -3.72442e-005 }

Residual vector's norm=5.17493e-005

Iteration 21

(x,y)={1.20098 -0.228069 }

Residual vector={-3.54821e-005 -1.55214e-005 }

Residual vector's norm=3.87284e-005

Iteration 22

(x,y)={1.201 -0.228096 }

Residual vector={-1.47872e-005 1.53285e-005 }

Residual vector's norm=2.12985e-005

Iteration 23

(x,y)={1.20098 -0.228108 }

Residual vector={1.46034e-005 6.38838e-006 }

Residual vector's norm=1.59396e-005

Iteration 24

(x,y)={1.20097 -0.228096 }

Residual vector={6.08616e-006 -6.30897e-006 }

Residual vector's norm=8.7661e-006

Result in 25 iterations:

(x,y)={1.20098 -0.228092 }

Residual vector={-6.01052e-006 -2.62931e-006 }

Residual vector's norm=6.56046e-006

**Для системи (2) для точки {0,1}:**

Iteration 1

(x,y)={0 1 }

Residual vector={0.468471 0 }

Residual vector's norm=0.468471

Iteration 2

(x,y)={-0.410831 1.11099 }

Residual vector={0.0248385 0.403072 }

Residual vector's norm=0.403837

Iteration 3

(x,y)={-0.516376 0.965909 }

Residual vector={-0.248281 0.199625 }

Residual vector's norm=0.318581

Iteration 4

(x,y)={-0.444958 0.829277 }

Residual vector={-0.265047 -0.114311 }

Residual vector's norm=0.288646

Iteration 5

(x,y)={-0.28374 0.808086 }

Residual vector={-0.0425968 -0.266488 }

Residual vector's norm=0.269871

…

Iteration 50

(x,y)={-0.332976 0.942928 }

Residual vector={4.23273e-005 -1.39312e-005 }

Residual vector's norm=4.4561e-005

Iteration 51

(x,y)={-0.332998 0.942945 }

Residual vector={2.51075e-005 3.25168e-005 }

Residual vector's norm=4.1082e-005

Iteration 52

(x,y)={-0.333019 0.942937 }

Residual vector={-1.17922e-005 3.1807e-005 }

Residual vector's norm=3.39226e-005

Iteration 53

(x,y)={-0.333019 0.94292 }

Residual vector={-2.52695e-005 -4.85461e-007 }

Residual vector's norm=2.52742e-005

Iteration 54

(x,y)={-0.333004 0.942914 }

Residual vector={-9.23003e-006 -2.21145e-005 }

Residual vector's norm=2.39634e-005

Iteration 55

(x,y)={-0.332994 0.942921 }

Residual vector={1.09841e-005 -1.48119e-005 }

Residual vector's norm=1.84403e-005

Iteration 56

(x,y)={-0.332998 0.94293 }

Residual vector={1.38426e-005 5.00106e-006 }

Residual vector's norm=1.47183e-005

Iteration 57

(x,y)={-0.333007 0.942932 }

Residual vector={1.95739e-006 1.35679e-005 }

Residual vector's norm=1.37084e-005

Iteration 58

(x,y)={-0.333011 0.942926 }

Residual vector={-8.13906e-006 5.86667e-006 }

Residual vector's norm=1.0033e-005

Result in 59 iterations:

(x,y)={-0.333007 0.942922 }

Residual vector={-6.91422e-006 -5.27385e-006 }

Residual vector's norm=8.69598e-006

**Для системи (2) для точки {1,-0.7}:**

Iteration 1

(x,y)={1 -0.7 }

Residual vector={0.52252 0.49 }

Residual vector's norm=0.716329

Iteration 2

(x,y)={0.581343 -0.635684 }

Residual vector={-0.0785079 -0.257945 }

Residual vector's norm=0.269628

Iteration 3

(x,y)={0.706519 -0.74026 }

Residual vector={0.0171775 0.0471543 }

Residual vector's norm=0.0501857

Iteration 4

(x,y)={0.682166 -0.72485 }

Residual vector={-0.00637213 -0.00924234 }

Residual vector's norm=0.0112261

Iteration 5

(x,y)={0.6881 -0.727135 }

Residual vector={0.000834978 0.00220804 }

Residual vector's norm=0.00236064

Iteration 6

(x,y)={0.686951 -0.726407 }

Residual vector={-0.00027469 -0.000431348 }

Residual vector's norm=0.000511386

Iteration 7

(x,y)={0.687219 -0.726519 }

Residual vector={4.21035e-005 0.000100613 }

Residual vector's norm=0.000109067

Iteration 8

(x,y)={0.687165 -0.726487 }

Residual vector={-1.20599e-005 -2.02391e-005 }

Residual vector's norm=2.35598e-005

Iteration 9

(x,y)={0.687178 -0.726492 }

Residual vector={2.06139e-006 4.60037e-006 }

Residual vector's norm=5.04111e-006

Result in 10 iterations:

(x,y)={0.687175 -0.726491 }

Residual vector={-5.37509e-007 -9.45458e-007 }

Residual vector's norm=1.08757e-006

**Висновки:**

Метод простих ітерацій та Н’ютона дають нам змогу знаходити розв’язки систем нелінійних рівнянь, щоправда потребують використання графічного методу для початкової локалізації розв’язку. Також реалізації цих алгоритмів є складною для систем з великою кількістю рівнянь.