Министерство науки и высшего образования Российской Федерации

**Муромский институт (филиал)**

федерального государственного бюджетного образовательного учреждения высшего образования

**«Владимирский государственный университет**

**имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(МИ ВлГУ)**

Факультет ИТР

Кафедра ФПМ

*КУРСОВАЯ*

*РАБОТА*

По дисциплине: Технологии и методы программирования

Тема: Разработка приложения для решения систем нелинейных уравнений методом Ньютона

Оценка работы Руководитель

Соколов М.С.

(фамилия, инициалы)

(подпись) (дата)

Члены комиссии Студент ПМИ-122

(группа)

Штырлов Д.М.

(фамилия, инициалы) (подпись) (фамилия, инициалы)

(фамилия, инициалы) (подпись) (подпись) (дата)

Муром 2024

Содержание

[Введение 3](#_Toc178604179)

[1. Анализ технического задания 4](#_Toc178604180)

[2. Программная реализация 10](#_Toc178604181)

[3. Тестирование 12](#_Toc178604182)

[Заключение 15](#_Toc178604183)

[Список использованной литературы 16](#_Toc178604184)

[Приложение A 17](#_Toc178604185)

[Приложение Б 23](#_Toc178604186)

# Введение

В математике и естественных науках нелинейная система (или нелинейная система) — это система, в которой изменение выходных данных не пропорционально изменению входных данных. Нелинейные проблемы представляют интерес для инженеров, биологов, физиков, математиков и многих других ученых, поскольку большинство систем по своей сути нелинейны в природе. Нелинейные динамические системы, описывающие изменения переменных с течением времени, могут казаться хаотичными, непредсказуемыми или противоречащими интуиции, контрастируя с гораздо более простыми линейными системами.

Как правило, поведение нелинейной системы описывается в математике нелинейной системой уравнений, которая представляет собой набор одновременных уравнений, в которых неизвестные появляются как переменные многочлена степени выше единицы или в аргументе функции, которая не является многочленом первой степени. Системы можно определить как нелинейные, независимо от того, присутствуют ли в уравнениях известные линейные функции.

Существует множество методов решения систем нелинейных уравнений. В работе рассматривается метод Ньютона. Основная идея этого метода состоит в выделении из уравнений системы линейных частей, которые являются главными при малых приращениях аргументов. Это позволяет свести исходную задачу к решению последовательности систем линейных уравнений.

Основные задачи при написании курсовой работы:

1. Изучить численный метод ньютона для решения системы уравнений

2. На основе изученного метода составить алгоритм для будущей программы

3. Реализовать решение системы уравнений

# 1. Анализ технического задания

Согласно техническому заданию на курсовое проектирование необходимо разработать программу, позволяющую решать систему нелинейных уравнений используя метод Ньютона. Так как нелинейным уравнением может называться любое, чье изменение выходных данных не пропорционально извинению входных, мною была выбрана конкретная система (1), однако написанный впоследствии алгоритм и программа смогут (при небольшой модификации) решать любые системы.

(1)

Для начала нужно определится с алгоритмом работы самого метода.

Данный метод является обобщением метода касательных. Существенную роль в этом методе играет специальная матрица – так называемая матрица Якоби (или якобиян):

Очевидно, что построить её можно лишь при условии, что каждая из функций, входящая в систему:

дифференцируема по каждой из переменных.

Метод касательных применительно к одному уравнению заключается в построении итерационной последовательности:

Обобщением этой формулы на системы уравнений является следующая формула:

где X - n-мерный вектор-столбец, f - n-мерный вектор-функция, k - номер итерации, - матрица, обратная матрице J на k-той итерации, J - матрица Якоби.

Отметим эмпирическое правило: если задана абсолютная погрешность приближенного решения системы, то процесс итерации в большинстве случаев можно остановить, когда для каждой из переменных выполнено неравенство:

С целью лучшего понимания следует разобрать пример:

С точностью .

Элементы Якобиана имеют вид:

Точки начального приближения x и y равны 0.7 и 0.7 соответственно. Находим:

Эмпирически ответ можно получить, рассуждая следующим образом. Пятая цифра после запятой в величине x и y совпадает с последующими цифрами, стоящими на том же месте в 4-й итерации

Для выполнения поставленной задачи применён язык программирования C+ +, что связанно с достоинствами данного языка:

1. Высокая производительность и скорость работы программ написанных на данном языке

2. Кроссплатформенность — совместимость и возможность запуска на большом количестве разнообразных платформ

3. Интегрируемость – язык является духовным наследником языка С и полностью с ним совместим, что в свою очередь позволяет связывать работу написанных на нем приложений с другими языками

В свою очередь стоит отметить и ряд недостатков:

1. Сложность — за долгое время развития языка, в C++ появилось

множество конструкций, зачастую малоиспользуемых и дублирующих друг

друга.

2. Ручное управление памятью и строгая типизация

В качестве IDE используется Microsoft Visual Studio, поскольку она позволяет выполнять такие операции как редактирование, сборка, тестирование, отладка в рамках одной программы. Помимо этого, версия community является бесплатной.

Для разработки графического интерфейса пользователя используется библиотека MFC.

Теперь для реализации программы стоит определиться с ее пошаговым алгоритмом:

1. Изначально будут заданы основные параметры, такие как коэффициенты a, b, с и начальное приближение x и y. в который они последовательно записаны, после чего буду помещены в контейнер STL, а именно Vector. Такой выбор был сделан из-за того, что он имеет удобную реализацию произвольного доступа к элементам и нефиксированный размер, по сравнению с другими контейнерами, как показано в таблице 1, что позволит при необходимости вносить в него неограниченное количество параметров, что позволит решать любые системы уравнений. Также ввод будет реализован из заранее созданного файла.

|  |  |
| --- | --- |
| Array –  Плюсы: нет дополнительного расхода памяти для хранения элементов.  Минусы: не эффективен для очень большого объемов данных. Фиксированный размер. | Vector  Плюсы: удобная реализация произвольного доступа к элементам.  Размер не фиксирован, выделяется много места. |

Таблица 1 - Контейнеры

2. После ввода параметров будет производиться расчет и нахождение корней уравнений с использование метода Ньютона. После чего результат будет выведен в поля ввода.

3. В итоге на выходе программа будет получать корни x и y при заданном начальном приближении.

Общий вид блок схемы работы алгоритма представлен на схеме 1.

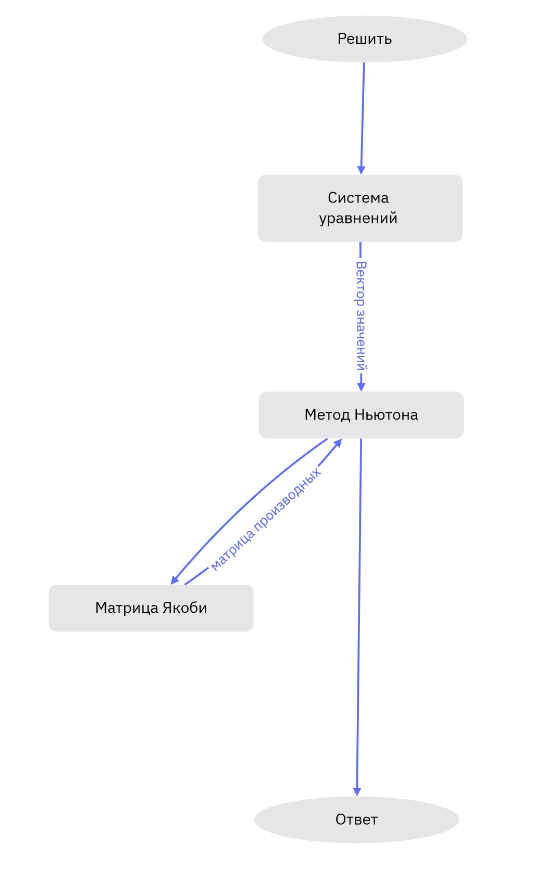


Схема 1 – Работа алгоритма

Интерфейс приложения будет содержать в себе:

1. Текстовые подсказки для ввода

2. Поля ввода для каждого коэффициента

3. Поля ввода для вывода результата работы программы

4. Меню, представленное на рисунке 2

5. Кнопки управления для запуска работы программы и закрытия приложения

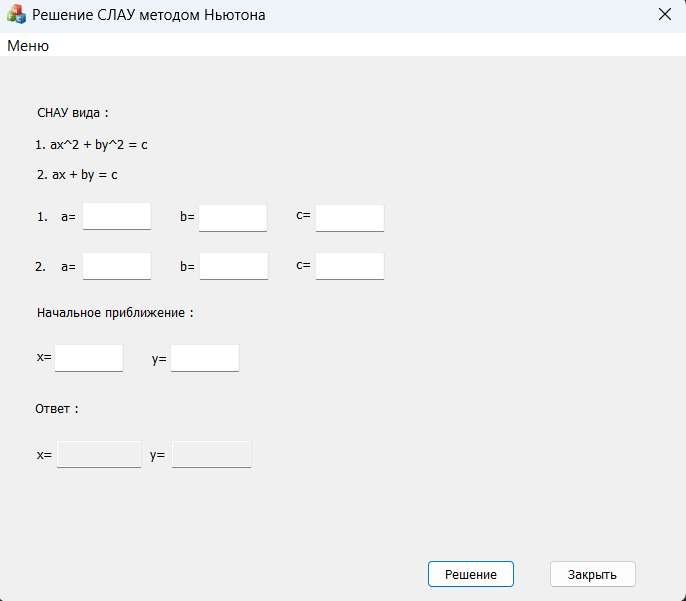
Меню, как важный элемент интерфейса и управления содержит:

1. Продублированную кнопку выхода из приложения

2. Кнопку для введения параметров из заранее созданного и заполненного файла

3. Кнопку для сохранения результата работы программы в автоматически создающийся файл

4. Кнопка справки с текстовым пояснением правил использования приложения



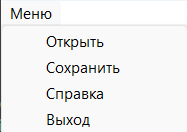


Рисунок 1 – Интерфейс Рисунок 2 – Меню

# 2. Программная реализация

Цель работы – решение системы уравнений, будет получена в процессе выполнения вышеописанного алгоритма, для этого следует рассмотреть каждый пункт и определить средства необходимые для его программной реализации.

1. Параметры будут вводиться и считывать из элемента Edit Control, либо при помощи меню и файла формата .txt при помощи метода GetDlgItemText и передаваться в функцию f() задающую систему уравнений на рисунке 3.1

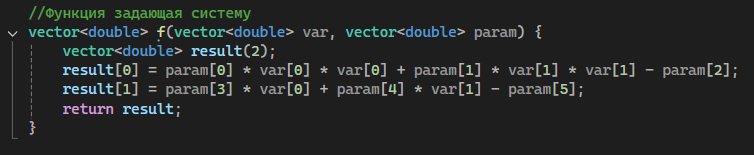


Рисунок 3.1 – Функция f()

2. Сам метод Ньютона будет реализован функцией (Рисунок 3.2), в том числе функцией нахождения матрицы Якоби (Рисунок 4), с применением методов библиотеки Math, например pow().

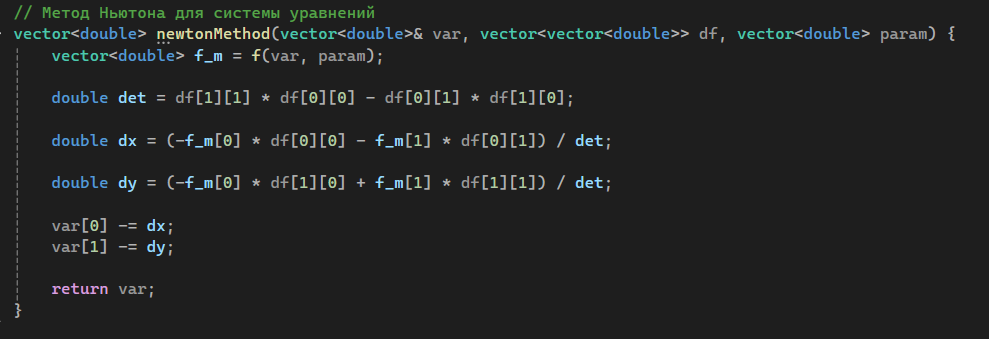


Рисунок 3.2 – Функция newtonMethod

Здесь df является массивом производных, полученным в ходе работы функции jacobian на рисунке 4.

Функция newtonMethod возвращает значение var которое является вектором из 2х значений, в конечном итоге по окончанию итераций являющихся корнями.

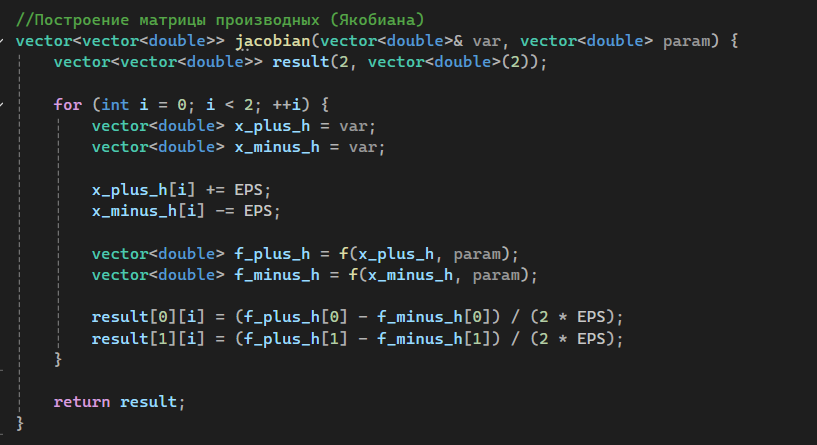


Рисунок 4 – Функция jacobian

3. Результат будет выводится в элементы Edit Control и при необходимости, с помощью пункта меню в автоматически создаваемый файл

Схема работы алгоритма представлена на схеме 2.

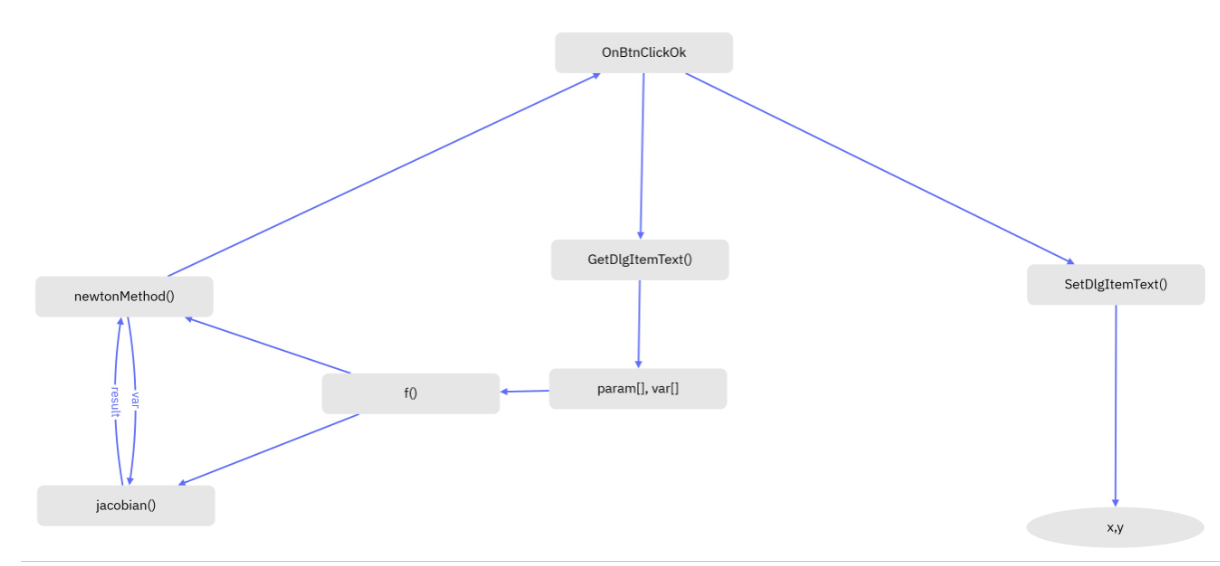


Схема 2

# 3. Тестирование

В ходе тестирования стоит проверить работоспособность всех элементов управления и самого алгоритма.

1. Рисунки 5, 6, открытие файла и заполнение полей

2. Рисунок 7 справка по заполнению файлов out.txt и in.txt

3. Рисунок 8 решение системы с указанными параметрами

4. Рисунок 9 сохранение параметров и корней в файл

5. Рисунок 10 ситуация, при которой корни не найдены

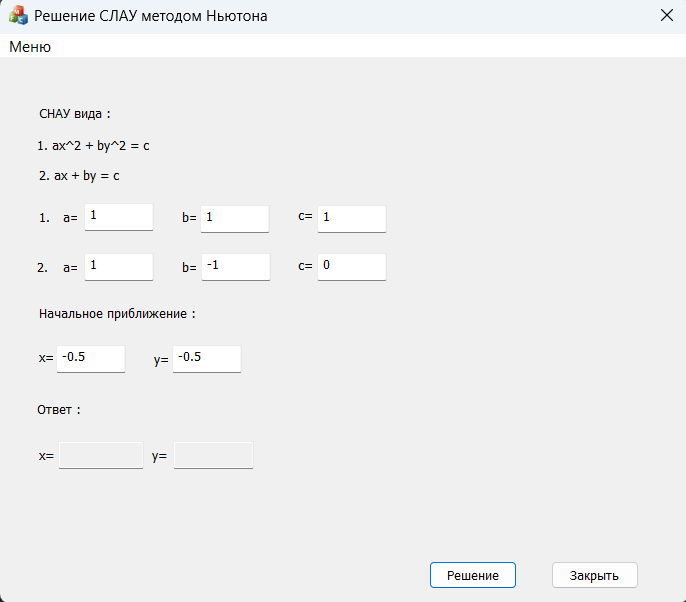


Рисунок 5

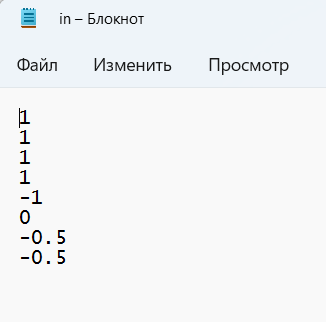


Рисунок 6

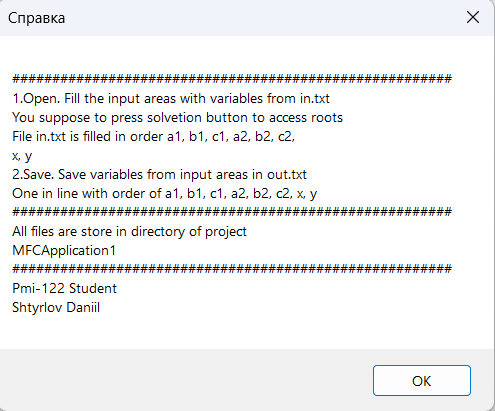
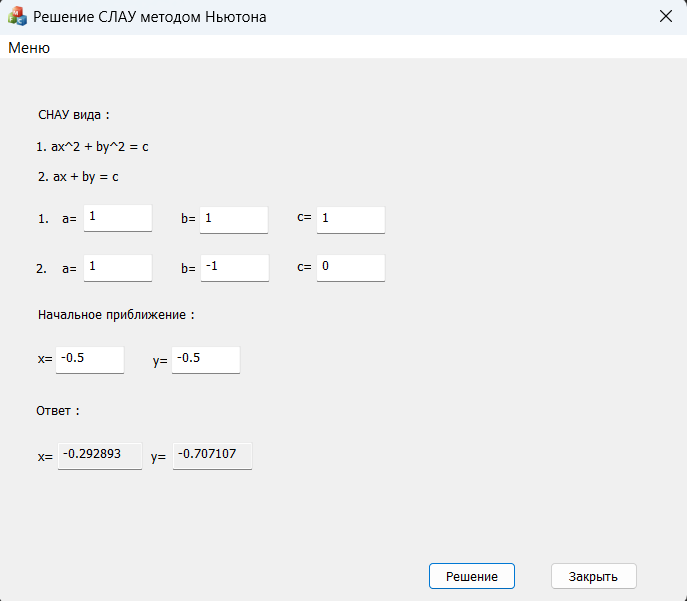
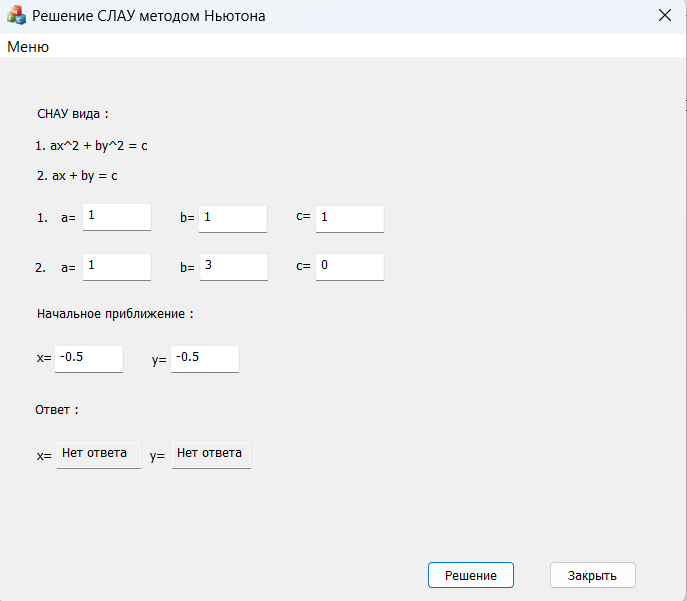


Рисунок 7



Рисунок 8

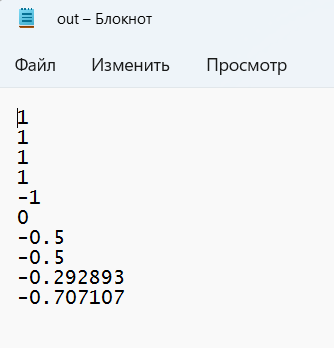


Рисунок 9

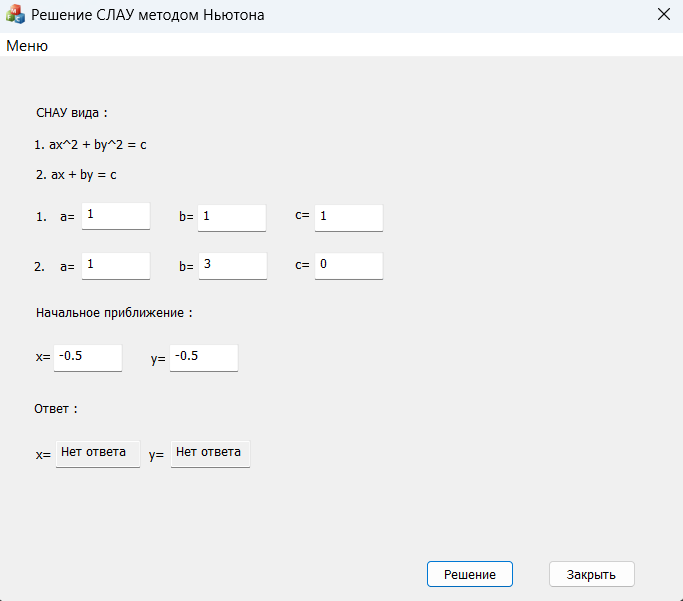
****

Рисунок 10

Как и было указано ранее приложение нацелено на решение определенной системы уравнений, однако алгоритм может решать любые системы при изменение кода на рисунке 11, где var[0] и var[1] – начальное приближение, param[0-5] – коэффициенты. Пример ввода других уравнений на рисунке 12.

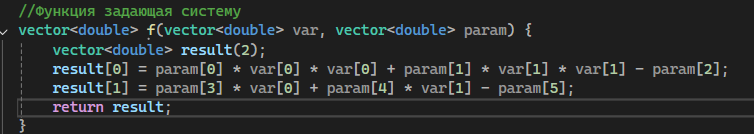


Рисунок 11

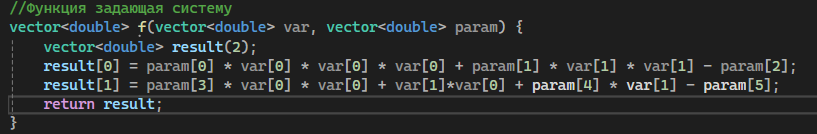


Рисунок 12

# Заключение

Таким образом, в ходе курсовой работы было написано приложение для решения систем нелинейных уравнений методом Ньютона, оно может быть полезно и применимо для учеников и студентов при выполнении заданий, для деятелей науки – математиков и физиков в решении прикладных задач.

Самым полезным применением программы, я вижу, использование инженерами и архитекторами, так как не редко при строительстве производятся расчеты связанные с колебаниями для увеличения устойчивости построек в сейсмически активных районах планеты.

В ходе работы мной были встречены немалые трудности, связанные со знанием языка C++ и многие из них были решены, однако приложение требует доработки, что даст ей большую сферу применения, универсальность и удобство в использовании.

# Список использованной литературы

1. Лапчик М. П., Рагулова М. И., Хеннер Е. К.; Под редакцией Лапчика М. П. Численные методы: // Учебное пособие для студ. вузов. – М.: Академия, 2004. – 384 с.

2. Множество пользователей сайта; Тематические статьи в ветке C++ : // <https://stackoverflow.com/>. – Интернет, 2024.

3. А.И. Зинина, В.И. Копнина; Численные методы линейной и нелинейной алгебры: // Учебное пособие: Саратов, 2016. – 22 с.

# Приложение A

Далее представлен код программы в полном объёме:

#include "pch.h"

#include "framework.h"

#include "MFCApplication1.h"

#include "MFCApplication1Dlg.h"

#include "afxdialogex.h"

#include "vector"

#include "cmath"

#include "string"

#include "fstream"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

using namespace std;

const double EPS = 0.000001;

// Диалоговое окно CAboutDlg используется для описания сведений о приложении

class CAboutDlg : public CDialogEx

{

public:

CAboutDlg();

// Данные диалогового окна

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_ABOUTBOX };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // поддержка DDX/DDV

// Реализация

protected:

DECLARE\_MESSAGE\_MAP()

};

CAboutDlg::CAboutDlg() : CDialogEx(IDD\_ABOUTBOX)

{

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)

END\_MESSAGE\_MAP()

// Диалоговое окно CMFCApplication1Dlg

CMFCApplication1Dlg::CMFCApplication1Dlg(CWnd\* pParent /\*=nullptr\*/)

: CDialogEx(IDD\_MFCAPPLICATION1\_DIALOG, pParent)

{

m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);

}

void CMFCApplication1Dlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

}

BEGIN\_MESSAGE\_MAP(CMFCApplication1Dlg, CDialogEx)

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_BN\_CLICKED(IDCANCEL, &CMFCApplication1Dlg::OnBnClickedCancel)

ON\_BN\_CLICKED(IDOK, &CMFCApplication1Dlg::OnBnClickedOk)

ON\_COMMAND(ID\_32774, &CMFCApplication1Dlg::OnMyExit)

ON\_COMMAND(ID\_32773, &CMFCApplication1Dlg::OnDocumentation)

ON\_COMMAND(ID\_32771, &CMFCApplication1Dlg::OnFileOpen)

ON\_COMMAND(ID\_32772, &CMFCApplication1Dlg::OnFileSave)

END\_MESSAGE\_MAP()

// Обработчики сообщений CMFCApplication1Dlg

BOOL CMFCApplication1Dlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

// Добавление пункта "О программе..." в системное меню.

// IDM\_ABOUTBOX должен быть в пределах системной команды.

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != nullptr)

{

BOOL bNameValid;

CString strAboutMenu;

bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);

ASSERT(bNameValid);

if (!strAboutMenu.IsEmpty())

{

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

// Задает значок для этого диалогового окна. Среда делает это автоматически,

// если главное окно приложения не является диалоговым

SetIcon(m\_hIcon, TRUE); // Крупный значок

SetIcon(m\_hIcon, FALSE); // Мелкий значок

// TODO: добавьте дополнительную инициализацию

return TRUE; // возврат значения TRUE, если фокус не передан элементу управления

}

void CMFCApplication1Dlg::OnSysCommand(UINT nID, LPARAM lParam)

{

if ((nID & 0xFFF0) == IDM\_ABOUTBOX)

{

CAboutDlg dlgAbout;

dlgAbout.DoModal();

}

else

{

CDialogEx::OnSysCommand(nID, lParam);

}

}

// При добавлении кнопки свертывания в диалоговое окно нужно воспользоваться приведенным ниже кодом,

// чтобы нарисовать значок. Для приложений MFC, использующих модель документов или представлений,

// это автоматически выполняется рабочей областью.

void CMFCApplication1Dlg::OnPaint()

{

if (IsIconic())

{

CPaintDC dc(this); // контекст устройства для рисования

SendMessage(WM\_ICONERASEBKGND, reinterpret\_cast<WPARAM>(dc.GetSafeHdc()), 0);

// Выравнивание значка по центру клиентского прямоугольника

int cxIcon = GetSystemMetrics(SM\_CXICON);

int cyIcon = GetSystemMetrics(SM\_CYICON);

CRect rect;

GetClientRect(&rect);

int x = (rect.Width() - cxIcon + 1) / 2;

int y = (rect.Height() - cyIcon + 1) / 2;

// Нарисуйте значок

dc.DrawIcon(x, y, m\_hIcon);

}

else

{

CDialogEx::OnPaint();

}

}

// Система вызывает эту функцию для получения отображения курсора при перемещении

// свернутого окна.

HCURSOR CMFCApplication1Dlg::OnQueryDragIcon()

{

return static\_cast<HCURSOR>(m\_hIcon);

}

void CMFCApplication1Dlg::OnBnClickedCancel()

{

// TODO: добавьте свой код обработчика уведомлений

CDialogEx::OnCancel();

}

//Функция задающая систему

vector<double> f(vector<double> var, vector<double> param) {

vector<double> result(2);

result[0] = param[0] \* var[0] \* var[0] + param[1] \* var[1] \* var[1] - param[2];

result[1] = param[3] \* var[0] + param[4] \* var[1] - param[5];

return result;

}

//Построение матрицы производных (Якобиана)

vector<vector<double>> jacobian(vector<double>& var, vector<double> param) {

vector<vector<double>> result(2, vector<double>(2));

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

vector<double> x\_plus\_h = var;

vector<double> x\_minus\_h = var;

x\_plus\_h[i] += EPS;

x\_minus\_h[i] -= EPS;

vector<double> f\_plus\_h = f(x\_plus\_h, param);

vector<double> f\_minus\_h = f(x\_minus\_h, param);

result[0][i] = (f\_plus\_h[0] - f\_minus\_h[0]) / (2 \* EPS);

result[1][i] = (f\_plus\_h[1] - f\_minus\_h[1]) / (2 \* EPS);

}

return result;

}

// Метод Ньютона для системы уравнений

vector<double> newtonMethod(vector<double>& var, vector<vector<double>> df, vector<double> param) {

vector<double> f\_m = f(var, param);

double det = df[1][1] \* df[0][0] - df[0][1] \* df[1][0];

double dx = (-f\_m[0] \* df[0][0] - f\_m[1] \* df[0][1]) / det;

double dy = (-f\_m[0] \* df[1][0] + f\_m[1] \* df[1][1]) / det;

var[0] -= dx;

var[1] -= dy;

return var;

}

void CMFCApplication1Dlg::OnBnClickedOk()

{

// TODO: добавьте свой код обработчика уведомлений

vector<int> EDC\_(8);

vector<CString> param\_str(6);

vector<double> param(6);

vector<CString> var\_str(2);

vector<double> var(2);

//Получение значений из эдит контролов

GetDlgItemText(IDC\_EDIT1, param\_str[0]);

GetDlgItemText(IDC\_EDIT2, param\_str[1]);

GetDlgItemText(IDC\_EDIT3, param\_str[2]);

GetDlgItemText(IDC\_EDIT4, param\_str[3]);

GetDlgItemText(IDC\_EDIT5, param\_str[4]);

GetDlgItemText(IDC\_EDIT6, param\_str[5]);

GetDlgItemText(IDC\_EDIT7, var\_str[0]);

GetDlgItemText(IDC\_EDIT8, var\_str[1]);

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

var[i] = \_tstof(var\_str[i]);

}

for (int i = 0; i <= 5; i++) {

param[i] = \_tstof(param\_str[i]);

}

vector<vector<double>> jac = jacobian(var, param);

int count = 0;

while (count < 200) {

count++;

double prev\_x = var[0];

double prev\_y = var[1];

newtonMethod(var, jac, param);

if (fabs(var[0] - prev\_x) < EPS && fabs(var[1] - prev\_y) < EPS)

{

break;

}

}

var\_str[0] = to\_string(var[0]).c\_str();

var\_str[1] = to\_string(var[1]).c\_str();

if (isnan(var[0]) && isnan(var[1])) {

SetDlgItemText(IDC\_EDIT9, L"Нет ответа");

SetDlgItemText(IDC\_EDIT10, L"Нет ответа");

}

else {

SetDlgItemText(IDC\_EDIT9, var\_str[0]);

SetDlgItemText(IDC\_EDIT10, var\_str[1]);

}

}

void CMFCApplication1Dlg::OnMyExit()

{

// TODO: добавьте свой код обработчика команд

CDialogEx::OnOK();

}

void CMFCApplication1Dlg::OnDocumentation()

{

// TODO: добавьте свой код обработчика команд

setlocale(LC\_ALL, "RUS");

string line, Doc\_text;

ifstream Doc("Documentation.txt");

if (!Doc) {

MessageBox(L"Невозможно открыть файл");

}

else {

while (getline(Doc, line)) {

Doc\_text.append(line);

Doc\_text.append("\n");

}

CString CDoc\_text(Doc\_text.c\_str());

MessageBox(CDoc\_text, L"Справка");

}

Doc.close();

}

void CMFCApplication1Dlg::OnFileOpen()

{

// TODO: добавьте свой код обработчика команд

ifstream infile("in.txt");

vector<CString> vars(8);

string fvar;

if (!infile) {

MessageBox(L"Невозможно открыть файл");

}

else {

for (int i = 0; i < 8; i++) {

getline(infile, fvar, '\n');

vars[i] = fvar.c\_str();

}

SetDlgItemText(IDC\_EDIT1,vars[0]);

SetDlgItemText(IDC\_EDIT2, vars[1]);

SetDlgItemText(IDC\_EDIT3, vars[2]);

SetDlgItemText(IDC\_EDIT4, vars[3]);

SetDlgItemText(IDC\_EDIT5, vars[4]);

SetDlgItemText(IDC\_EDIT6, vars[5]);

SetDlgItemText(IDC\_EDIT7, vars[6]);

SetDlgItemText(IDC\_EDIT8, vars[7]);

}

infile.close();

}

void CMFCApplication1Dlg::OnFileSave()

{

// TODO: добавьте свой код обработчика команд

ofstream outfile("out.txt");

vector<int> IDC(10);

vector<CString> parameters(10);

vector<const char\*> out\_param(10);

string save;

IDC[0] = IDC\_EDIT1;

IDC[1] = IDC\_EDIT2;

IDC[2] = IDC\_EDIT3;

IDC[3] = IDC\_EDIT4;

IDC[4] = IDC\_EDIT5;

IDC[5] = IDC\_EDIT6;

IDC[6] = IDC\_EDIT7;

IDC[7] = IDC\_EDIT8;

IDC[8] = IDC\_EDIT9;

IDC[9] = IDC\_EDIT10;

if (!outfile) {

MessageBox(L"Невозможно сохранить в файл");

}

else {

for (int i = 0; i < 10; i++) {

GetDlgItemText(IDC[i], parameters[i]);

outfile << \_tstof(parameters[i]) << '\n';

}

}

outfile.close();

}

# Приложение Б

Исходный файл .cpp и ПЗ выложены на GitHub по ссылке:

https://github.com/DanyaShtyrlov/CourseWork