МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тульский государственный университет»

Институт *прикладной математики и компьютерных наук*

Кафедра *вычислительной техники*

Курсовая работа по дисциплине

**ПРОГРАММИРОВАНИЕ**

на тему: Программный комплекс численного решения систем нелинейных уравнений.

Студент группы 220681 Шайхаттаров Д.В. \_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (Подпись, дата)

Руководитель работы Асс. Демидова А.В. \_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О., должность) (Подпись, дата)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тула, 2020

**Содержание**

[1. Техническое задание на проектирование 3](#_Toc51726791)

[2. Постановка задачи на проектирование 4](#_Toc51726792)

[3. Математическое обеспечение 4](#_Toc51726793)

[4. Алгоритмическое обеспечение 6](#_Toc51726794)

[5. Программная реализация 8](#_Toc51726795)

[6. Тестирование 10](#_Toc51726796)

[7. Инструкция программисту 12](#_Toc51726797)

[8. Инструкция пользователю 13](#_Toc51726798)

[9. Инструкция по установке 16](#_Toc51726799)

[Заключение 20](#_Toc51726800)

[Список использованных источников 21](#_Toc51726801)

[Приложения 22](#_Toc51726802)

# Техническое задание на проектирование

Необходимо разработать программное обеспечение с помощью технологии Windows Forms, которое сможет решать системы нелинейных уравнений, используя численных методы.

1. Входные данные

В качестве входных данных выступают строки, содержащие функции f(x,y), входящие в систему уравнений, x,y координата точки начального приближения, а также пользователь должен заполнить матрицу якоби, исходя из введённых им функций. Эти данные вводятся пользователем с клавиатуры, или автоматический все поля заполняются с помощью кнопки «ввести пример».

1. Выходные данные

Выходными данными являются координаты x и y, где эти функции пересекаются, что и является решением системы уравнений, а также значения функций в точке с этими координатами для проверки правильности расчётов.

1. Требования к среде разработки

Необходима среда разработки Visual Studio 2019 с установленными компонентами языка C#.

1. Компоненты

* TextBox
* Button
* Label
* DataGridView
* ToolStrip
* FlowLayoutPanel
* ToolStripContainer
* ColorDialog

1. Операции:

* Авторизация
* Нахождение решения системы нелинейных уравнений

1. Способ решения

Для решения поставленной задачи можно использовать технологию объектно-ориентированного программирования на языке С# и среду разработки Visual Studio 2019.

# Постановка задачи на проектирование

Необходимо разработать программное обеспечение, которое реализует нахождение компонент связности, мостов и точек сочленения. При этом в работе должны быть разработаны:

* интерфейс пользователя;
* система авторизации;
* инсталляционный пакет;
* функциональная часть.

Также при создании программы необходимо использовать следующие элементы:

* RichTextBox
* TableLayoutPanel
* ContextMenuStrip
* ColorDialog

# Математическое обеспечение

Решением системы уравнений является нахождение точки, где эти функции пересекаются. Нелинейными уравнениями мы называем любые уравнения график, которых не является прямой. Исходя из этого я делаю вывод, что задачей моего программного комплекса является решение системы уравнений любого вида, единственное ограничение, которое я ставлю – это, что функций должно быть 2 и они должны быть расположены в координатной плоскости x,y. Для решения этой задачи я выбрал численный метод – «метод секущих для решения систем нелинейных уравнений».

Метод секущих:

Пусть задана функция f(x) действительного переменного. Требуется найти корни уравнения f(x)=0.

Задача нахождения корней уравнения обычно решается в 2 этапа. На первом этапе проводится отделение корней, то есть выделение отрезков, содержащих только один корень. На втором этапе, используя начальное приближение, строится итерационный процесс, позволяющий уточнить значение отыскиваемого корня.

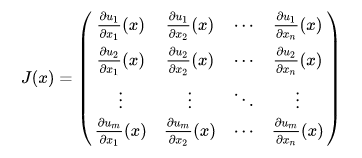
Метод секущих получается из метода касательных заменой f'(x) разностным приближением: f'(x^k) \approx \frac{f(x^k)-f(x^{k-1})}{x^k-x^{k-1}}..

В результате получим формулу итерационного процесса:

x^{k+1}=x^k-\frac{x^k-x^{k-1}}{f(x^k)-f(x^{k-1})}f(x^k), \;k=1,2,\ldots

Метод секущих является двухшаговым, то есть новое приближение x^(k+1) определяется двумя предыдущими итерациями x^k и x^(k-1). В методе необходимо задавать два начальных приближения.

Но чтобы перейти от просто решения нелинейного уравнения, то есть нахождения точек пересечения f(x,y) с осью OX, нам надо воспользоваться матрицей якоби (Рисунок 1.).



1. Результат работы программы

Для этого метод секущих для рассчёта f(x), мы усовершенствуем методом Бройдена, его идея заключается заключается в аппроксимации матрицы Якоби с использованием уже вычисленных значений функций, образующих систему.

# Алгоритмическое обеспечение

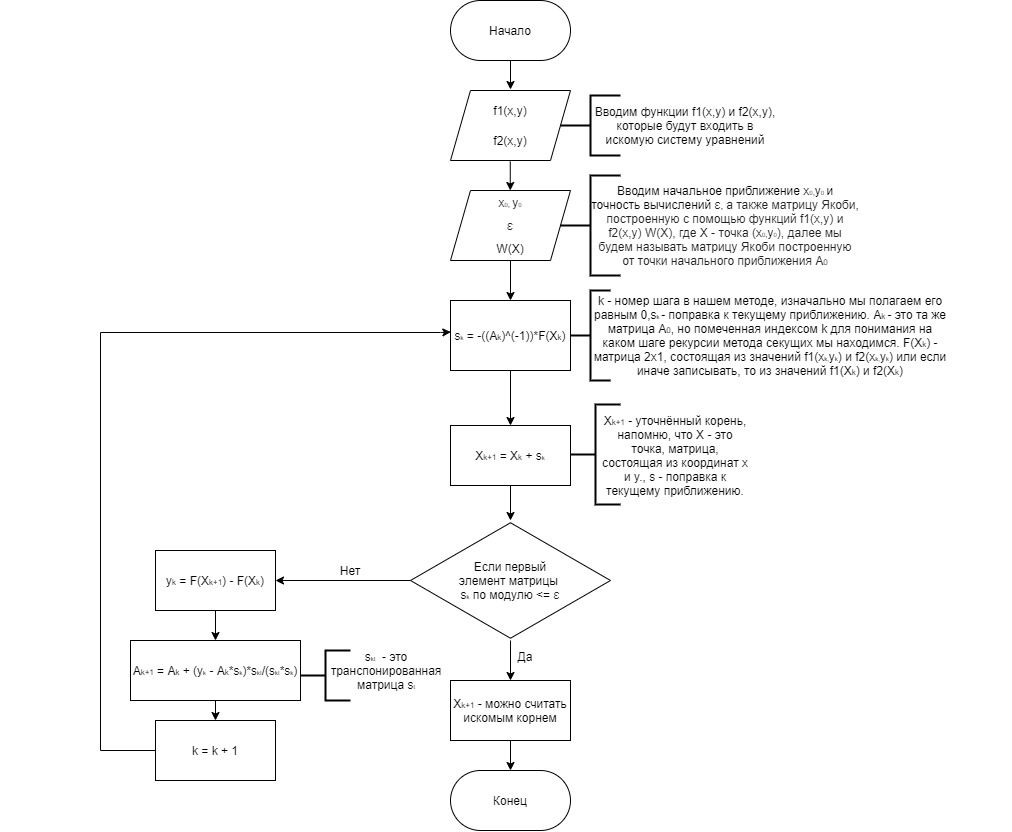
Алгоритм метода секущих для нелинейных систем:

1. Задать начальное приближение искомой точки x0,y0 и малое положительное число е, которое будет обозначать точность наших вычислений, далее для простоты матрицу 2x1 включающую значения x и y мы будем называть z.
2. Положить k=0 и A=W(z), где W(z) — матрица Якоби.
3. Решить систему линейных алгебраических уравнений

A\*s=-F(z) относительно s — поправки к текущему приближению.

1. Вычислить z1= z + s.
2. Если первый элемент |s|<=e, процесс завершить и считать решением значение x,y из z1, иначе вычислить y=F(z1)-F(z), A1 = A + (y-A\*s)\*St/(St\*s). St- транспонированная матрица S. И далее вернуться к пункту 3.

Данный алгоритм вы можете увидеть в виде блок-схемы на рисунке 2



1. Блок-схема алгоритма

# Программная реализация

Основные компоненты программы реализованы следующим образом:

1. Реализация метода секущих:

**public** **void** Secant\_Method(**string** formula1, **string** formula2, **double** x, **double** y,**double** eps, **double**[,] W)

{

**double**[,] invM = InverseMatrix(W);

// Строим обратную матрицу

**double**[] F0 = **new** **double**[2];

F0 [0] = F(x, y, formula1);

F0 [1] = F(x, y, formula2);

//Значение функций, используя точки начального приближения

**double**[] s0 = **new** **double**[2];

s0[0] = -(F0[0] \* invM[0, 0] + F0[1] \* invM[0, 1]);

s0[1] = -(F0[0] \* invM[1, 0] + F0[1] \* invM[1, 1]);

//s0

**double** x1 = x + s0[0];

**double** y1 = y + s0[1];

//Уточнённые координаты искомой точки.

**double**[] F1 = **new** **double**[2];

F1[0] = F(x1, y1, formula1);

F1[1] = F(x1, y1, formula2);

//Значение функций, используя точки x1,y1

**double**[] Y = **new** **double**[2];

Y[0] = F1[0] - F0[0];

Y[1] = F1[1] - F0[1];

// y0

**double**[] y0\_a0s0 = **new** **double** [2] ;

y0\_a0s0[0] = Y[0] - (W[0, 0] \* s0[0] + W[0, 1] \* s0[1]);

y0\_a0s0[1] = Y[1] - (W[1, 0] \* s0[0] + W[1, 1] \* s0[1]);

// y0-a0\*s0

**double** s0ts0 = Math.Pow(s0[0], 2) + Math.Pow(s0[1],2);

// s0\*s0t

**double**[,] y0\_a0s0\_st0 = **new** **double**[2,2];

y0\_a0s0\_st0[0, 0] = y0\_a0s0[0] \* s0[0];

y0\_a0s0\_st0[0, 1] = y0\_a0s0[0] \* s0[1];

y0\_a0s0\_st0[1, 0] = y0\_a0s0[1] \* s0[0];

y0\_a0s0\_st0[1, 1] = y0\_a0s0[1] \* s0[1];

//y0-a0\*s0-s0t

y0\_a0s0\_st0 = MatrixDecrease(y0\_a0s0\_st0, s0ts0);

// matrix\*(-1)

**double**[,] A = MatrixAddition(y0\_a0s0\_st0,W);

// Сложение матриц

**if** (Math.Abs(s0[0]) > eps)

{

result\_x.Text = String.Format("{0:F3}", x1);

result\_y.Text = String.Format("{0:F3}", y1);

f1\_txt.Text = String.Format("{0:F3}", F1[0]);

f2\_txt.Text = String.Format("{0:F3}", F1[1]);

Secant\_Method(formula1, formula2, x1, y1, eps, A);

//Рекурсия

}

}

1. Функции с помощью которых производятся действия над матрицей:

**public** **double**[,] MatrixAddition (**double**[,] W, **double**[,] A)

{

W[0, 0] += A[0, 0];

W[0, 1] += A[0, 1];

W[1, 0] += A[1, 0];

W[1, 1] += A[1, 1];

**return** W;

}

**public** **double**[,] MatrixDecrease(**double**[,] W, **double** d)

{

W[0, 0] = W[0, 0]/d;

W[1, 1] = W[1, 1]/d;

W[1, 0] = W[1, 0]/d;

W[0, 1] = W[0, 1]/d;

**return** W;

}

**public** **double**[,] MatrixNegation(**double**[,] W)

{

W[0, 0] = -W[0, 0];

W[1, 1] = -W[1, 1];

W[1, 0] = -W[1, 0];

W[0, 1] = -W[0, 1];

**return** W;

}

**public** **double**[,] InverseMatrix(**double**[,] W)

{

**double** d = W[0, 0] \* W[1, 1] - W[0, 1] \* W[1, 0];

**double**[,] A = TransMatrix(W);

**double**[,] invM = **new** **double**[2,2];

invM[0, 0] = A[1, 1] / d;

invM[1, 1] = A[0, 0] / d;

invM[0, 1] = A[1, 0] / (-d);

invM[1, 0] = A[0, 1] / (-d);

**return** invM;

}

**public** **double**[,] TransMatrix(**double**[,] W)

{

**double**[,] T = **new** **double**[2, 2];

T[0, 0] = W[0, 0];

T[1, 1] = W[1, 1];

T[0, 1] = W[1, 0];

T[1, 0] = W[0, 1];

**return** T;

}

1. Реализация рассчёта значения функции взятой из строки:

**public** **double** F(**double** x, **double** y, **string** formula)

{

ExpressionContext context = **new** ExpressionContext();

context.Imports.AddType(**typeof**(Math));

context.Variables["x"] = x;

context.Variables["y"] = y;

IGenericExpression<**double**> eGeneric = context.CompileGeneric<**double**>(formula + "+x\*0");

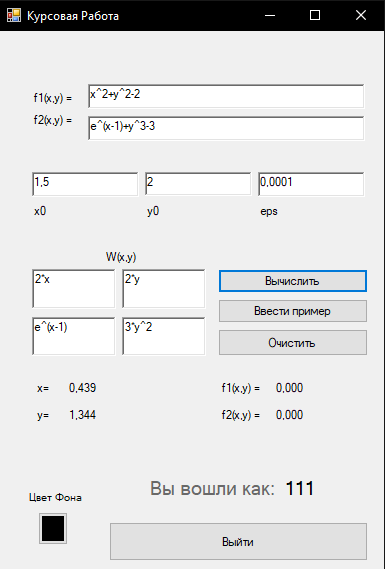
**return** eGeneric.Evaluate();

}

Полный код программы находится в приложениях.

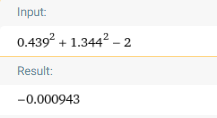
# Тестирование

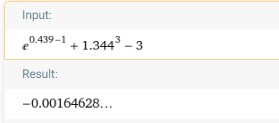
Воспользовавшись одним из заранее встроенных примеров и изменив его, мы заполнили поля программы и решили систему уравнений (Рисунок 3).



1. Результат работы программы

Для того, чтобы проверить правильность вычислений программы воспользуемся сервисом wolfram alpha (Рисунок 4).





1. Результат вычислений сервиса wolfram alpha

С помощью сервиса Wolfram alpha мы подтвердили правильность работы нашей программы.

# Инструкция программисту

1. Содержимое программы

После установки программы в её папке будет содержаться такие файлы: WindowsFormsApp6.exe – исполнительный файл, который запускает программу, unins000.exe и unins000.dat с помощью которых можно удалить данную программу и Flee.Net45.dll – внешняя библитека, позволяющая использовать формулу получаемую из строки для получения значения функции при заданных переменных.

1. Условия применения программы

Для работы программы необходимо наличие на компьютере библиотек .Net Framework 4 версии и выше. Компьютер должен иметь следующие минимальные системные требования:

* 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц или выше.
* 1 ГБ (для 32-разрядного процессора) или 2 ГБ (для 64-разрядного процессора) ОЗУ.
* Графическое устройство DirectX 9 с драйвером WDDM 1.0 или более поздней версии.

1. Характеристики программы

Установленная программа занимает на диске 112 КБ. Время работы различных функций программы при количестве переменных, равном 4, не превышают 1 секунды.

1. Обращение к программе

Программу можно запустить как с помощью ярлыка на рабочем столе, так и с помощью исполнительного файла KursovayaProg.exe, находящегося в папке программы.

1. Сообщения

При работе программы могут выводиться следующие сообщения об ошибках:

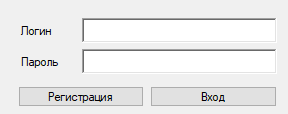
* Неверный логин и/или пароль!
* Данный пользователь уже зарегистрирован!

1. Удаление программы

Для удаления программы необходимо зайти в панель управления и выбрать пункт «Удаление программы». Далее из списка программ необходимо выбрать необходимую и нажать кнопку «Удалить». Далее возникнет окно, где необходимо подтвердить удаление программы.

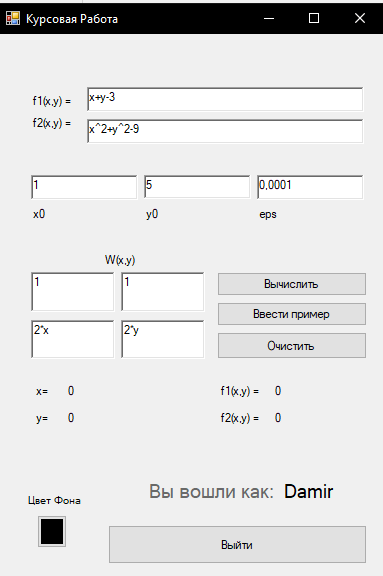
# Инструкция пользователю

При запуске программы возникнут текстовые поля авторизации (Рисунок 2), куда нужно ввести логин и пароль. После ввода данных нужно нажать на кнопку «Вход» или «Зарегистрироваться». Остальные элементы управления при этом будут недоступны. При вводе неверной пары логин-пароль возникнет новое окно, где пользователю сообщат об этом. При попытке регистрации уже существующего пользователя, также возникнет соответствующее окно (Рисунок 5).



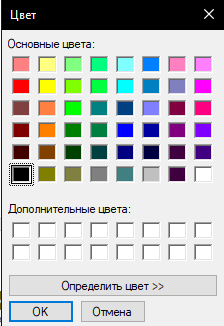
1. Поля авторизации

После входа поля авторизации скроются, покажется кнопка выхода и разблокируются кнопки: «Вычислить», «Ввести пример», «Очистить» необходимые для работы программы (Рисунок 6).



1. Основное окно программы

В программе предусмотрена возможность выбора цвета фона . Для этого необходимо нажать на кнопку, находящуюся под надписью «Цвет Фона». После этого появится окно (Рисунок 7), в котором имеется возможность выбрать новый цвет.

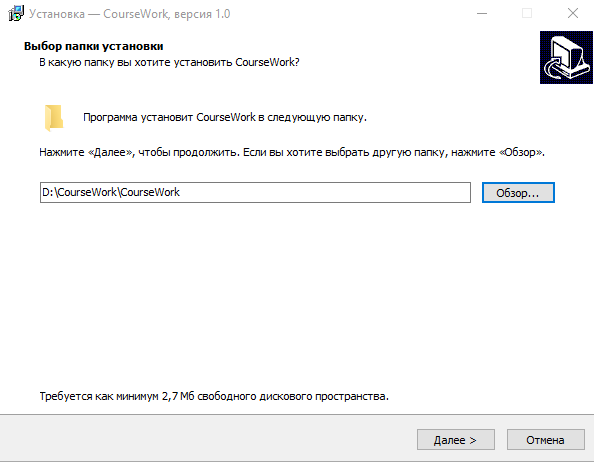


1. Окно выбора цвета

Для введения своих функций воспользуйтесь данным алгоритмом:

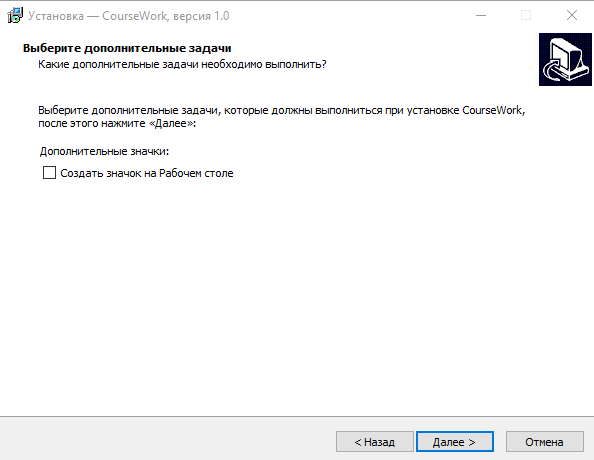
1. Необходимо ввести 2 функции f(x,y) ), которые будут входить в нашу систему уравнений. Поля в которые необходимо вводить функции находятся справа от надписей «f1(x,y)» и «f2(x,y)».
2. Далее необходимо ввести начальное приближение x0,y0 и e –точность вычислений. Поля для ввода находятся ниже полей для ввода функций.
3. Далее вам необходимо разбить первую функцию, как вам будет удобно и продифференцировать одну её часть по dx и записать в левое верхнее поле матрицы W(x,y), а вторую часть по dy и записать уже в правое верхнее, тоже самое проделать со второй функцией, но записывать надо уже в нижние поля.
4. Далее нажмите кнопку «Вычислить» и вы увидите точку пересечение первой и второй функции(решение системы уравнений).
5. Инструкция по установке

Для установки программы необходимо запустить инсталлятор, который называется “Setup\_KP”. При его запуске возникнет следующее окно (Рисунок 8):



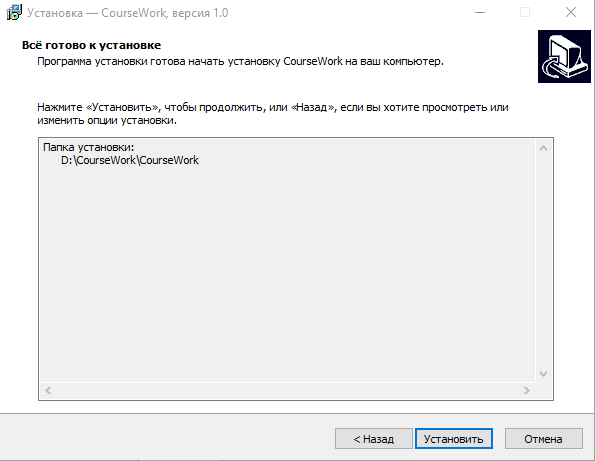
1. Окно, возникающее при запуске инсталлятора

На данном этапе есть возможность отменить установку с помощью кнопки «Отмена». В текущем окне у вас есть возможность выбрать папку для расположения программы. Для дальнейшей установки необходимо нажать кнопку «Далее», чтобы перейти к окну, которое позволит вам выбрать создавать или не создавать ярлык на рабочем столе (Рисунок 9):



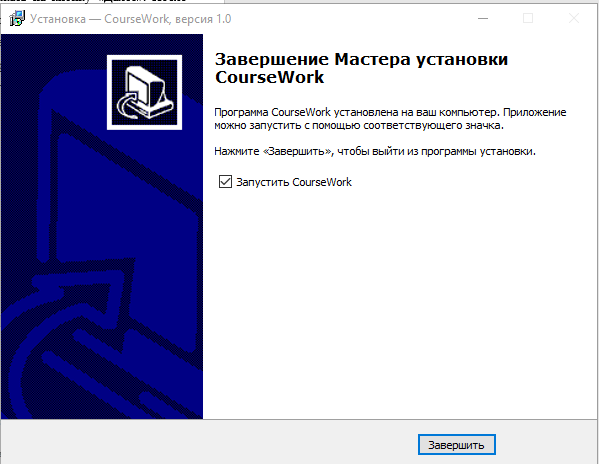
1. Окно выбора папки для программы

На данном этапе есть возможность отменить установку с помощью кнопки «Отмена» или вернуться к прошлому окну с помощью кнопки «Назад». Для дальнейшей установки, после выбора всех параметров, нужно нажать на кнопку «Далее». После этого возникнет окно, требующее подтверждения установки с выбранными ранее параметрами (Рисунок 10):



1. Окно подтверждения установки

На данном этапе есть возможность отменить установку с помощью кнопки «Отмена» или вернуться к прошлому окну с помощью кнопки «Назад». Для дальнейшей установки необходимо нажать на кнопку «Далее». После этого операционная система может вывести сообщение, где инсталлятору необходимо дать разрешение на изменение содержимого диска. После разрешения инсталлятор установит программу по указанному пути и возникнет окно, сообщающее об успешной установке (Рисунок 11):



1. Окно, сообщающее об успешной установке программы

Теперь нужно нажать на кнопку «Закрыть», чтобы закрыть это окно. После установки на рабочем столе возникнет ярлык «Курсовая по программированию», при нажатии на который запустится установленная программа.

Заключение

В результате выполнения курсовой работы все поставленные цели были достигнуты, задачи решены в полном объёме, профессиональные навыки по разработке и отладке на языке высокого уровня программных средств были приобретены. Также на практике освоены основные принципы создания программных средств и технологии программирования на языке высокого уровня с использованием системных вызовов и библиотек стандартных подпрограмм.

За время выполнения курсовой работы были реализованы: интерфейс пользователя, система авторизации, инсталляционный пакет, функциональная часть.

Список использованных источников

1. Metanit, Руководство по программированию в Windows Forms [Электронный ресурс]. - URL:

<https://www.metanit.com/sharp/windowsforms/>

1. Machine Learning, Метод секущих [Электронный ресурс]. - URL: <http://www.machinelearning.ru/wiki/index.php?title=Метод_секущих>
2. Методы решения систем нелинейных уравнений [Электронный ресурс]. – URL: <http://mathhelpplanet.com/static.php?p=metody-resheniya-sistem-nelineynykh-uravneniy>
3. Решение нелинейных уравнений и систем нелинейных уравнений [Электронный ресурс]. – URL: <https://mainfodotru.files.wordpress.com/2017/09/numeric-methods-part2.pdf>
4. Wolfram Alpha [Электронный ресурс]. – URL: <https://www.wolframalpha.com>
5. Приложения

**Приложение 1**

**Исходный код файла CourseWork\_Form.cs**

**using** Flee.PublicTypes;

**using** System;

**using** System.Windows.Forms;

**using** System.IO;

**namespace** WindowsFormsApp6

{

**public** partial **class** CourseWork\_Form : Form

{

**public** CourseWork\_Form()

{

InitializeComponent();

}

**private** **void** Calc\_button\_Click(**object** sender, EventArgs e)

{

**double** x = Convert.ToDouble(x\_start.Text);

**double** y = Convert.ToDouble(y\_start.Text);

**double** epsel = Convert.ToDouble(form\_eps.Text);

**string** formulaX = formulaX\_txt.Text;

**string** formulaY = formulaY\_txt.Text;

//Создание переменных.

**double**[,] W = **new** **double**[2, 2];

W[0, 0] = F(x, y, txt00.Text);

W[0, 1] = F(x, y, txt01.Text);

W[1, 0] = F(x, y, txt10.Text);

W[1, 1] = F(x, y, txt11.Text);

//Заполнение матрицы якоби.

Secant\_Method(formulaX, formulaY, x, y, epsel, W);

// Вызов Метода Секущих.

}

**public** **void** Secant\_Method(**string** formula1, **string** formula2, **double** x, **double** y,**double** eps, **double**[,] W)

{

**double**[,] invM = InverseMatrix(W);

// Строим обратную матрицу

**double**[] F0 = **new** **double**[2];

F0 [0] = F(x, y, formula1);

F0 [1] = F(x, y, formula2);

//Значение функций, используя точки начального приближения

**double**[] s0 = **new** **double**[2];

s0[0] = -(F0[0] \* invM[0, 0] + F0[1] \* invM[0, 1]);

s0[1] = -(F0[0] \* invM[1, 0] + F0[1] \* invM[1, 1]);

//s0

**double** x1 = x + s0[0];

**double** y1 = y + s0[1];

//Уточнённые координаты искомой точки.

**double**[] F1 = **new** **double**[2];

F1[0] = F(x1, y1, formula1);

F1[1] = F(x1, y1, formula2);

//Значение функций, используя точки x1,y1

**double**[] Y = **new** **double**[2];

Y[0] = F1[0] - F0[0];

Y[1] = F1[1] - F0[1];

// y0

**double**[] y0\_a0s0 = **new** **double** [2] ;

y0\_a0s0[0] = Y[0] - (W[0, 0] \* s0[0] + W[0, 1] \* s0[1]);

y0\_a0s0[1] = Y[1] - (W[1, 0] \* s0[0] + W[1, 1] \* s0[1]);

// y0-a0\*s0

**double** s0ts0 = Math.Pow(s0[0], 2) + Math.Pow(s0[1],2);

// s0\*s0t

**double**[,] y0\_a0s0\_st0 = **new** **double**[2,2];

y0\_a0s0\_st0[0, 0] = y0\_a0s0[0] \* s0[0];

y0\_a0s0\_st0[0, 1] = y0\_a0s0[0] \* s0[1];

y0\_a0s0\_st0[1, 0] = y0\_a0s0[1] \* s0[0];

y0\_a0s0\_st0[1, 1] = y0\_a0s0[1] \* s0[1];

//y0-a0\*s0-s0t

y0\_a0s0\_st0 = MatrixDecrease(y0\_a0s0\_st0, s0ts0);

// matrix\*(-1)

**double**[,] A = MatrixAddition(y0\_a0s0\_st0,W);

// Сложение матриц

**if** (Math.Abs(s0[0]) > eps)

{

result\_x.Text = String.Format("{0:F3}", x1);

result\_y.Text = String.Format("{0:F3}", y1);

f1\_txt.Text = String.Format("{0:F3}", F1[0]);

f2\_txt.Text = String.Format("{0:F3}", F1[1]);

Secant\_Method(formula1, formula2, x1, y1, eps, A);

//Рекурсия

}

}

**public** **double**[,] MatrixAddition (**double**[,] W, **double**[,] A)

{

W[0, 0] += A[0, 0];

W[0, 1] += A[0, 1];

W[1, 0] += A[1, 0];

W[1, 1] += A[1, 1];

**return** W;

}

**public** **double**[,] MatrixDecrease(**double**[,] W, **double** d)

{

W[0, 0] = W[0, 0]/d;

W[1, 1] = W[1, 1]/d;

W[1, 0] = W[1, 0]/d;

W[0, 1] = W[0, 1]/d;

**return** W;

}

**public** **double**[,] MatrixNegation(**double**[,] W)

{

W[0, 0] = -W[0, 0];

W[1, 1] = -W[1, 1];

W[1, 0] = -W[1, 0];

W[0, 1] = -W[0, 1];

**return** W;

}

**public** **double**[,] InverseMatrix(**double**[,] W)

{

**double** d = W[0, 0] \* W[1, 1] - W[0, 1] \* W[1, 0];

**double**[,] A = TransMatrix(W);

**double**[,] invM = **new** **double**[2,2];

invM[0, 0] = A[1, 1] / d;

invM[1, 1] = A[0, 0] / d;

invM[0, 1] = A[1, 0] / (-d);

invM[1, 0] = A[0, 1] / (-d);

**return** invM;

}

**public** **double**[,] TransMatrix(**double**[,] W)

{

**double**[,] T = **new** **double**[2, 2];

T[0, 0] = W[0, 0];

T[1, 1] = W[1, 1];

T[0, 1] = W[1, 0];

T[1, 0] = W[0, 1];

**return** T;

}

//Действия с матрицами.

**public** **double** F(**double** x, **double** y, **string** formula)

{

ExpressionContext context = **new** ExpressionContext();

context.Imports.AddType(**typeof**(Math));

context.Variables["x"] = x;

context.Variables["y"] = y;

IGenericExpression<**double**> eGeneric = context.CompileGeneric<**double**>(formula + "+x\*0");

**return** eGeneric.Evaluate();

}

//Метод рассчёта функции, используя строку, как формулу по которой будет произведён рассчёт.

**private** **void** Clear\_button\_Click(**object** sender, EventArgs e)

{

formulaX\_txt.Text = "";

formulaY\_txt.Text = "";

x\_start.Text = "";

y\_start.Text = "";

form\_eps.Text = "";

txt00.Text = "";

txt01.Text = "";

txt10.Text = "";

txt11.Text = "";

result\_x.Text = "0";

result\_y.Text = "0";

f1\_txt.Text = "0";

f2\_txt.Text = "0";

}

**private** **void** ToolStripMenuItem3\_Click(**object** sender, EventArgs e)

{

formulaX\_txt.Text = "cos(x)+y^2-3";

formulaY\_txt.Text = "x^2\*2\*y+y^2-9";

txt00.Text = "-sin(x)";

txt01.Text = "2\*y";

txt10.Text = "4\*x\*y";

txt11.Text = "2\*y";

x\_start.Text = "1";

y\_start.Text = "1";

form\_eps.Text = "0,0001";

}

**private** **void** ToolStripMenuItem5\_Click(**object** sender, EventArgs e)

{

formulaX\_txt.Text = "x+y-3";

formulaY\_txt.Text = "x^2+y^2-9";

txt00.Text = "1";

txt01.Text = "1";

txt10.Text = "2\*x";

txt11.Text = "2\*y";

x\_start.Text = "1";

y\_start.Text = "5";

form\_eps.Text = "0,001";

}

**private** **void** ToolStripMenuItem4\_Click(**object** sender, EventArgs e)

{

formulaX\_txt.Text = "x^2+y^2-2";

formulaY\_txt.Text = "e^(x-1)+y^3-2";

txt00.Text = "2\*x";

txt01.Text = "2\*y";

txt10.Text = "e^(x-1)";

txt11.Text = "3\*y^2";

x\_start.Text = "1,5";

y\_start.Text = "2";

form\_eps.Text = "0,01";

}

**private** **void** Example\_button\_Click(**object** sender, EventArgs e)

{

contextMenuExample.Show(**this**.Location.X + 415, **this**.Location.Y + 309);

}

**private** **void** Choose\_color\_Click(**object** sender, EventArgs e)

{

bg\_color\_dialog.ShowDialog();

**this**.BackColor = bg\_color\_dialog.Color;

}

//Начало системы авторизации

**private** **void** Registration\_button\_Click(**object** sender, EventArgs e)

{

**using** (StreamWriter sw = **new** StreamWriter("log.txt", **true**, System.Text.Encoding.Default)){}

**if** (Login\_field.Text != "" && Password\_field.Text != "")

{

**bool** check = **true**;

**using** (StreamReader sr = **new** StreamReader("log.txt", System.Text.Encoding.Default))

{

**string** line;

**while** ((line = sr.ReadLine()) != **null**)

{

**string**[] words = line.Split(' ');

**if** (words[0] == Login\_field.Text)

{

MessageBox.Show("Такой пользователь уже зарегистрирован");

check = **false**;

}

}

}

**using** (StreamWriter sw = **new** StreamWriter("log.txt", **true**, System.Text.Encoding.Default))

{

**if**(check != **false**)

sw.Write(Login\_field.Text + " " + Password\_field.Text + "\n");

}

}

}

**private** **void** Login\_button\_Click(**object** sender, EventArgs e)

{

**bool** check = **false**;

**if** (Login\_field.Text != "" && Password\_field.Text != "")

{

**using** (StreamReader sr = **new** StreamReader("log.txt", System.Text.Encoding.Default))

{

**string** line;

**while** ((line = sr.ReadLine()) != **null**)

{

**string**[] words = line.Split(' ');

**if** (words[0] == Login\_field.Text && words[1] == Password\_field.Text)

{

calc\_button.Enabled = **true**;

example\_button.Enabled = **true**;

clear\_button.Enabled = **true**;

Name\_label.Text = Login\_field.Text;

Login\_field.Text = "";

Password\_field.Text = "";

Login\_field.Visible = **false**;

Password\_field.Visible = **false**;

Login\_button.Visible = **false**;

Registration\_button.Visible = **false**;

Exit\_button.Visible = **true**;

Login\_label.Visible = **false**;

Password\_label.Visible = **false**;

Name\_label.Visible = **true**;

Name\_label\_pre.Visible = **true**;

check = **true**;

}

}

**if** (check != **true**)

MessageBox.Show("Неверный логин и/или пароль");

}

}

}

**private** **void** Exit\_button\_Click(**object** sender, EventArgs e)

{

calc\_button.Enabled = **false**;

example\_button.Enabled = **false**;

clear\_button.Enabled = **false**;

Login\_button.Visible = **true**;

Registration\_button.Visible = **true**;

Exit\_button.Visible = **false**;

Login\_field.Visible = **true**;

Password\_field.Visible = **true**;

Login\_label.Visible = **true**;

Password\_label.Visible = **true**;

Name\_label.Visible = **false**;

Name\_label\_pre.Visible = **false**;

Name\_label.Text = "User";

}

**private** **void** Login\_field\_KeyPress(**object** sender, KeyPressEventArgs e)

{

**if** (!((e.KeyChar >= 97 && e.KeyChar <= 122) || (e.KeyChar >= 65 && e.KeyChar <= 90) || (e.KeyChar >= 48 && e.KeyChar <= 57) || e.KeyChar == 8))

{

e.Handled = **true**;

}

}

**private** **void** Password\_field\_KeyPress(**object** sender, KeyPressEventArgs e)

{

**if** (!((e.KeyChar >= 97 && e.KeyChar <= 122) || (e.KeyChar >= 65 && e.KeyChar <= 90) || (e.KeyChar >= 48 && e.KeyChar <= 57) || e.KeyChar == 8))

{

e.Handled = **true**;

}

}

//Ограничение возможных символов для ввода.

//Конец системы авторизации.

}

}