Московский авиационный институт

(Национальный исследовательский университет)

Лабораторная работа №2

По курсу «Численные методы»

Тема « Решение нелинейных уравнений и систем нелинейных уравнений»

Вариант №5

Выполнил: Цой В.Л.

Группа: 08-301

Проверил: Черкасов М.А.

Оценка:

Москва 2012

**Введение.**

Все программы реализованы на языке С++ . Для реализации лабораторной работы использовалась кросс-платформенная среда разработки **Qt**.

Во всех частях этой лабораторной работы точкой входа в программу является функция *main*, которая описана в файле *main.cpp*. Все используемые функции в *main* объявлены в заголовочном файле *wid.h,* и в свою очередь описаны в файле исходных текстов *wid.cpp.*

Графики функций построены в среде Maple, все полученные результаты записываются в файл *otput.txt*.

**1.Решение нелинейных уравнений**

**1.1.Метод простых итераций**

**График функции** y=cos(x)+0.25\*x-0.5 :

****

**1.1.1** main.cpp

#include <QtCore/QCoreApplication>

#include "wid.h"

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

int iter; // Счётчик колличества итераций

float accuracy = (float)0.1, left = (float)0.0, right = M\_PI, x, xk, q;

QString name = "output.txt"; // Фаил вывода

if((Sign(Function(left)) != Sign(Function(right)))&&(Sign(Derivative(left)) == Sign(Derivative(right))))

{

if((Fi(left)>=left)&&(Fi(left)<=right)&&(Fi(right)>=left)&&(Fi(right)<=right))

{

if((abs(Fi\_der(left))<1)&&(abs(Fi\_der(right))<1))

{

ofstream out;

out.open(QFile::encodeName(name).data()); // Открытие файла на запись

out << "k" << "\t" << "x" << "\t" << "fi(x)" << "\n";

xk = (left+right)/2; // Выбор начального приближения

iter = 0;

do

{

x = xk;

out << iter << "\t" << xk << "\t";

q = abs(Fi\_der(xk));

xk = Fi(x);

out << Fi(xk) << "\n";

iter++;

}

while(!CriterionTermination(xk, x, q, accuracy)); // Проверка удовлетворения заданной точности

out << iter << "\t" << xk << "\n\nAnswer: x = " << xk << "\n"; // Вывод ответа

out.close();

}

else

{

qWarning("Please, choose a different interval !");

qWarning("Becouse, |fi'(x)| >= 1 for some x = [%f, %f].", left, right);

}

}

else

{

qWarning("Please, choose a different interval !");

qWarning("Becouse, fi(x) != [%f,%f] for some x = [%f, %f].", left, right, left, right);

}

}

else

{

qWarning("Please, choose a different interval !");

qWarning("Becouse, function is not monotonic");

qWarning("or there is no solution in this interval: [%f, %f]", left, right);

}

cout << "OK";

return a.exec();

}

**1.1.2** wid.h

#ifndef WID\_H

#define WID\_H

#include <QVector>

#include <QObject>

#include <qmath.h>

#include <QtGui/QFileDialog>

#include <QDebug>

#include <QtDebug>

#include <iostream>

#include <QByteArray>

#include <qmath.h>

#include <stdio.h>

#include "math.h"

#include <cmath>

#include <fstream>

using namespace std;

#define pi 3.14159265358979323846264338327950288419716939937510

#define minimum 0.000000000000001

float Sign(float \_a); /// Определение знака числа

float Function(float \_x); /// Заданная функция

float Derivative(float \_x); /// Производная функции

float Second\_derivative(float \_x); /// Вторая производная функции

float Fi(float \_x); /// Функция fi

float Fi\_der(float \_x); /// Производная функции fi

float Fi\_sec\_der(float \_x); /// Вторая производная функции fi

bool CriterionTermination(float \_xk, float \_x, float \_q, float accuracy); /// Проверка удовлетворения заданной точности

#endif // WID\_H

**1.1.2** wid.cpp

#include "wid.h"

bool CriterionTermination(float \_xk, float \_x,float \_q, float accuracy)

{

if((\_q/(1-\_q))\*abs(\_xk-\_x) < accuracy)

return true;

return false;

}

float Fi(float \_x)

{

return qAcos(- 0.25\*\_x + 0.5);

}

float Fi\_der(float \_x)

{

return 0.25/(qSqrt(1 - (0.5 - 0.25\*\_x)\*(0.5 - 0.25\*\_x)));

}

float Fi\_sec\_der(float \_x)

{

return (0.5 - 0.25\*\_x)/(16\*qSqrt((1 - (0.5 - 0.25\*\_x)\*(0.5 - 0.25\*\_x))\*(1 - (0.5 - 0.25\*\_x)\*(0.5 - 0.25\*\_x))\*(1 - (0.5 - 0.25\*\_x)\*(0.5 - 0.25\*\_x))));

}

float Function(float \_x)

{

return qCos(\_x) + 0.25\*\_x - 0.5;

}

float Derivative(float \_x)

{

return - qSin(\_x) + 0.25;

}

float Second\_derivative(float \_x)

{

return - qCos(\_x);

}

float Sign(float \_a)

{

if (\_a < 0)

return -1.0;

else if (\_a == 0)

return 0.0;

else

return 1.0;

}

**1.1.3.** Вывод результатов (файл otput.txt)

**k x fi(x)**

**0 1.5708 1.43621**

**1 1.46329**

**Answer: x = 1.46329**

**1.2.Метод Ньютона**

**1.2.1.**main.cpp

#include <QtCore/QCoreApplication>

#include "wid.h"

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

int iter; // Счётчик числа итераций

float accuracy = 0.0001, left = 0.0, right = M\_PI, x, xk;

QString name = "output.txt"; // Фаил вывода

if((Sign(Function(left)) != Sign(Function(right)))&&(Sign(Derivative(left)) == Sign(Derivative(right))))

{

ofstream out;

out.open(QFile::encodeName(name).data()); // Открытие файла на запись

out << "k" << "\t" << "x" << "\t" << "f(x)" << "\t" << "f'(x)" << "\t" << "-f(x)/f'(x)" << "\n";

if(Function(left)\*Second\_derivative(left) > 0.0) // Выбор начального приближения

xk = left;

else

xk = right;

iter = 0;

do

{

x = xk;

out << iter << "\t" << xk << "\t";

xk = x-(Function(x))/(Derivative(x));

out << Function(xk) << "\t" << Derivative(xk) << "\t" << -(Function(xk))/(Derivative(xk)) << "\n";

iter++;

}

while(!CriterionTermination(xk, x, accuracy)); // Проверка удовлетворения заданной точности

out << iter << "\t" << xk << "\n\nAnswer: x = " << xk << "\n";

out.close();

}

else

qWarning("Please, choose a different interval !");

cout << "OK";

return a.exec();

}

**1.2.2.**wid.h

#ifndef WID\_H

#define WID\_H

#include <QVector>

#include <QObject>

#include <qmath.h>

#include <QtGui/QFileDialog>

#include <QDebug>

#include <QtDebug>

#include <iostream>

#include <QByteArray>

#include <stdio.h>

#include "math.h"

#include <cmath>

#include <fstream>

using namespace std;

#define pi 3.14159265358979323846264338327950288419716939937510

#define minimum 0.000000000000001

float Sign(float \_a); /// Определение знака числа

float Function(float \_x); /// Заданная функция

float Derivative(float \_x); /// Производная функции

float Second\_derivative(float \_x); /// Вторая производная функции

bool CriterionTermination(float \_xk, float \_x, float accuracy); /// Проверка удовлетворения заданной точности

#endif // WID\_H

**1.2.2.**wid.cpp

#include "wid.h"

bool CriterionTermination(float \_xk, float \_x, float accuracy)

{

if(abs(\_xk-\_x) < accuracy)

return true;

return false;

}

float Function(float \_x)

{

return qCos(\_x) + 0.25\*\_x - 0.5;

}

float Derivative(float \_x)

{

return - qSin(\_x) + 0.25;

}

float Second\_derivative(float \_x)

{

return - qCos(\_x);

}

float Sign(float \_a)

{

if (\_a < 0)

return -1.0;

else if (\_a == 0)

return 0.0;

else

return 1.0;

}

**1.2.2.**Вывод результатов

k x f(x) f'(x) -f(x)/f'(x)

0 3.14159 1.96017 0.529416 -3.70251

1 6 -0.590029 -0.497376 -1.18628

2 2.29749 0.221383 -0.646234 0.342575

3 1.1112 -0.019805 -0.743161 -0.0266497

4 1.45378 -4.45995e-005 -0.739698 -6.02942e-005

5 1.42713 1.87596e-008 -0.739689 2.53615e-008

6 1.42707

Answer: x = 1.42707

**2.Решение нелинейных систем уравнений**

**Графики функций** **:**



**2.1. Метод простых итераций**

**2.1.1.** main.cpp

#include <QtCore/QCoreApplication>

#include "wid.h"

int main(int argc, char \*argv[])

{

QCoreApplication asd(argc, argv);

int iter; // Счётчик числа итераций

float accuracy = 0.01, left1 = 0.25, right1 = 0.75, left2 = 2.0, right2 = 2.5, x[N], xk[N], q;

QString name = "output.txt"; // Имя файла вывода резуцльтата

ofstream out;

out.open(QFile::encodeName(name).data()); // Открытие файла на запись

out << "k" << "\t" << "x1" << "\t" << "fi1(x1,x2)" << "\n";

out << "\t" << "x2" << "\t" << "fi2(x1,x2)" << "\n";

out << "--------------------------------------------------" << "\n";

xk[0] = (left1 + right1)/2;

xk[1] = (left2 + right2)/2;

iter = 0;

do

{

x[0] = xk[0];

x[1] = xk[1];

if((x[0] > left1)&&(x[0] < right1)&&(x[1]>left2)&&(x[1]<right2))

{

if(abs(Fi1\_der\_x1(x[0], x[1])) + abs(Fi1\_der\_x2(x[0], x[1])) < abs(Fi2\_der\_x1(x[0], x[1])) + abs(Fi2\_der\_x2(x[0], x[1])))

q = abs(Fi2\_der\_x1(x[0], x[1])) + abs(Fi2\_der\_x2(x[0], x[1]));

else

q = abs(Fi1\_der\_x1(x[0], x[1])) + abs(Fi1\_der\_x2(x[0], x[1]));

if(q >= 1)

{

qWarning ("Please, choose a different interval !");

qFatal ("Becouse, max ||fi'(x1, x2)|| >= 1.");

}

out << iter << "\t" << x[0] << "\t" << Fi1(x[0], x[1]) << "\n";

out << "\t" << x[1] << "\t" << Fi2(x[0], x[1]) << "\n";

xk[0] = Fi1(x[0], x[1]);

xk[1] = Fi2(x[0], x[1]);

out << "--------------------------------------------------" << "\n";

iter++;

}

else

{

qWarning ("Please, choose a different interval !");

qFatal ("Becouse, successive approximations of the solution left the search area.");

}

}

while(!CriterionTermination(xk, x, q, accuracy)); // Проверка выполнения критерия окончания

out << iter << "\t" << xk[0] << "\n" << "\t" << xk[1] << "\n\nAnswer: \tx1 = " << xk[0] << "\n\t\tx2 = " << xk[1] << "\n"; // Вывод ответа

cout << "OK";

out.close(); // Закрытие файла

return asd.exec();

}

**2.1.2.** wid.h

#ifndef WID\_H

#define WID\_H

#include <QVector>

#include <QObject>

#include <qmath.h>

#include <QtGui/QFileDialog>

#include <QDebug>

#include <QtDebug>

#include <iostream>

#include <QByteArray>

#include <stdio.h>

#include "math.h"

#include <cmath>

#include <fstream>

#define pi 3.14159265358979323846264338327950288419716939937510

#define minimum 0.000000000000001

#define N 2 //Размерность исходной матрицы

using namespace std;

float Sign(float \_a);/// Определение знака числа

float Fi1(float \_x1, float \_x2);/// Заданная функция 1

float Fi2(float \_x1, float \_x2);/// Заданная функция 2

float Fi1\_der\_x1(float \_x1, float \_x2);/// Производная функции 1 по х1

float Fi1\_der\_x2(float \_x1, float \_x2);/// Производная функции 1 по х2

float Fi2\_der\_x1(float \_x1, float \_x2);/// Производная функции 2 по х1

float Fi2\_der\_x2(float \_x1, float \_x2);/// Производная функции 2 по х2

bool CriterionTermination(float \_xk[N], float \_x[N], float \_q, float accuracy); /// Проверка удовлетворения заданной точности

#endif // WID\_H

**2.1.3.** wid.cpp

#include "wid.h"

bool CriterionTermination(float \_xk[N], float \_x[N], float \_q, float accuracy)

{

float max;

if(abs(\_xk[0] - \_x[0]) < abs(\_xk[1] - \_x[1]))

max = abs(\_xk[1] - \_x[1]);

else

max = abs(\_xk[0] - \_x[0]);

if((\_q / (1 - \_q)) \* max < accuracy)

return true;

return false;

}

float Fi1(float \_x1, float \_x2)

{

return qCos(\_x2) + 1;

}

float Fi2(float \_x1, float \_x2)

{

return log10(\_x1 + 1) + 2;

}

float Fi1\_der\_x1(float \_x1, float \_x2)

{

return 0;

}

float Fi1\_der\_x2(float \_x1, float \_x2)

{

return -qSin(\_x2);

}

float Fi2\_der\_x1(float \_x1, float \_x2)

{

return 1/((\_x1+1)\*qLn(10));

}

float Fi2\_der\_x2(float \_x1, float \_x2)

{

return 0;

}

float Sign(float \_a)

{

if (\_a < 0)

return -1.0;

else if (\_a == 0)

return 0.0;

else

return 1.0;

}

**2.1.4.** Вывод результатов(файл otput.txt)

k x1 fi1(x1,x2)

x2 fi2(x1,x2)

--------------------------------------------------

0 0.5 0.371826

2.25 2.17609

--------------------------------------------------

1 0.371826 0.430995

2.17609 2.1373

--------------------------------------------------

2 0.430995 0.463316

2.1373 2.15564

--------------------------------------------------

3 0.463316 0.447932

2.15564 2.16534

--------------------------------------------------

4 0.447932 0.439871

2.16534 2.16075

--------------------------------------------------

5 0.439871 0.443679

2.16075 2.15832

--------------------------------------------------

6 0.443679 0.445695

2.15832 2.15947

--------------------------------------------------

7 0.445695 0.444741

2.15947 2.16008

--------------------------------------------------

8 0.444741

2.16008

Answer: x1 = 0.444741

x2 = 2.16008

**2.2. Метод Ньютона**

**2.2.1.**main.cpp

#include <QtCore/QCoreApplication>

#include "wid.h"

int main(int argc, char \*argv[])

{

QCoreApplication asd(argc, argv);

int iter; // Счётчик числа итераций

float accuracy = 0.01, left1 = 0.25, right1 = 0.75, left2 = 2.0, right2 = 2.5, x[N], xk[N], a[N][N], detA1, detA2, detJ;

QString name = "output.txt"; // Имя файла на вывод информации

ofstream out;

out.open(QFile::encodeName(name).data()); // Открытие файла на запись

out << "k" << "\t" << "x1" << "\t" << "f1(x1,x2)" << "\t" << "df1(x1,x2)/dx1" << "\t" << "df1(x1,x2)/dx2" << "\t" << "det A1" << "\t" << "det A2" << "\t" << "det J" << "\n";

out << "\t" << "x2" << "\t" << "f2(x1,x2)" << "\t" << "df2(x1,x2)/dx1" << "\t" << "df2(x1,x2)/dx2" << "\n";

out << "----------------------------------------------------------------------------------------------------" << "\n";

xk[0] = (left1 + right1)/2; // Начальное приближение по х1

xk[1] = (left2 + right2)/2; // Начальное приближение по х2

iter = 0;

do

{

x[0] = xk[0];

x[1] = xk[1];

a[0][0] = F1\_der\_x1(x[0], x[1]);

a[0][1] = F1\_der\_x2(x[0], x[1]);

a[1][0] = F2\_der\_x1(x[0], x[1]);

a[1][1] = F2\_der\_x2(x[0], x[1]);

detJ = Determinant(a);

a[0][0] = Function1(x[0], x[1]);

a[0][1] = F1\_der\_x2(x[0], x[1]);

a[1][0] = Function2(x[0], x[1]);

a[1][1] = F2\_der\_x2(x[0], x[1]);

detA1 = Determinant(a);

a[0][0] = F1\_der\_x1(x[0], x[1]);

a[0][1] = Function1(x[0], x[1]);

a[1][0] = F2\_der\_x1(x[0], x[1]);

a[1][1] = Function2(x[0], x[1]);

detA2 = Determinant(a);

out << iter << "\t" << x[0] << "\t" << Function1(x[0], x[1]) << "\t" << F1\_der\_x1(x[0], x[1]) << "\t" << F1\_der\_x2(x[0], x[1]) << "\t";

out << detA1 << "\t" << detA2 << "\t" << detJ << "\n";

out << "\t" << x[1] << "\t" << Function2(x[0], x[1]) << "\t" << F2\_der\_x1(x[0], x[1]) << "\t" << F2\_der\_x2(x[0], x[1]) << "\n";

xk[0] = x[0] - detA1/detJ;

xk[1] = x[1] - detA2/detJ;

out << "----------------------------------------------------------------------------------------------------" << "\n";

iter++;

}

while(!CriterionTermination(xk, x, accuracy)); // Проверка выполнения критерия окончания

out << iter << "\t" << xk[0] << "\n" << "\t" << xk[1] << "\n\nAnswer: \tx1 = " << xk[0] << "\n\t\tx2 = " << xk[1] << "\n"; // Вывод ответа

out.close(); // Закрытие файла

cout << "OK";

return asd.exec();

}

**2.2.2.**wid.h

#ifndef WID\_H

#define WID\_H

#include <QVector>

#include <QObject>

#include <qmath.h>

#include <QtGui/QFileDialog>

#include <QDebug>

#include <QtDebug>

#include <iostream>

#include <QByteArray>

#include <stdio.h>

#include "math.h"

#include <cmath>

#include <fstream>

using namespace std;

#define pi 3.14159265358979323846264338327950288419716939937510

#define minimum 0.000000000000001

#define N 2 //Размерность исходной матрицы

float Sign(float \_a);/// Определение знака числа

float Function1(float \_x1, float \_x2);/// Заданная функция 1

float Function2(float \_x1, float \_x2);/// Заданная функция 2

float F1\_der\_x1(float \_x1, float \_x2);/// Производная функции 1 по х1

float F1\_der\_x2(float \_x1, float \_x2);/// Производная функции 1 по х2

float F2\_der\_x1(float \_x1, float \_x2);/// Производная функции 2 по х1

float F2\_der\_x2(float \_x1, float \_x2);/// Производная функции 2 по х2

bool CriterionTermination(float \_xk[N], float \_x[N], float accuracy);/// Проверка удовлетворения заданной точности

float Determinant(float \_mas[N][N]);/// Вычисление определителя

#endif // WID\_H

**2.2.3.**wid.cpp

#include "wid.h"

float Determinant(float \_mas[N][N])

{

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

}

bool CriterionTermination(float \_xk[N], float \_x[N], float accuracy)

{

float max;

if(abs(\_xk[0] - \_x[0]) < abs(\_xk[1] - \_x[1]))

max = abs(\_xk[1] - \_x[1]);

else

max = abs(\_xk[0] - \_x[0]);

if(max < accuracy)

return true;

return false;

}

float Function1(float \_x1, float \_x2)

{

return \_x1 - qCos(\_x2) - 1;

}

float Function2(float \_x1, float \_x2)

{

return \_x2 - log10(\_x1 + 1) - 2;

}

float F1\_der\_x1(float \_x1, float \_x2)

{

return 1;

}

float F1\_der\_x2(float \_x1, float \_x2)

{

return qSin(\_x2);

}

float F2\_der\_x1(float \_x1, float \_x2)

{

return -1/((\_x1+1)\*qLn(10));

}

float F2\_der\_x2(float \_x1, float \_x2)

{

return 1;

}

float Sign(float \_a)

{

if (\_a < 0)

return -1.0;

else if (\_a == 0)

return 0.0;

else

return 1.0;

}

**2.2.4.**Вывод результатов(otput.txt)

k x1 f1(x1,x2) df1(x1,x2)/dx1 df1(x1,x2)/dx2 det A1 det A2 det J

x2 f2(x1,x2) df2(x1,x2)/dx1 df2(x1,x2)/dx2

----------------------------------------------------------------------------------------------------

0 0.5 0.128174 1 0.778073 0.0706672 0.111019 1.22528

2.25 0.0739087 -0.28953 1

----------------------------------------------------------------------------------------------------

1 0.442325 -0.0024803 1 0.831721 -0.0027544 -0.00041728 1.25044

2.15939 0.000329554 -0.301107 1

----------------------------------------------------------------------------------------------------

2 0.444528

2.15973

Answer: x1 = 0.444528

x2 = 2.15973

**Вывод:**

При решении нелинейных уравнений, например методом простых итераций, необходимо задать начальный интервал, на котором должно находиться решение уравнения, что зачастую затруднительно, т.к. построить график некоторых функций бывает весьма затруднительно. В этом и состоит один из недостатков данного метода.

При использования метода Ньютона надо учитывать то, что его сходимость сильно зависит от начальной точки. Однако при удачной начальной точке метод Ньютона сходится достаточно быстро.

При вычислении производной делается весьма сильное допущение: производная функции заменяется конечной разностью, что является грубым, т.к. приходится отказываться от предельного перехода. Однако в некоторых случаях такое допущение является достаточно удобным в реализации и в использовании.