Министерство образования и науки Кыргызской Республики

Кыргызский Государственный Технический Университет

им. И. Раззакова

Кафедра программного обеспечения компьютерных систем

БАКАЛАВРСКАЯ УЧЕБНАЯ ПРОГРАММА

НАПРАВЛЕНИЕ—710400 «ПРОГРАММНАЯ ИНЖЕНЕРИЯ»

Дисциплина «Методы оптимизации»

**Отчет по практической работе №4**

**«Разработка ПО для поиска решения одномерной задачи оптимизации**

**на основе *Newton’s* method of optimization – *Метод Ньютона*»**

по дисциплине

«МЕТОДЫ ОПТИМИЗАЦИИ»

Выполнил: Жанболот уулу Аскабек

Группа: ПИ-2-20

Проверил: Тен И.Г.

Бишкек 2022

**Практическая работа №4**

«Разработка ПО для поиска решения ***одномерной задачи оптимизации***

на основе ***Newton’s*** method of optimization – ***Метод Ньютона***»

**Что дано:**

* Задается аналитическое выражение для произвольной целевой функции f(x);
* ***Спецификация м***етода оптимизации (***Newton’s method***) для нахождения оптимального значения решающей переменной;
* Структура интерфейсной формы системы поиска решения задачи оптимизации аналогична Практической работе №1, но с учетом особенностей изучаемого метода оптимизации — поиска экстремума целевой функции, реализующая итерационный метод — ***Newton’s method***.

**Что требуется:**

* Разработать ***проект*** ПО для поиска ***решения задач оптимизации*** для произвольной заданной допустимой погрешности;
* Разработать ***код*** ПО для поиска ***решения задач оптимизации*** для произвольной заданной допустимой погрешности;
* Доказать, что найдено ***оптимальное*** решение с погрешностью решения не более заданной допустимой погрешности.

**Раздел №1**

**Наименование работы –** поиск решения одномерной задачи оптимизации методом Ньютона (NM)

**Раздел №2**

СПЕЦИФИКАЦИЯ ПРОБЛЕМЫ №1: Поиск решения одномерной задачи оптимизации методом Ньютона.

Найти решение задачи оптимизации для произвольной заданной допустимой погрешности. Итерационный метод для решения оптимизационных задач, который позволяет определить экстремум целевой функции. При поиске экстремума целевой функции используется производная первого и второго порядка функции.

**Раздел №3**

**3.1: Описание метода**

To solve the one-dimensional optimization problem  
   
using Newton’s Method and given single an initial approximate x(0) of the searching variable value with a required error tolerance *tol*.

**3.2: Mathematical description of the algorithm:  
 **

For all k=0,1,2,…;

Where k – is a number of iteration; x(k) – is an approximate value of the searching variable x on the k-th iteration; f**’**(x(k)–is a prime derivative value of the objective function at point x(k); f**’’**(x(k)–is a second derivative value of the objective function at point x(k).

**3.3: Flow-chart of the NEWTON’S METHOD to find out extremum of an objective function**





**3.4: Computational description of the algorithm:**

**INPUT** {f(X); X0; Epsilon; Delta; R; k\_max}

**Body of algorithm**

/\* \*/

/\* \*/

Cond: = 0; /\* Condition for loop termination \*/

FX0: = f(X0); /\* Compute the objective function value at given point X0\*/

DFX0: = f ‘(X0); /\* Compute the first derivative value of the objective function at given point X0\*/

DDFX0: = f ‘’(X0); /\* Compute the second derivative value of the objective function at given point X0\*/

|DO FOR K: = 1 TO Max UNTIL Cond≠0;

| DDFX0: = f ‘’(X0); /\* Second derivative of the objective function at given point \*/

| |IF abs( DDFX0)<=Tolerance THEN

| | Cond:=1; QND: =0; /\* QND stands for Quotient is Numerator/Denominator \*/

| | ELSE DP:= DFX0/DDFX0; /\* Cond1:=1 stands for that Denominator Value is zero \*/

| |IF K=1 THEN DP0:= DP;

| |ENDIF

| |IF SIGN(DP0)=SIGN(DP) /\* Handle an Auto-Oscillation \*/

| | THEN X1:= X0 – DP; /\* If an Oscillation is absent \*/

| | ELSE X1:= X0 – DP/R; /\* If an Oscillation is present \*/

| |ENDIF

| | DP0:=DP;

| | FX1:=f(X1); /\* Objective function value at new point X1\*/

| | DFX1:=f ‘(X1); /\* First derivative of objective function value at new point X1\*/

| | RelError:= 2\*ABS(DP)/(ABS(X1) + Tolerance); /\* Relative error \*/

| | |IF RelError < Delta THEN /\* Check of convergence of the sequence \*/

| | | |If Cond ≠ 1 THEN Cond:= 2;

| | | |ENDIF

| | |ENDIF

| | X0:= X1; DFX0:= DFX1; /\* Update values \*/

| |ENDIF

|ENDDO

**OUTPUT**

PRINT ‘The optimum solution X\* equal’ X1

PRINT ‘The optimum solution was found with the desired tolerance’ Epsilon

PRINT ‘The minimum of objective function f(x\*) is’ FX1

PRINT ‘The value of first derivative f ’(x) is‘ DFX1

/\* *Report about searching process state is given from program \*/*

IF Cond = 0 THEN

PRINT “The maximum number of iterations was exceeded”

IF Cond = 1 THEN

PRINT “Division by zero was encountered”

IF Cond = 2 THEN

PRINT “The solution was found with the desired tolerance”

**Раздел № 4 Стадия проектирования системы для поиска решения одномерной задачи оптимизации методом Ньютона**

**4.1: Функциональные требования:**

1) Программа должна производить поиск экстремума произвольной нелинейной функции, с заданной погрешностью.

2) В программе должно быть предоставлено доказательство валидности найденного решения.

3) Программа должна определять траекторию к определенному excel-file “MO\_LookingForOnePoint.xlsx”.

4) Программа должна открывать лист “Russian” excel файла “MO\_LookingForOnePoint.xlsx”, и вставлять целевую функцию, левую и правую границы в определенные ячейки листа.

5) Программа должна вычислять значение целевой функции f(x), для любого значения аргумента x.

6) В программе должна производиться проверка валидности введеных данных.

7) В программе должна быть реализована Newton’sMethod для поиска экстремума функции.

**Раздел №5**

**ПРОЕКТИРОВАНИЕ По**: Документирование этапов проектирования интерфейсной формы системы, реализующей метод Ньютона – для поиска решения одномерной задачи оптимизации

**5.1: Документирование процесса задания свойств элементов интерфейсной формы системы, выбранной для внедрения и реализующей Newton’s Method**

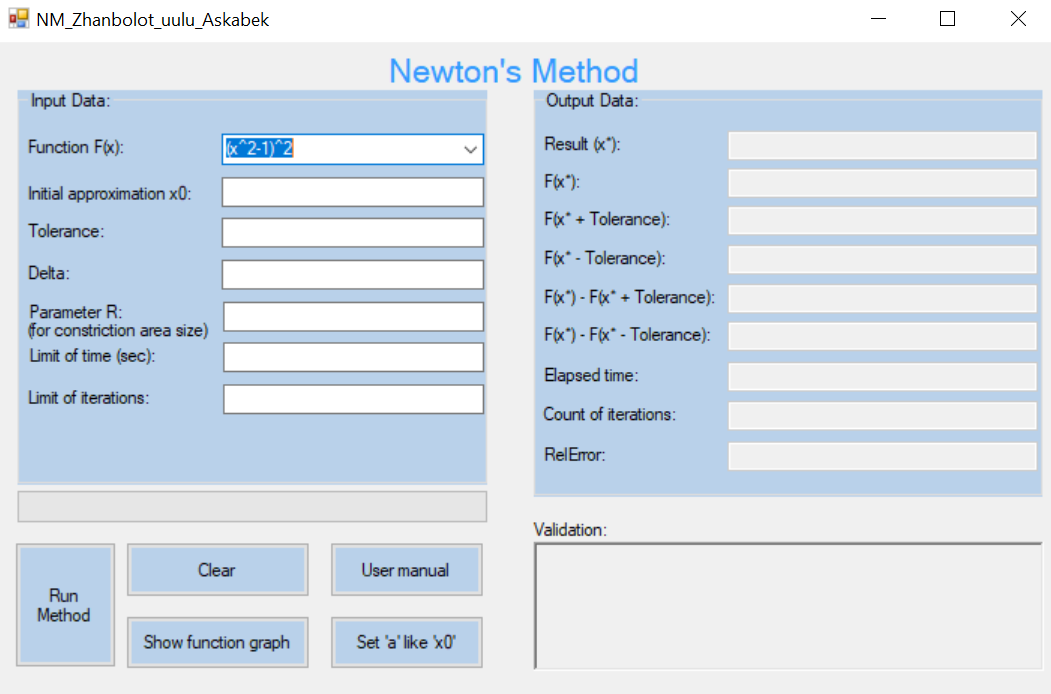


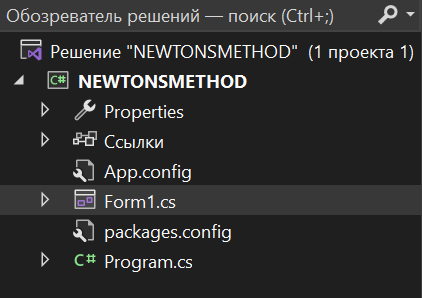
Table 1: Settings for control properties of the software system implementing Newton’s method (the table is intended to document the activity of a student to settings control properties of the interface form – the table is ***mandatory***)

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of control** | **Control** | **Property** | **Setting** |
| 1 | Label1 | Appearance (Text) | Newton’s Method |
| Design (Name) | label1 |
| 2 | GroupBox1 | Appearance (Text) | Input Data |
| Design (Name) | groupBox1 |
| 3 | Label2 | Appearance (Text) | Function F(x) |
| Design (Name) | label2 |
| 4 | ComboBox1 | Appearance (Text) | (x^2-1)^2 |
| The rest (Items) | Collection |
| Items (Collection) | (x-1)\*(x-2)^2  x^2-4\*sin(x)  x^3-5\*x^2+x+5  (x-2)^2-ln(x)  (x-2)^2-log(x) |
| Design (Name) | Function |
| 5 | Label3 | Appearance (Text) | Initial approximation x0 |
| Design (Name) | label3 |
| 6 | TextBox1 | Appearance (Text) |  |
| Design (Name) | InitialApproximation |
| 7 | Label4 | Appearance (Text) | Tolerance |
| Design (Name) | label4 |
| 8 | TextBox2 | Appearance (Text) |  |
| Design (Name) | Tolerance |
| 9 | Label5 | Appearance (Text) | Limit of time: |
| Design (Name) | label6 |
| 10 | TextBox3 | Appearance (Text) |  |
| Design (Name) | limitOfTime |
| 11 | Label6 | Appearance (Text) | Limit of iterations: |
| Design (Name) | label5 |
| 12 | TextBox4 | Appearance (Text) |  |
| Design (Name) | limitOfIterations |
| 13 | ProgressBar1 | Behavior (Visible) | true |
| Design (Name) | progressBar1 |
| 14 | GroupBox2 | Appearance (Text) | Output Data |
| Design (Name) | groupBox2 |
| 15 | Button1 | Appearance (Text) | User manual |
| Design (Name) | button5 |
| 16 | Button2 | Appearance (Text) | Run Method |
| Design (Name) | button1 |
| 17 | Button3 | Appearance (Text) | Show function graph |
| Design (Name) | FunctionGraph |
| 18 | Button4 | Appearance (Text) | Clear |
| Design (Name) | button3 |
| 19 | Button5 | Appearance (Text) | Set ‘a’ like ‘x0’ |
| Design (Name) | button2 |
| 20 | GroupBox3 | Appearance (Text) | Output Data |
| Design (Name) | groupBox2 |
| 21 | Label7 | Appearance (Text) | Result (x\*): |
| Design (Name) | label12 |
| 22 | TextBox5/ Behavior  (ReadOnly) | ReadOnly | true |
| Design (Name) | ResultX |
| 23 | Label8 | Appearance (Text) | Elapsed time: |
| Design (Name) | label9 |
| 24 | TextBox6/ Behavior  (ReadOnly) | ReadOnly | true |
| Design (Name) | elapsedTime |
| 25 | Label9 | Appearance (Text) | Count of iterations: |
| Design (Name) | label10 |
| 26 | TextBox7/ Behavior  (ReadOnly) | ReadOnly | true |
| Design (Name) | countofiterations |
| 27 | Label10 | Appearance (Text) | F(x\*): |
| Design (Name) | label13 |
| 28 | TextBox8/ Behavior  (ReadOnly) | ReadOnly | true |
| Design (Name) | fx |
| 29 | Label11 | Appearance (Text) | F(X\*+Tolerance): |
| Design (Name) | label16 |
| 30 | TextBox9/ Behavior  (ReadOnly) | ReadOnly | true |
| Design (Name) | fxplustolerance |
| 31 | Labe112 | Appearance (Text) | F(X\*-Tolerance): |
| Design (Name) | label15 |
| 32 | TextBox10/ Behavior  (ReadOnly) | ReadOnly | true |
| Design (Name) | fxminustolerance |
| 33 | Label13 | Appearance (Text) | Validation |
| Design (Name) | label17 |
|  |  |
| 34 | Label14 | Appearance (Text) | F(X\*)-F(X\*+Tolerance) |
| Design (Name) | label11 |
| 35 | TextBox13/ Behavior  (ReadOnly) | ReadOnly | true |
|  |  | Design (Name) | fxminusplustolerance |
| 36 | Label15 | Appearance (Text) | F(X\*)-F(X\*-Tolerance) |
|  |  | Design (Name) | label8 |
| 37 | TextBox12/ Behavior  (ReadOnly) | ReadOnly | true |
|  |  | Design (Name) | fxminusminustolerance |
| 38 | TextBox13/Behavior  (ReadOnly) | ReadOnly | true |
|  |  | Design (Name) | absError |
| 39 | Label16 | Appearance (Text) | RelError: |
|  |  | Design (Name) | label21 |

**Раздел №6**

**Стадии *конструирования* программного обеспечения для поиска решения одномерной задачи оптимизации, реализующей Newton’s Method:**

**6.1:** Код программы на C# Windows Forms(.NET Framework), ***ассоцированный с интерфейсной формой*** “ Form1.cs ”, который ***реализует функции ввода и вывода данных***, ***реализует логику*** Newton’s метода и составляет Public Class “Form1”



**6.2:** **Листинг программы**

using System;

using System.Text;

using Microsoft.Office.Interop.Excel;

using System.Windows.Forms;

using Application = Microsoft.Office.Interop.Excel.Application;

using TextBox = System.Windows.Forms.TextBox;

using MessageBox = System.Windows.Forms.MessageBox;

using aziretParser;

using System.Diagnostics;

using System.Drawing;

namespace POCKETSEARCHMETHOD

{

public partial class Form1 : Form

{

private const string nameOfExcel = @"\Zhanbolot\_uulu\_Askabek\_LookingForOnePoint.xlsm";

string inputFuncFX = "";

decimal df0, df1, ddf0, ddf1, dp, dp0, relerror;

decimal x0 = 0;

decimal x1 = 0;

decimal f0;

decimal f1;

decimal e\_tol = 0;

decimal delta = 0;

int k\_max = 0, cond, QND;

decimal t\_max = 0;

decimal parameterR = 0;

decimal fplusTol;

decimal fminusTol;

Application xls;

Workbook book = null;

Worksheet sheet = null;

public Form1()

{

InitializeComponent();

xls = new Application();

}

public int getSign(decimal number)

{

if (number < 0)

{

return -1;

}

else

{

return 1;

}

}

public void OpenExcel()

{

if (!checkFunction(1)) return;

string function;

decimal startPoint;

try

{

if (book == null)

{

book = xls.Workbooks.Open(System.IO.Directory.GetCurrentDirectory() + nameOfExcel);

}

if (sheet == null)

{

sheet = book.Sheets["Russian"];

sheet.Activate();

}

xls.Visible = true;

function = Function.Text;

if (InitialApproximation.Text != "" && InitialApproximation.Text != "-" && InitialApproximation.Text != "+" && InitialApproximation.Text != ".")

{

startPoint = Decimal.Parse(InitialApproximation.Text);

}

else

{

startPoint = 1;

}

sheet.Cells[4, 9] = startPoint;

sheet.Cells[2, 1] = "f(x)=" + Function.Text;

StringBuilder builder = new StringBuilder(function);

builder.Replace("exp", ":");

builder.Replace("x", "D4");

builder.Replace(":", "exp");

function = builder.ToString();

sheet.Range["E4:E10003"].Value = "=" + function;

}

catch

{

book = xls.Workbooks.Open(System.IO.Directory.GetCurrentDirectory() + nameOfExcel);

sheet = book.Sheets["Russian"];

sheet.Activate();

xls.Visible = true;

function = Function.Text;

if (InitialApproximation.Text != "" && InitialApproximation.Text != "-" && InitialApproximation.Text != "+" && InitialApproximation.Text != ".")

{

startPoint = Decimal.Parse(InitialApproximation.Text);

}

else

{

startPoint = 1;

}

sheet.Cells[4, 9] = startPoint;

sheet.Cells[2, 1] = "f(x)=" + Function.Text;

StringBuilder builder = new StringBuilder(function);

builder.Replace("exp", ":");

builder.Replace("x", "D4");

builder.Replace(":", "exp");

function = builder.ToString();

sheet.Range["E4:E10003"].Value = "=" + function;

}

}

private bool parseTry(TextBox t, String type)

{

try

{

if (type == "Decimal")

Decimal.Parse(t.Text, System.Globalization.NumberStyles.Float);

else if (type == "Integer")

int.Parse(t.Text);

return true;

}

catch

{

return false;

}

}

private void Clean(Control control)

{

foreach (var element in control.Controls)

{

switch (element.GetType().Name)

{

case "TextBox":

((TextBox)element).Text = String.Empty;

break;

case "RadioButton":

((RadioButton)element).Checked = false;

break;

case "RichTextBox":

((RichTextBox)element).Text = String.Empty;

break;

case "GroupBox":

Clean((Control)element);

break;

default:

break;

}

}

}

private bool IsOKForDecimalTextBox(char theCharacter, TextBox theTextBox, bool positive)

{

if (!char.IsControl(theCharacter) && !char.IsDigit(theCharacter) && (theCharacter != ',') && (theCharacter != '.')

&& (theCharacter != '-') && (theCharacter != '+') && (theCharacter != 'E') && (theCharacter != 'e'))

{

return false;

}

if(positive && theCharacter == '-' && (theTextBox.Text.IndexOf('E') == -1 && theTextBox.Text.IndexOf('e') == -1))

{

return false;

}

if (theCharacter == ',' && (theTextBox.Text.IndexOf(',') > -1 || theTextBox.Text.IndexOf('.') > -1))

{

return false;

}

if (theCharacter == '.' && (theTextBox.Text.IndexOf('.') > -1 || theTextBox.Text.IndexOf(',') > -1))

{

return false;

}

if (theCharacter == 'e' && (theTextBox.Text.IndexOf('e') > -1 || theTextBox.Text.IndexOf('E') > -1))

{

return false;

}

if (theCharacter == 'E' && (theTextBox.Text.IndexOf('E') > -1 || theTextBox.Text.IndexOf('e') > -1))

{

return false;

}

if (theCharacter == '-' && (theTextBox.Text.IndexOf('-') > -1 || theTextBox.Text.IndexOf('+') > -1))

{

return false;

}

if (theCharacter == '+' && (theTextBox.Text.IndexOf('+') > -1 || theTextBox.Text.IndexOf('-') > -1))

{

return false;

}

if (((theCharacter == '-') || (theCharacter == '+')) && (theTextBox.SelectionStart != 0 && (theTextBox.Text.IndexOf('E') == -1 && theTextBox.Text.IndexOf('e') == -1)))

{

return false;

}

if ((char.IsDigit(theCharacter) || (theCharacter == ',') || (theCharacter == '.')) && ((theTextBox.Text.IndexOf('-') > -1)

|| (theTextBox.Text.IndexOf('+') > -1)) && theTextBox.SelectionStart == 0)

{

return false;

}

return true;

}

public decimal Fx(decimal x)

{

decimal result;

result = aziretParser.Computer.Compute(inputFuncFX, x);

return result;

}

private void button4\_Click(object sender, EventArgs e)

{

OpenExcel();

}

private void button3\_Click(object sender, EventArgs e)

{

Clean(this);

progressBar1.Visible = false;

}

private void InitialApproximation\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = !IsOKForDecimalTextBox(e.KeyChar, InitialApproximation, false);

if (e.KeyChar == '.')

{

e.KeyChar = ',';

}

}

private void Tolerance\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = !IsOKForDecimalTextBox(e.KeyChar, Tolerance, true);

if (e.KeyChar == '.')

{

e.KeyChar = ',';

}

}

private void SearchStep\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = !IsOKForDecimalTextBox(e.KeyChar, SearchStep, true);

if (e.KeyChar == '.')

{

e.KeyChar = ',';

}

}

private void ParametrR\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((int)e.KeyChar == (int)48 && ParametrR.Text == "")

{

e.Handled = true;

return;

}

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar);

}

private String checkParse()

{

String errorMessage = "";

if (!parseTry(InitialApproximation, "Decimal"))

{

errorMessage += "Invalid value of the field x0 (the starting point of the approximation)! Change the input and perform the calculation!\n\n";

}

else

{

x0 = Decimal.Parse(InitialApproximation.Text, System.Globalization.NumberStyles.Float);

}

if (!parseTry(SearchStep, "Decimal"))

{

errorMessage += "Invalid value of the field search step! Change the input and perform the calculation!\n\n";

}

else

{

delta = Decimal.Parse(SearchStep.Text, System.Globalization.NumberStyles.Float);

}

if (parseTry(Tolerance, "Decimal"))

{

e\_tol = Decimal.Parse(Tolerance.Text, System.Globalization.NumberStyles.Float);

}

else

{

errorMessage += "Invalid value of the Tolerance(e) field (entered tolerance)! Change the input and perform the calculation!\n\n";

}

if (!parseTry(LimitOfIterations, "Integer"))

{

errorMessage += "Invalid value of the field limit of iterations! Change the input and perform the calculation!\n\n";

}

else

{

k\_max = Int32.Parse(LimitOfIterations.Text);

}

if (!parseTry(ParametrR, "Integer"))

{

errorMessage += "Invalid value of the field parameter R! Change the input and perform the calculation!\n\n";

}

else

{

parameterR = Decimal.Parse(ParametrR.Text);

}

if (!parseTry(LimitOfTime, "Decimal"))

{

errorMessage += "Invalid value of the field limit of time! Change the input and perform the calculation!\n\n";

}

else

{

t\_max = Decimal.Parse(LimitOfTime.Text, System.Globalization.NumberStyles.Float);

}

return errorMessage;

}

public bool fullCheck()

{

bool check = false;

if (Function.Text == "" || InitialApproximation.Text == "" ||

Tolerance.Text == "" || LimitOfIterations.Text == "" ||

LimitOfTime.Text == "" || SearchStep.Text == "" || ParametrR.Text == "")

{

MessageBox.Show("All fields must be filled in! Enter the missing information and make the calculation!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

if (checkParse() != "")

{

MessageBox.Show(checkParse(), "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

if (isRigth() && checkFunction(x0))

{

check = true;

}

}

}

return check;

}

public string getComparisonSign(decimal a, decimal b)

{

if (a > b)

{

return ">";

}

else if (a < b)

{

return "<";

}

else

{

return "=";

}

}

private bool isRigth()

{

bool b = true;

if (e\_tol <= 0)

{

MessageBox.Show("The value of the tolerance field must be greater than 0! Change the input and perform the calculation!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

b = false;

}

if (delta <= 0)

{

MessageBox.Show("The value of the search step field must be greater than 0! Change the input and perform the calculation!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

b = false;

}

if (k\_max <= 0)

{

MessageBox.Show("The value of the limit of iterations field must be greater than 0! Change the input and perform the calculation!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

b = false;

}

if (t\_max <= 0)

{

MessageBox.Show("The value of the limit of time field must be greater than 0! Change the input and perform the calculation!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

b = false;

}

if (parameterR <= 1)

{

MessageBox.Show("The value of the parameter R field must be greater than 1! Change the input and perform the calculation!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

b = false;

}

if (b)

{

return true;

}

return false;

}

private bool checkFunction(decimal x0)

{

inputFuncFX = Function.Text;

if (inputFuncFX == "" || inputFuncFX.IndexOf('x') == -1)

{

MessageBox.Show("The function is entered incorrectly! Change the input and perform the calculation!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

Clean(this);

return false;

}

try

{

if (inputFuncFX.Contains("log") && x0 <= 0 || inputFuncFX.Contains("ln") && x0 <= 0)

{

MessageBox.Show("If you entered function with 'log' or 'ln' value of X0 must greater than zero!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

else

{

decimal F1 = Fx(x0);

return true;

}

}

catch

{

MessageBox.Show("The function or initial approximation is entered incorrectly! Change the input and perform the calculation!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

Clean(this);

return false;

}

}

public void FillResult(string solution, string iterations, string resultTolerance, string fminustol, string fplustol, string fxvalue, string fminusplus, string fminusminus, string searchStep)

{

ResultX.Text = solution;

countofiterations.Text = iterations;

fxplustolerance.Text = fplustol;

fxminustolerance.Text = fminustol;

fxminusplustolerance.Text = fminusplus;

fxminusminustolerance.Text = fminusminus;

fx.Text = fxvalue;

}

public string getError(TextBox tol, decimal error)

{

Console.WriteLine(tol);

if (tol.Text.Contains("E"))

{

return error.ToString("0E0");

}

else if (tol.Text.Contains("e"))

{

return error.ToString("0e0");

}

else

{

return error.ToString();

}

}

private void LimitOfTime\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = !IsOKForDecimalTextBox(e.KeyChar, LimitOfTime, true);

if(e.KeyChar == '.')

{

e.KeyChar = ',';

}

}

private void LimitOfIterations\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((int)e.KeyChar == (int)48 && LimitOfIterations.Text == "")

{

e.Handled = true;

return;

}

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar);

}

private void button5\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show("1) Choose a function or write your's on field 'Function'\n" +

"2) Click on the button 'Show function graph'\n" +

"3) In the opened file select the values for a\n" +

"then save the document and return to the program\n" +

"4) If you need 'a' value to insert,\n" +

"click the button 'Set 'a' like 'X0'' or write your's\n" +

"5) Enter tolerance\n" +

"6) Enter delta\n" +

"7) Enter parameter R for constriction area size\n" +

"8) Enter limit of time in sec\n" +

"9) Enter limit of iterations \n" +

"10) Select search parameter\n" +

"Then click the button 'Run Method'.", "Information",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void button2\_Click(object sender, EventArgs e)

{

try

{

if (book == null)

{

book = xls.Workbooks.Open(System.IO.Directory.GetCurrentDirectory() + nameOfExcel);

}

if (sheet == null)

{

sheet = book.Sheets["Russian"];

sheet.Activate();

}

book.Save();

InitialApproximation.Text = sheet.Cells[4, 9].Value.ToString();

}

catch

{

book = xls.Workbooks.Open(System.IO.Directory.GetCurrentDirectory() + nameOfExcel);

sheet = book.Sheets["Russian"];

sheet.Activate();

book.Save();

InitialApproximation.Text = sheet.Cells[4, 9].Value.ToString();

}

xls.Visible = false;

book = null;

sheet = null;

}

private void button1\_Click(object sender, EventArgs e)

{

delta = 0;

x0 = 0;

x1 = 0;

f0 = 0;

f1 = 0;

e\_tol = 0;

k\_max = 0;

t\_max = 0;

parameterR = 0;

inputFuncFX = "";

fminusTol = 0;

fplusTol = 0;

string extremium;

try

{

if (fullCheck())

{

xls.Visible = false;

book = null;

sheet = null;

progressBar1.Value = 0;

Clean(groupBox2);

validation.Text = String.Empty;

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

//

cond = 0;

f0 = Fx(x0);

string firstDerivative = Derivative.ReturnDerivative(inputFuncFX);

string secondDerivative = Derivative.ReturnDerivative(firstDerivative);

df0 = Computer.Compute(firstDerivative, x0);

ddf0 = Computer.Compute(secondDerivative, x0);

dp = df0 / ddf0;

int k = 0;

do

{

k = k + 1;

progressBar1.Visible = true;

progressBar1.Maximum = (int)(k + 0.00000001);

progressBar1.Value = k;

if (k > k\_max)

{

stopwatch.Stop();

f1 = Fx(x1);

fminusTol = Fx(x1 - e\_tol);

fplusTol = Fx(x1 + e\_tol);

DialogResult result = MessageBox.Show("Iteration limit reached. Do you want to add iterations?",

"Information", MessageBoxButtons.YesNo, MessageBoxIcon.Information);

if (result == DialogResult.Yes)

{

k\_max += k\_max;

LimitOfIterations.Text = k\_max.ToString();

}

else

{

k--;

validation.Text += "Result X\* not found because of limit of iterations = " + k\_max + "." +

"\nSince the following condition is false, namely:" +

"\nSign(f(X\*)-f(X\*+Tolerance)) = " + getSign(f1 - fplusTol) + " and Sign(f(X\*)-f(X\*-Tolerance)) = " + getSign(f1 - fminusTol) + "!" +

"\nResult X\* is not extremum of the function.";

validation.ForeColor = Color.Red;

FillResult(x1.ToString("F28"), k.ToString(), getError(Tolerance, Math.Abs(x1 - x0)), fminusTol.ToString("F28"), fplusTol.ToString("F28"), f1.ToString("F28"), (f1 - fplusTol).ToString("F28"), (f1 - fminusTol).ToString("F28"), getError(absError, Math.Abs(delta)));

absError.Text = getError(Tolerance, Math.Abs(x1 - x0));

DialogResult answer = MessageBox.Show("Result X\* not found because of maximum limit of iterations = " + k\_max + "." +

"\nSince the following condition is false, namely:" +

"\nSign(f(X\*)-f(X\*+Tolerance)) = " + getSign(f1 - fplusTol) + " and Sign(f(X\*)-f(X\*-Tolerance)) = " + getSign(f1 - fminusTol) + "!" +

"\nResult X\* is not extremum of the function." +

"\n\nYou probably entered the value of 'a' incorrectly on Ecxel!" +

"\nYou need to open the graph and select the correct points [a;b]!" +

"\n\nDo you want to open file?", "Error", MessageBoxButtons.YesNo, MessageBoxIcon.Error);

if (answer == DialogResult.Yes)

{

OpenExcel();

}

break;

}

stopwatch.Start();

}

if (stopwatch.ElapsedMilliseconds >= t\_max \* 1000)

{

stopwatch.Stop();

f1 = Fx(x1);

fminusTol = Fx(x1 - e\_tol);

fplusTol = Fx(x1 + e\_tol);

DialogResult result = MessageBox.Show("Time limit reached. Do you want to add time?",

"Information", MessageBoxButtons.YesNo, MessageBoxIcon.Information);

if (result == DialogResult.Yes)

{

t\_max += t\_max;

LimitOfTime.Text = t\_max.ToString();

}

else

{

validation.Text += "Result X\* not found because of limit of time = " + t\_max + " sec." +

"\nSince the following condition is false, namely:" +

"\nSign(f(X\*)-f(X\*+Tolerance)) = " + getSign(f1 - fplusTol) + " and Sign(f(X\*)-f(X\*-Tolerance)) = " + getSign(f1 - fminusTol) + "!" +

"\nResult X\* is not extremum of the function.";

validation.ForeColor = Color.Red;

FillResult(x1.ToString("F28"), k.ToString(), getError(Tolerance, Math.Abs(x1 - x0)), fminusTol.ToString("F28"), fplusTol.ToString("F28"), f1.ToString("F28"), (f1 - fplusTol).ToString("F28"), (f1 - fminusTol).ToString("F28"), getError(absError, Math.Abs(delta)));

absError.Text = getError(Tolerance, Math.Abs(x1 - x0));

DialogResult answer = MessageBox.Show("Result X\* not found because of maximum time limit = " + t\_max + " sec." +

"\nSince the following condition is false, namely:" +

"\nSign(f(X\*)-f(X\*+Tolerance)) = " + getSign(f1 - fplusTol) + " and Sign(f(X\*)-f(X\*-Tolerance)) = " + getSign(f1 - fminusTol) + "!" +

"\nResult X\* is not extremum of the function." +

"\n\nYou probably entered the value of 'a' incorrectly on Ecxel!" +

"\nYou need to open the graph and select the correct points [a;b]!" +

"\n\nDo you want to open file?", "Error", MessageBoxButtons.YesNo, MessageBoxIcon.Error);

if (answer == DialogResult.Yes)

{

OpenExcel();

}

break;

}

stopwatch.Start();

}

if (Math.Abs(ddf0) <= e\_tol)

{

cond = 1;

QND = 0;

}

else

{

dp = df0 / ddf0;

}

if (k == 1)

{

dp0 = dp;

}

if (Math.Sign(dp0) == Math.Sign(dp))

{

x1 = x0 - dp;

}

else

{

x1 = x0 - dp / parameterR;

}

f1 = Fx(x1);

firstDerivative = Derivative.ReturnDerivative(inputFuncFX);

secondDerivative = Derivative.ReturnDerivative(firstDerivative);

df1 = Computer.Compute(firstDerivative, x1);

ddf1 = Computer.Compute(secondDerivative, x1);

dp0 = dp = df1 / ddf1;

relerror = Convert.ToDecimal(2) \* Math.Abs(dp) / (Math.Abs(x1) + e\_tol);

if (relerror < delta)

{

if (cond != 1)

{

cond = 2;

}

}

x0 = x1;

df0 = df1;

ddf0 = ddf1;

fminusTol = Fx(x1 - e\_tol);

fplusTol = Fx(x1 + e\_tol);

decimal controlFminusTol = Fx(x1 - Convert.ToDecimal(0.0001));

decimal controlFplusTol = Fx(x1 - Convert.ToDecimal(0.0001));

if (cond != 0)

{

if (f1 < fminusTol && f1 < fplusTol)

{

FillResult(x1.ToString("F28"), k.ToString(), getError(Tolerance, Math.Abs(x1 - x0)), fminusTol.ToString("F28"), fplusTol.ToString("F28"), f1.ToString("F28"), (f1 - fplusTol).ToString("F28"), (f1 - fminusTol).ToString("F28"), getError(absError, Math.Abs(relerror)));

validation.Text += "Since the following condition is true, namely:" +

"\nSign(f(X\*)-f(X\*+Tolerance)) = " + getSign(f1 - fplusTol) + " and Sign(f(X\*)-f(X\*-Tolerance)) = " + getSign(f1 - fminusTol) + "!" +

"\nResult X\* is minimizer of the function. It has been found with the error = " + relerror + ". This is less than or equal to given Tolerance!";

validation.ForeColor = Color.Green;

absError.Text = Convert.ToString(relerror);

break;

}

if ((f1 >= fminusTol && f1 >= fplusTol))

{

FillResult(x1.ToString("F28"), k.ToString(), getError(Tolerance, Math.Abs(x1 - x0)), fminusTol.ToString("F28"), fplusTol.ToString("F28"), f1.ToString("F28"), (f1 - fplusTol).ToString("F28"), (f1 - fminusTol).ToString("F28"), getError(absError, Math.Abs(relerror)));

if (f1 >= controlFminusTol && f1 >= controlFplusTol)

{

validation.Text += "Since the following condition is true, namely:" +

"\nSign(f(X\*)-f(X\*+Tolerance)) = " + getSign(f1 - fplusTol) + " and Sign(f(X\*)-f(X\*-Tolerance)) = " + getSign(f1 - fminusTol) + "!" +

"\nResult X\* is maximizer of the function. It has been found with the error = " + relerror + ". This is less than or equal to given Tolerance!";

validation.ForeColor = Color.Green;

}

else

{

validation.Text += "Since the following condition is false, namely: " + "\nSign(f(X\*)-f(X\*+Tolerance)) = " + getSign(f1 - fplusTol) + " and Sign(f(X\*)-f(X\*-Tolerance)) = " + getSign(f1 - fminusTol) + "!" + "Cannot be proven to be the minimum or maximum. Because " + "\nSign(f(X\*)-f(X\*+1e-10)) = " + getSign(f1 - controlFplusTol) + " and Sign(f(X\*)-f(X\*-1e-10)) = " + getSign(f1 - controlFminusTol) + "!";

validation.ForeColor = Color.Red;

}

absError.Text = Convert.ToString(relerror);

break;

}

FillResult(x1.ToString("F28"), k.ToString(), getError(Tolerance, Math.Abs(x1 - x0)), fminusTol.ToString("F28"), fplusTol.ToString("F28"), f1.ToString("F28"), (f1 - fplusTol).ToString("F28"), (f1 - fminusTol).ToString("F28"), getError(absError, Math.Abs(relerror)));

validation.Text += "Since the following condition is false, namely:" +

"\nSign(f(X\*)-f(X\*+Tolerance)) = " + getSign(f1 - fplusTol) + " and Sign(f(X\*)-f(X\*-Tolerance)) = " + getSign(f1 - fminusTol) + "!" +

"\nResult X\* is not maximizer or maximizer of the function. It's inflection point!";

validation.ForeColor = Color.Red;

absError.Text = Convert.ToString(relerror);

break;

}

} while (true);

stopwatch.Stop();

elapsedtime.Text = stopwatch.ElapsedMilliseconds / 1000.0 + " sec";

timer1.Enabled = true;

timer1.Start();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

Clean(this);

progressBar1.Value = 0;

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

progressBar1.Value = 0;

timer1.Enabled = false;

timer1.Stop();

}

private void Function\_TextChanged(object sender, EventArgs e)

{

Clean(groupBox2);

validation.Text = String.Empty;

}

private void InitialApproximation\_TextChanged(object sender, EventArgs e)

{

Clean(groupBox2);

validation.Text = String.Empty;

}

private void Tolerance\_TextChanged(object sender, EventArgs e)

{

Clean(groupBox2);

validation.Text = String.Empty;

}

private void SearchStep\_TextChanged(object sender, EventArgs e)

{

Clean(groupBox2);

validation.Text = String.Empty;

}

private void ParametrR\_TextChanged(object sender, EventArgs e)

{

Clean(groupBox2);

validation.Text = String.Empty;

}

private void LimitOfTime\_TextChanged(object sender, EventArgs e)

{

Clean(groupBox2);

validation.Text = String.Empty;

}

private void LimitOfIterations\_TextChanged(object sender, EventArgs e)

{

Clean(groupBox2);

validation.Text = String.Empty;

}

private void Maximum\_CheckedChanged(object sender, EventArgs e)

{

Clean(groupBox2);

validation.Text = String.Empty;

}

private void Minimum\_CheckedChanged(object sender, EventArgs e)

{

Clean(groupBox2);

validation.Text = String.Empty;

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

xls.Quit();

}

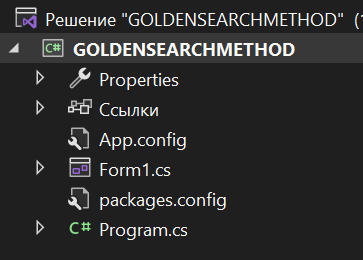
}

}

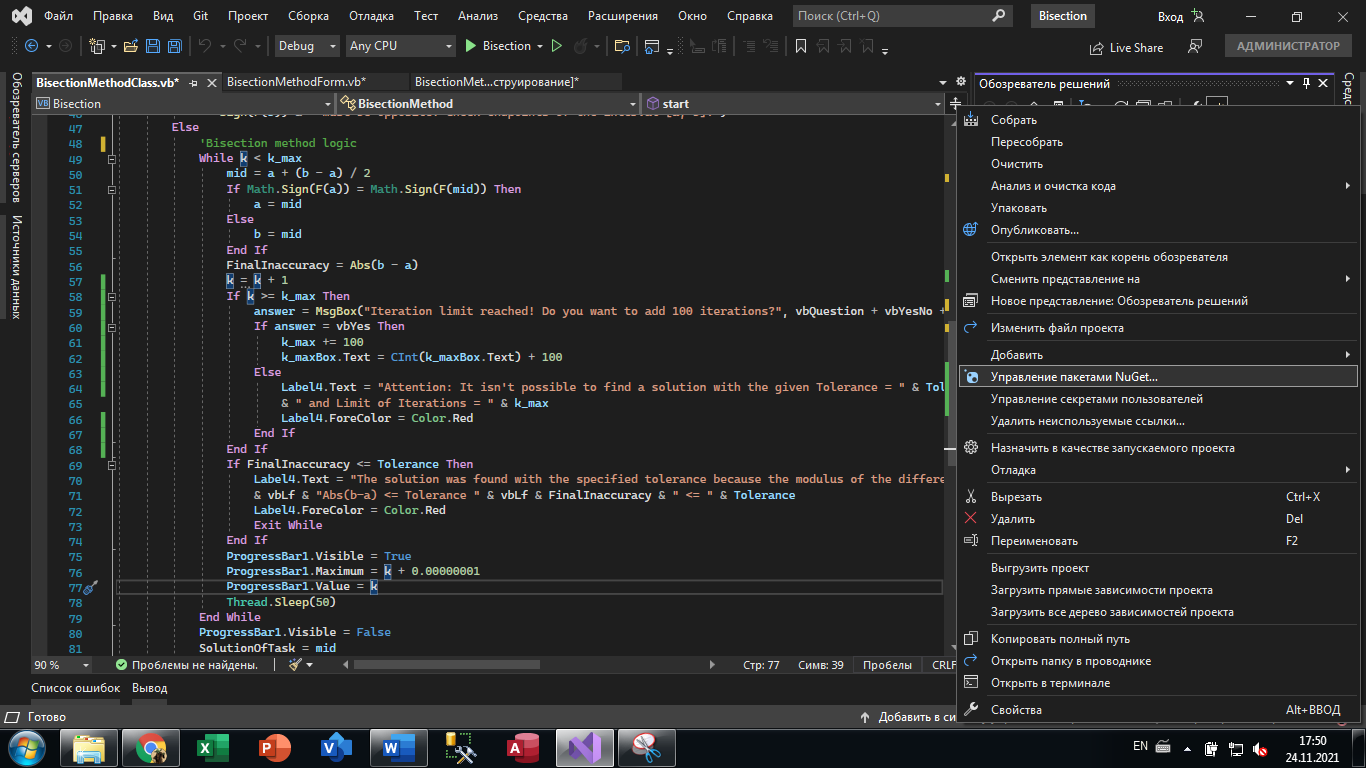
**6.3:** Подключение библиотечных программ “info.lundin.math.dll” и «Microsoft.Office.Interop.Excel» к программному проекту для выполнения функции парсинга и открытия excel файла

***Примечание***: *Если вставить приведенные в этом отчете коды программы, то Visual Studio выделит строчки кода листинга программы, в которых есть ссылки на библиотечные функции «info.lundin.math» и* «Microsoft.Office.Interop.Excel»*, как ошибочные. Это связано с тем, что в проект не включены ссылки на эти функции. Ниже приведена инструкция по включению в проект библиотек.*

Шаг №3.1: Правой кнопкой мыши открыть контекстное меню на выделенной синим цветом строке:

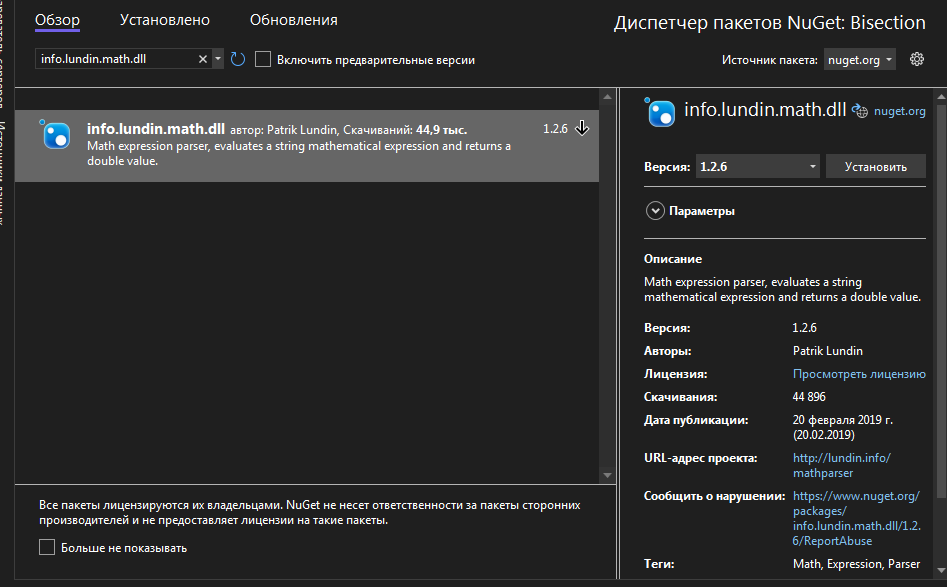


Шаг №3.2: Щелкнуть мышкой на строчке «Управление пакетами NuGet»:

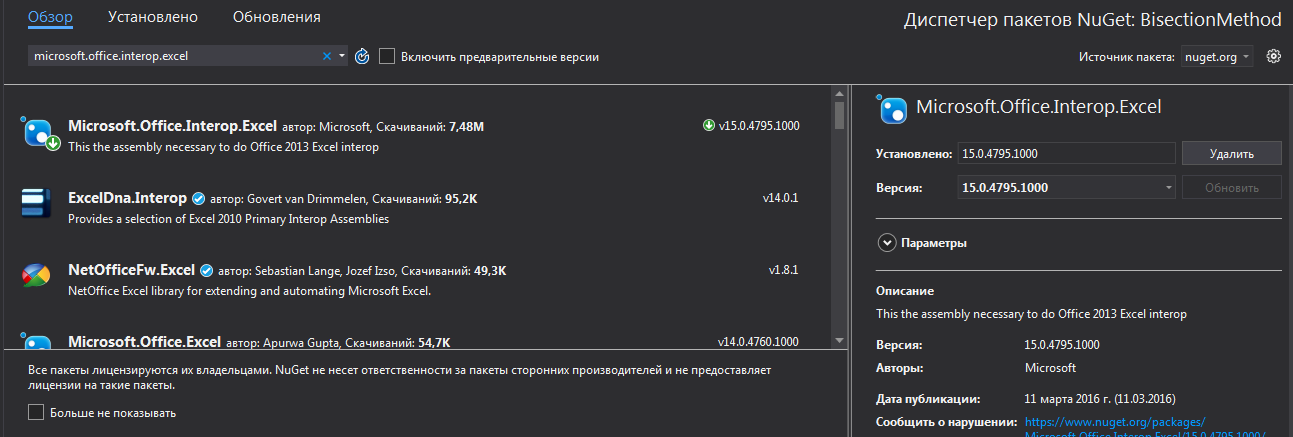


Шаг №3.3: В диалоговом окне в поиске ввести имя файла “info.lundin.math.dll”:

И установить пакет



Шаг №3.4: В диалоговом окне в поиске ввести имя файла “ microsoft.office.interop.excel”:

И установить пакет

Шаг №3.5: В листинге программы исчезнут все пометки об ошибках в коде программы, связанных с тем, что операторы «Imports info.lundin.math» и «Microsoft.Office.Interop.Excel» были неопределены, если нижеприведенные строки кода программы были уже введены до введения ссылки на эти библиотечные функции. Если же эти строки кода не были до сих пор введены, то теперь можно ввести эти коды, в которых используется функция парсинга, как это показано ниже:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*это примеры использования парсинга “info.lundin.math.dll”\*\*\*\*\*\*\*\*\*\*

ExpressionParser parser = new ExpressionParser();

parser.Values.Add("x", (double)x0);

double F1 = parser.Parse(inputFuncFX);

\*\*\*\*\*\*\*\*\*\*это примеры использования “Microsoft.Office.Interop.Excel”\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public void OpenExcel()

{

Application xls;

Workbook book;

Worksheet sheet;

string func;

double startPoint;

xls = new Application();

book = xls.Workbooks.Open(System.IO.Directory.GetCurrentDirectory() + nameOfExcel);

sheet = book.Sheets["Russian"];

xls.Visible = true;

sheet.Activate();

func = ComboBoxFunction.Text;

startPoint = Double.Parse(TextBoxStartPoint.Text);

sheet.Cells[4, 9] = startPoint;

sheet.Cells[4, 10] = startPoint + 1;

sheet.Cells[2, 1] = "f(x)=" + ComboBoxFunction.Text;

StringBuilder builder = new StringBuilder(func);

builder.Replace("exp", ":");

builder.Replace("x", "D4");

builder.Replace(":", "exp");

func = builder.ToString();

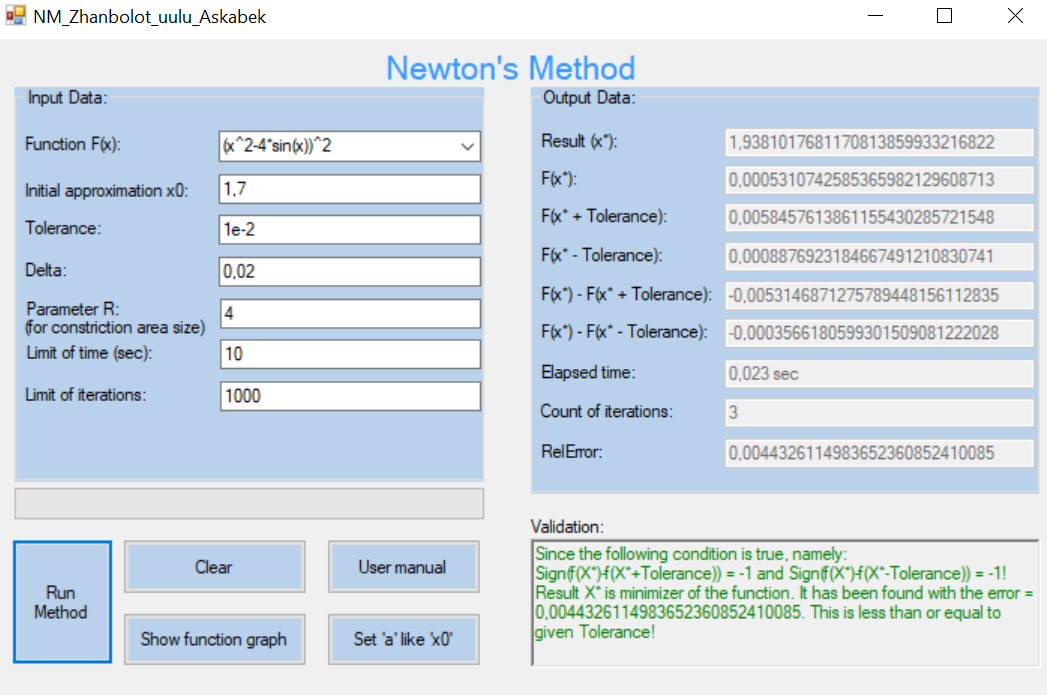
sheet.Range["E4:E10003"].Value = "=" + func;

}

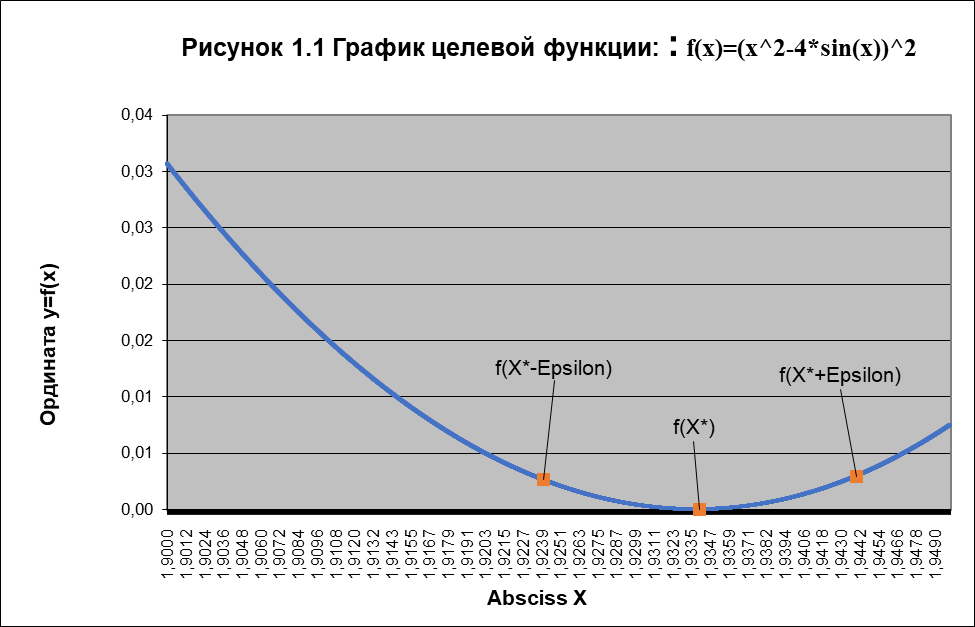
**Раздел 7**

**Тесты для проверки ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, РЕАЛИЗУЮЩЕГО NEWTON’S METHOD**

Тест №1: Цель теста проверка валидности решения задачи минимизации

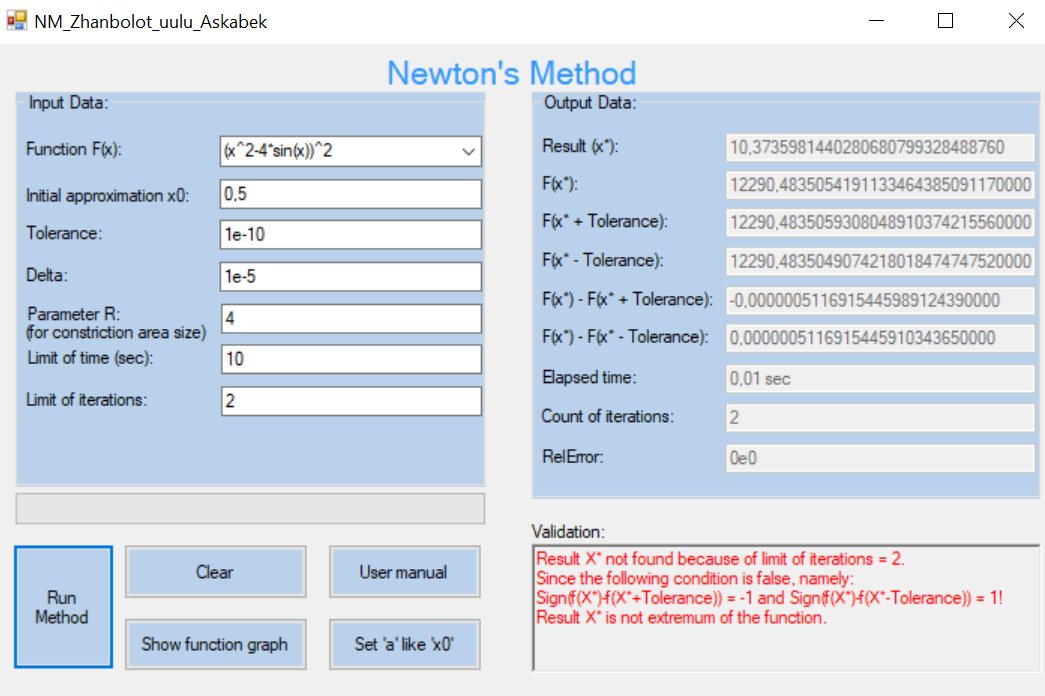


ЗАКЛЮЧЕНИЕ по результатам проведения Теста №1:



***Screen-shot 7.1*** The graph of the function 𝑓(𝑥) = (𝑥^2 − 4 ∗ sin(𝑥))^2 which visualizes the validation’s sentence such as: the value of the point x\* is a minimizer of the function 𝑓(𝑥 ∗ ) = 0,0005310742585365982129608713 𝑡ℎ𝑒 𝑜𝑛𝑒 𝑖𝑠 𝑐𝑜𝑚𝑝𝑢𝑡𝑒𝑑 𝑤𝑖𝑡ℎ 𝑡ℎ𝑒 𝒅𝒆𝒔𝒊𝒓𝒂𝒃𝒍𝒆 𝒂𝒄𝒄𝒖𝒓𝒂𝒄𝒚 since the following conditions are true, namely: 𝑠𝑖𝑔𝑛[𝑓(𝑥 ∗ + 𝑇𝑜𝑙𝑒𝑟𝑎𝑛𝑐𝑒)] = -1 𝒂𝒏𝒅 𝑠𝑖𝑔𝑛[𝑓(𝑥 ∗ − 𝑇𝑜𝑙𝑒𝑟𝑎𝑛𝑐𝑒)] = -1

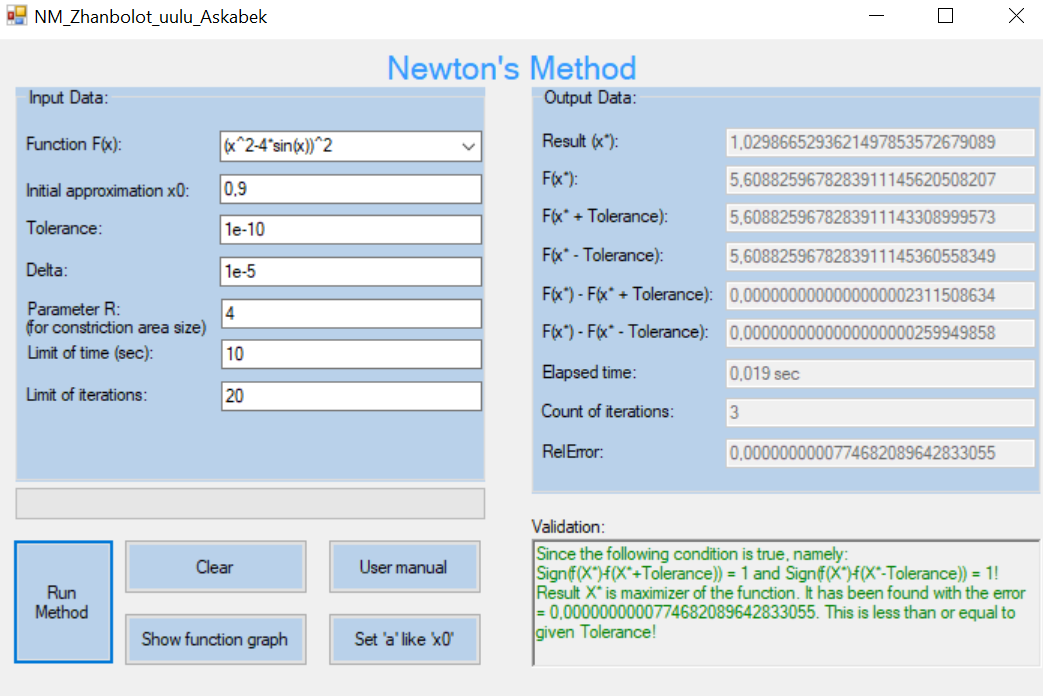
Тест №2: Цель теста проверка валидности на ограничение итераций



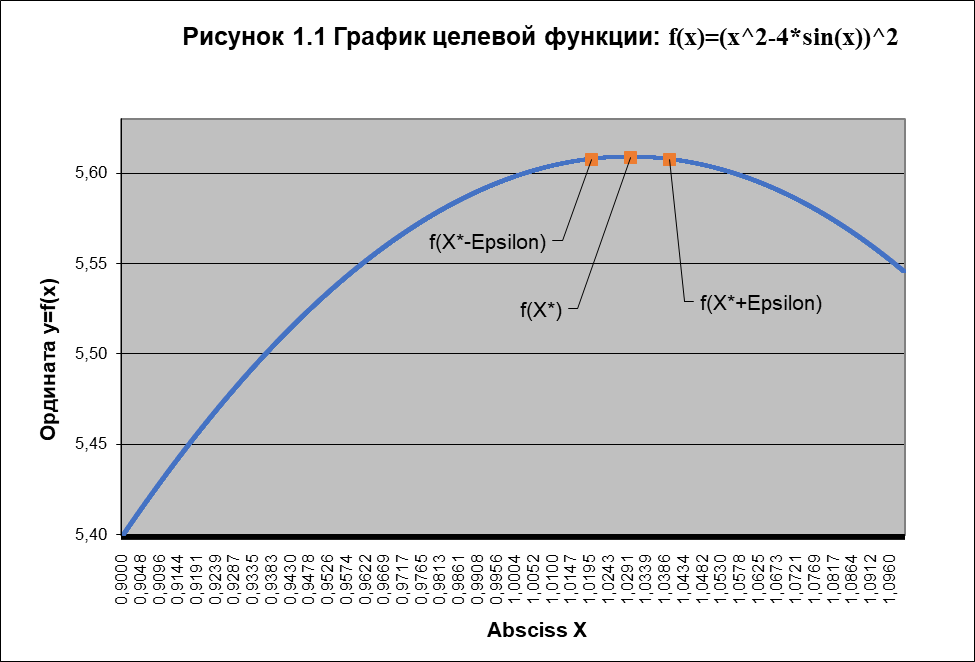
ЗАКЛЮЧЕНИЕ по результатам проведения Теста №2:

***Screen-shot 7.2*** The graph of the function 𝑓(𝑥) = (𝑥^2 − 4 ∗ sin(𝑥))^2 which visualizes the validation’s sentence such as: the value of the point x\* is not extremum of the function 𝑓(𝑥∗) = 12290,48054191402280680799328700000 since the following conditions are true, namely: 𝑠𝑖𝑔𝑛[𝑓(𝑥 ∗ + 𝑇𝑜𝑙𝑒𝑟𝑎𝑛𝑐𝑒)] = 1 𝒂𝒏𝒅 𝑠𝑖𝑔𝑛[𝑓(𝑥 ∗ − 𝑇𝑜𝑙𝑒𝑟𝑎𝑛𝑐𝑒)] = -1

Тест №3: Цель теста проверка валидности решения задачи максимизации

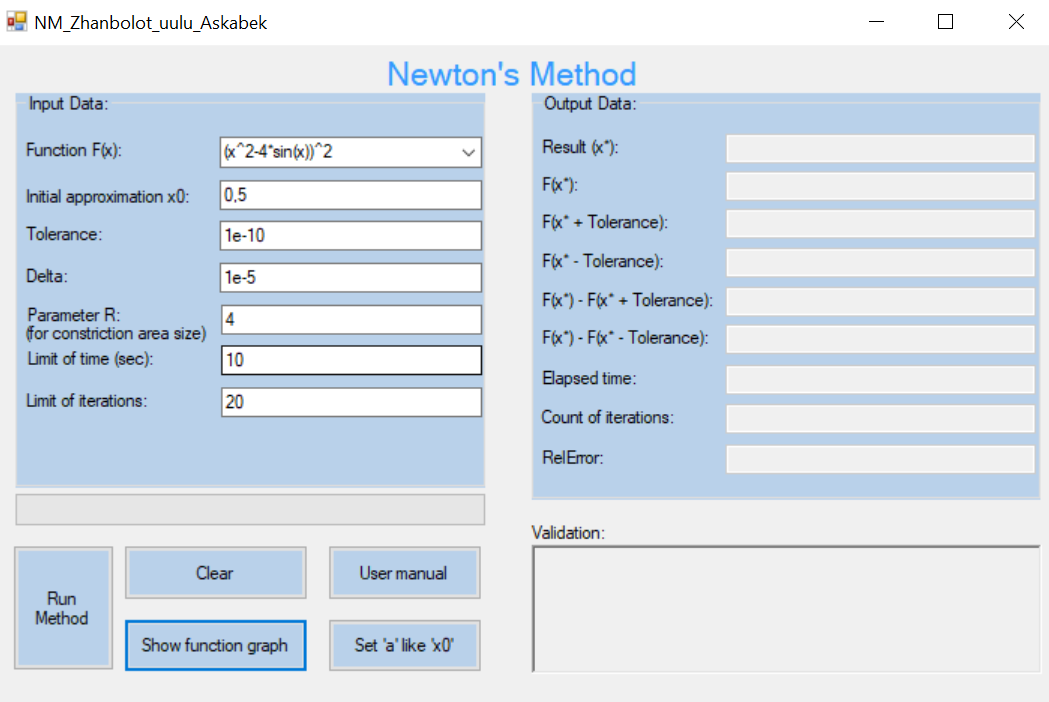


ЗАКЛЮЧЕНИЕ по результатам проведения Теста №3:

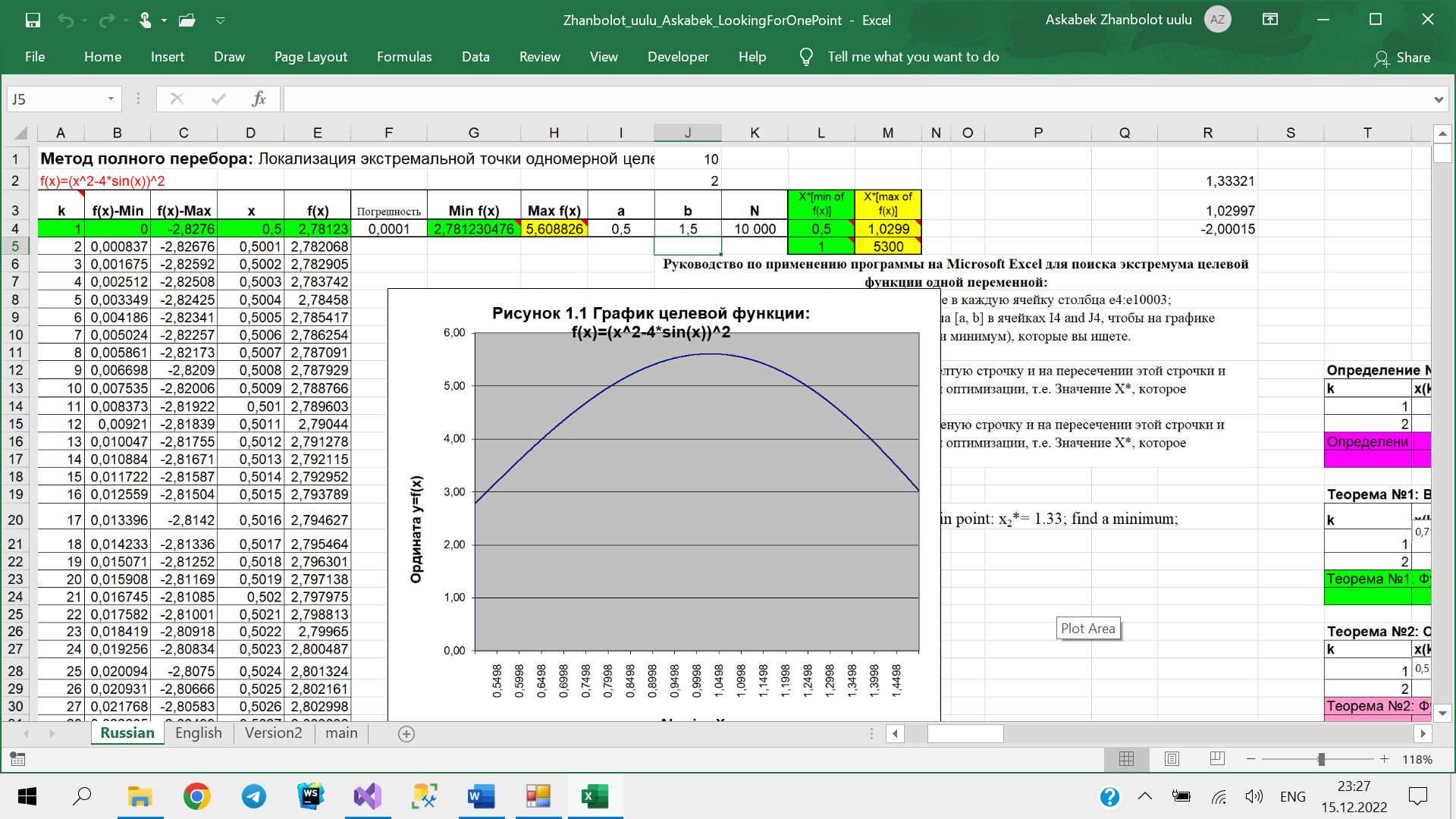


***Screen-shot 7.3*** The graph of the function 𝑓(𝑥) = (𝑥^2 − 4 ∗ sin(𝑥))^2 which visualizes the validation’s sentence such as: the value of the point x\* is a maximizer of the function 𝑓(𝑥 ∗ ) = 5,6088259678283911145820508207 𝑡ℎ𝑒 𝑜𝑛𝑒 𝑖𝑠 𝑐𝑜𝑚𝑝𝑢𝑡𝑒𝑑 𝑤𝑖𝑡ℎ 𝑡ℎ𝑒 𝒅𝒆𝒔𝒊𝒓𝒂𝒃𝒍𝒆 𝒂𝒄𝒄𝒖𝒓𝒂𝒄𝒚 since the following conditions are true, namely: 𝑠𝑖𝑔𝑛[𝑓(𝑥 ∗ + 𝑇𝑜𝑙𝑒𝑟𝑎𝑛𝑐𝑒)] = 1 𝒂𝒏𝒅 𝑠𝑖𝑔𝑛[𝑓(𝑥 ∗ − 𝑇𝑜𝑙𝑒𝑟𝑎𝑛𝑐𝑒)] = 1

Тест №4: Цель теста проверка открытия excel-file и вставки целевой функции с левой и правой границей в определенные ячейки листа “Russian”

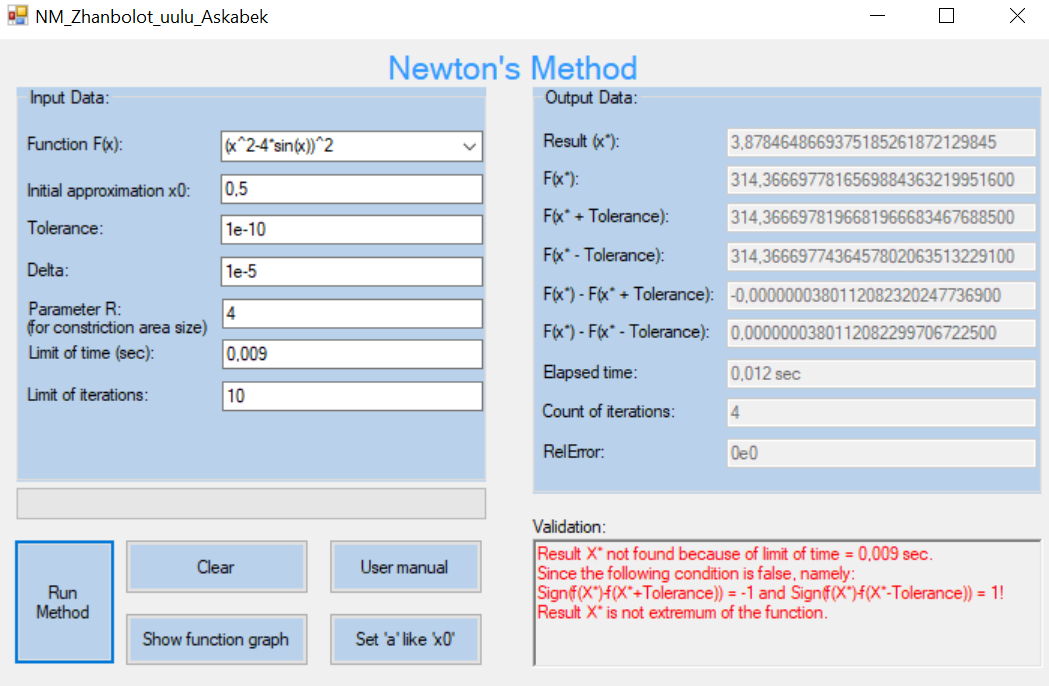


ЗАКЛЮЧЕНИЕ по результатам проведения Теста №4:



***Screen-shot 7.4*** Из excel-file видим, что в ячейке A2 установлена целевая функция, которую ввели в программе GoldenSectionSearchMethod; в ячейкe I4 установлена левая граница целевой функции, где левая граница имеет значение a= 0,5 из программы Newton’sMethod.

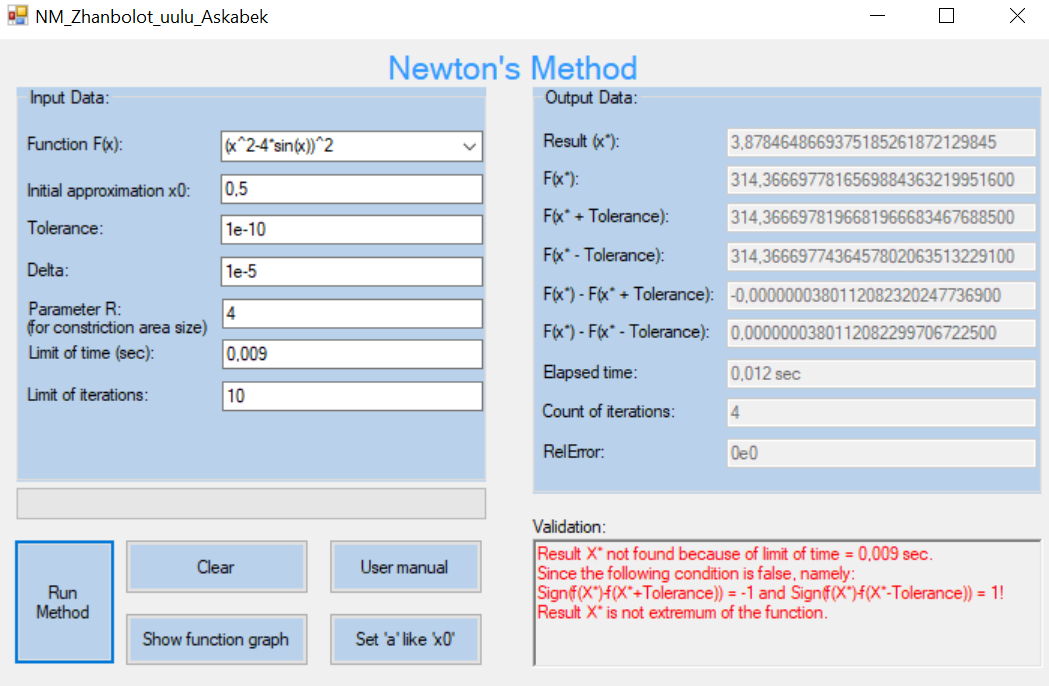
Тест №5: Цель теста проверка валидности на ограничение времени



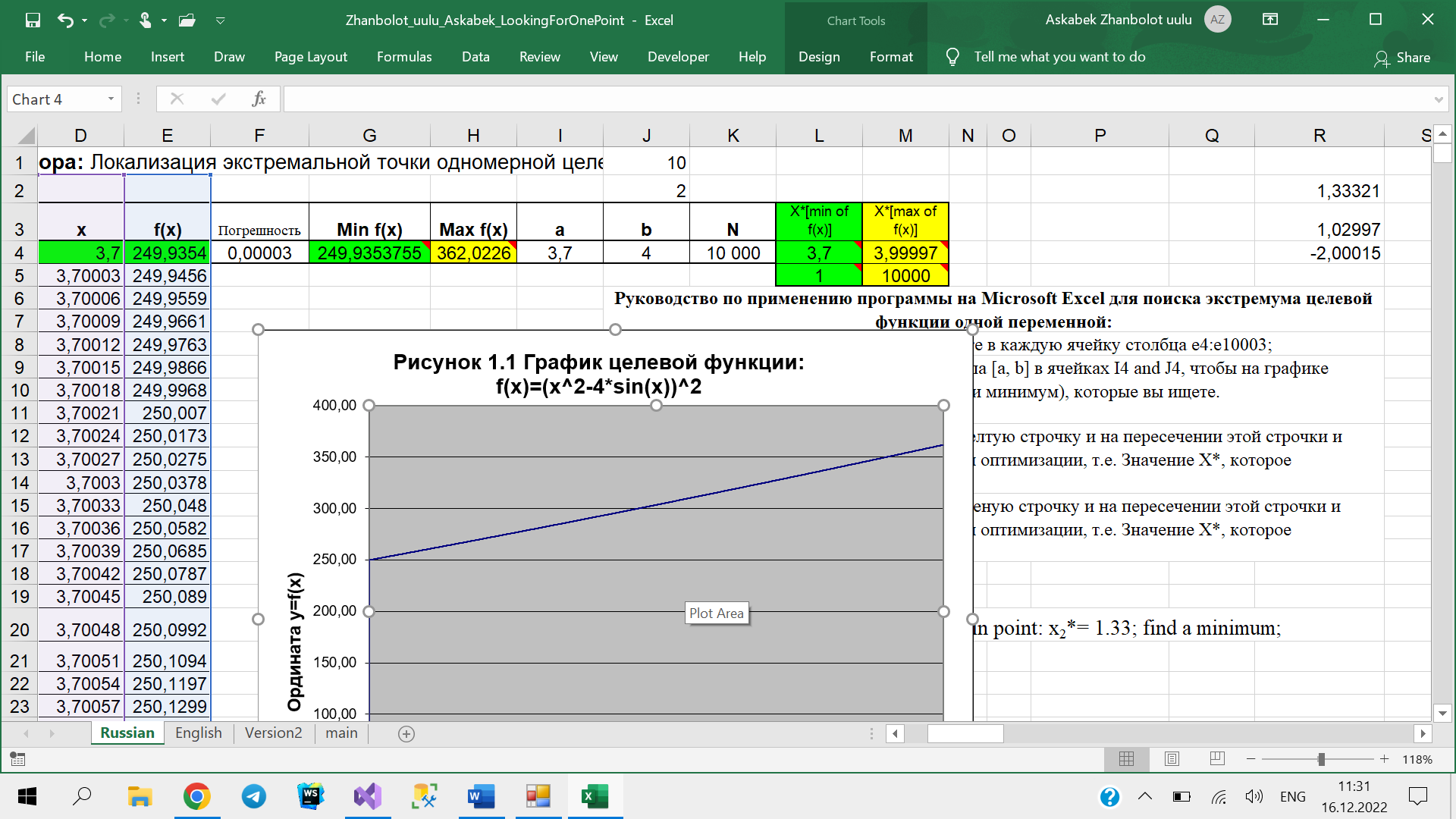
ЗАКЛЮЧЕНИЕ по результатам проведения Теста №5:

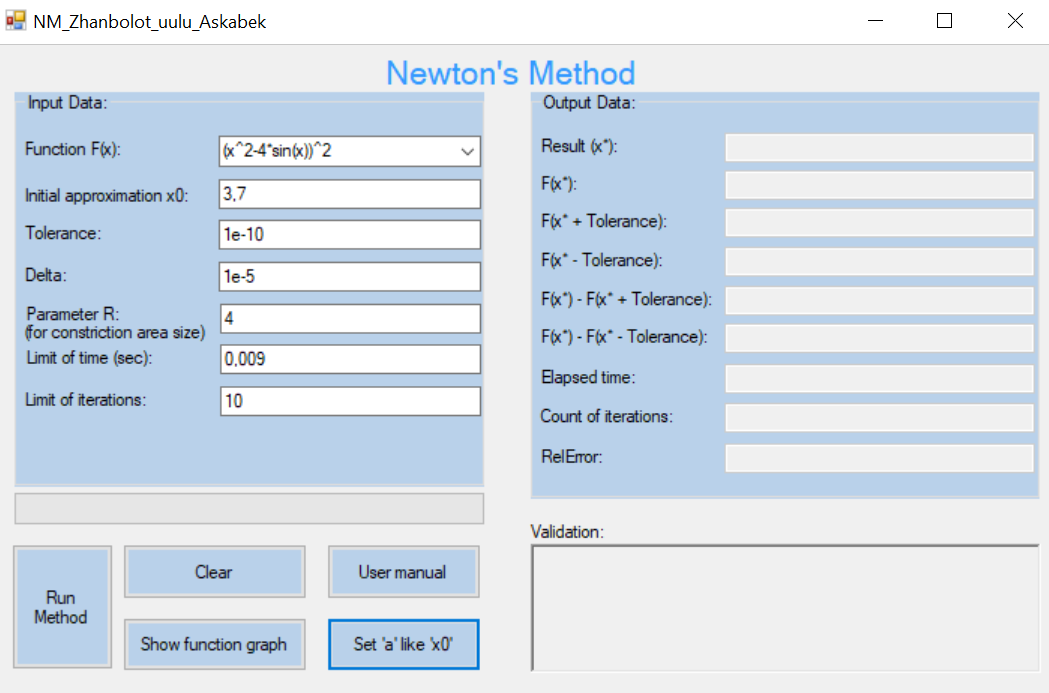
***Screen-shot 7.5*** The graph of the function 𝑓(𝑥) = (𝑥^2 − 4 ∗ sin(𝑥))^2 which visualizes the validation’s sentence such as: the value of the point x\* is not a extremum of the function 𝑓(𝑥∗) = 3,8781726394148963911123129845 since the following conditions are false, namely: 𝑠𝑖𝑔𝑛[𝑓(𝑥 ∗ + 𝑇𝑜𝑙𝑒𝑟𝑎𝑛𝑐𝑒)] = -1 𝒂𝒏𝒅 𝑠𝑖𝑔𝑛[𝑓(𝑥 ∗ − 𝑇𝑜𝑙𝑒𝑟𝑎𝑛𝑐𝑒)] = 1

Тест №6: Цель теста проверка импорта данных из Excel-файла



ЗАКЛЮЧЕНИЕ по результатам проведения Теста №6:





***Screen-shot 7.6*** После изменения значения в ячейке I4 , где хранится значение левой границы целевой функции, “Set ‘a’ like ‘x0’” в программе Newton’sMethod. Значение ячейки I4 равен 1,5.