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 ПРАКТИЧЕСКИМ РАБОТАМ**

по дисциплине

«МЕТОДЫ ОПТИМИЗАЦИИ»

Выполнил: студент группы ПИ-1-14

Аюпов А.

Проверил: кандидат технических наук, профессор

Тен Иосиф Григорьевич

Ver 2.2

Бишкек 2016

**Содержание**

[**Практическая работа №1** 3](#_Toc469472967)

[**Часть №1** 3](#_Toc469472968)

[**Часть №2** 3](#_Toc469472969)

[**Наименование работы** 3](#_Toc469472970)

[**Спецификация проблемы** 3](#_Toc469472971)

[**Спецификация метода** 3](#_Toc469472972)

[**Проектирование** 4](#_Toc469472973)

[**Листинг** 7](#_Toc469472974)

[**Тесты** 9](#_Toc469472975)

[**Практическая работа №2**](#_Toc469472976) 12

[**Часть №1**](#_Toc469472977) 12

[**Часть №2** 12](#_Toc469472978)

[**Наименование работы**](#_Toc469472979) 12

[**Спецификация проблемы**](#_Toc469472980) 12

[**Спецификация метода**](#_Toc469472981) 12

[**Проектирование**](#_Toc469472982) 13

[**Конструирование (листинг)**](#_Toc469472983) 16

[**Тесты**](#_Toc469472984) 18

[**Практическая работа №3**](#_Toc469472985) 21

[**Часть №1** 21](#_Toc469472986)

[**Часть №2** 21](#_Toc469472987)

[**Наименование работы** 21](#_Toc469472988)

[**Спецификация проблемы**](#_Toc469472989) 21

[**Спецификация метода**](#_Toc469472990) 21

[**Проектирование** 21](#_Toc469472991)

[**Конструирование (листинг)**](#_Toc469472992) 26

[**Тесты** 28](#_Toc469472993)

[**Практическая работа №4** 31](#_Toc469472994)

[**Часть №1**](#_Toc469472995) 31

[**Часть №2**](#_Toc469472996) 31

[**Наименование работы** 31](#_Toc469472997)

[**Спецификация проблемы** 31](#_Toc469472998)

[**Спецификация метода** 31](#_Toc469472999)

[**Проектирование** 33](#_Toc469473000)

[**Конструирование (листинг)** 37](#_Toc469473001)

[**Тесты** 39](#_Toc469473002)

Подключение библеотек……………………………………………………………………..**42**

# **Практическая работа №1**

# **Часть №1**

«Разработка ПО для поиска минимума ***нелинейной унимодальной функции***»

на основе итерационного метода: ***Even Search Method***

**Что дано:**

* ***Спецификация проблемы*** поиска минимума нелинейной функции f(x);
* ***Спецификация*** итерационного метода (***Even Search Method***) для нахождения минимума ***нелинейной унимодальной функции***
* ***Блок-схема*** Even Search method;
* ***Интерфейсная форма*** системы поиска минимума ***нелинейной унимодальной функции***, реализующей итерационный метод — ***Even Search Method***;
* ***Тесты*** для проверки ПО.

**Что требуется:**

* Разработать ***проект*** ПО для поиска минимума произвольной ***нелинейной унимодальной функции*** ***f(x)*** для произвольной заданной допустимой погрешности;
* Сконструировать систему, реализующую итерационный метод Even Search Method на основе использования парсера;
* Разработать ***код*** ПО для поиска минимума произвольной нелинейной функции f(x) для произвольной заданной допустимой погрешности;
* Провести валидацию системы – доказать идентичность результатов решения задач с помощью разработанного ПО заданным тестам.

# **Часть №2**

**Наименование работы –** Нахождение минимума нелинейной функции *методом равномерного поиска*

**СПЕЦИФИКАЦИЯ ПРОБЛЕМЫ №2: Нахождение МИНИМУМА нелинейной функции методом равномерного поиска**

* Найти минимум произвольной нелинейной функции

с заданной допустимой погрешностью ***Tolerance*** методом равномерного поиска. Нелинейная функция имеет ***произвольный*** аналитический вид, составленный из математических функций (полиномов различных степеней, тригонометрических – sin(x), cos(x), exp(x), ln(x), log(x) и. т. д.), который имеет математический смысл, и для которой существует хотя бы одно решение задачи.

**СПЕЦИФИКАЦИЯ (Описание) метода равномерного поиска**

Метод относится к пассивным стратегиям. Задается количество интервалов N, на которое разбивается исходный интервал L0 = [a0, b0]. Вычисления производятся в N +1 равноотстоящих друг от друга точках. Путем сравнения величин f(xi), i = 0,1,…,N находится точка xk, в которой значение функции наименьшее. Искомая точка минимума считается заключенной в интервале [xk-1, xk+1].

**Описание алгоритма решения проблемы в виде пошаговой итерационной процедуры**

1. Задать нелинейную функцию.
2. Задать начальную точку поиска x0 и точность tol.
3. Вычислить значение функции  f(x0).
4. Определить точку xi = xi-1+tol и значение функции  f(xi).
5. Если fx0 > fxi, то переходим на след шаг; иначе перейти на шаг 11
6. Если fx0 > fxi, то присвоить x0 значение xi а fx0=fxi
7. Присвоить xi+1=xi+tol затем вычислить функцию f(xi).
8. Вывести x0 и fx0

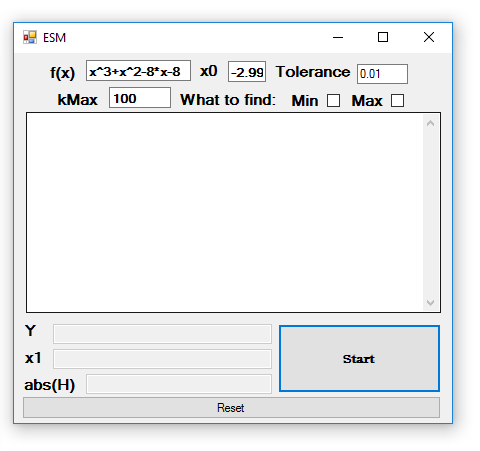
Поиск завершен

**Стадии *проектирования* системы для поиска минимума нелинейной функции , реализующей метод равномерного поиска:**

Стадия №.1: Разработка блок-схемы метода равномерного поиска



Стадия №.2: Проектирование интерфейса системы, реализующей метод равномерного поиска



Документирование процесса задания свойств элементов интерфейсной формы системы.

|  |  |  |
| --- | --- | --- |
| **Control** | **Property** | **Setting** |
| TextBox1 | Desing (Name) | FuncBox |
| Textbox1 | Appearance (Text) | x^3+x^2-8\*x-8 |
| TextBox2 | Desing (Name) | X0Box |
| TextBox2 | Appearance (Text) | -2.99 |
| TextBox3 | Design (Name) | ToleranceBox |
| TextBox3 | Appearance (Text) | 0,001 |
| TextBox4 | Desing (Name) | kmaxBox |
| TextBox4 | Appearance (Text) | 200 |
| Label1 | Design (Name) | FuncLabel |
| Label1 | Appearance (Text) | f(x) |
| Label2 | Design (Name) | X0Label |
| Label2 | Appearance (Text) | x0 |
| Label3 | Design (Name) | ToleranceLabel |
| Label3 | Appearance (Text) | Tolerance |
| Label4 | Design (Name) | k\_maxLabel |
| Label4 | Appearance (Text) | kMax |
| Label5 | Design (Name) | MaxMinLabel |
| Label5 | Appearance (Text) | What to find: |
| CheckBox1 | Design (Name) | checkBoxMin |
| CheckBox1 | Appearance (Text) | Min |
| CheckBox2 | Design (Name) | checkBoxMax |
| CheckBox2 | Appearance (Text) | Max |
| Button1 | Design (Name) | Start\_Btn |
| Button1 | Appearance (Text) | Start |
| Button2 | Design (Name) | Reset\_Btn |
| Button2 | Appearance (Text) | Reset |
| TextBox5 | Design (Name) | Calcs |
| Label6 | Design (Name) | SolYLabel |
| Label6 | Appearance (Text) | Y |
| Label7 | Appearance (Text) | x1 |
| Label7 | Design (Name) | SolXLabel |
| Label8 | Appearance (Text) | Abs(H) |
| Label8 | Design (Name) | AbsHLabel |
| Textbox6 | Design (Name) | SolY |
| Textbox6 | **Behavior (Enabled)** | False |
| Textbox7 | Design (Name) | SolX |
| Textbox7 | **Behavior (Enabled)** | False |
| Textbox8 | Design (Name) | AbsH |
| Textbox8 | **Behavior (Enabled)** | False |

## 

**Листинг программы на C#, ассоцированный с интерфейсной формой “ESM.cs”**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using info.lundin.math;

namespace FinalSolution

{

public partial class ESM : Form

{

public ESM()

{

InitializeComponent();

}

public ExpressionParser parser = new ExpressionParser();

public void Form1\_Load(object sender, EventArgs e)

{

parser.Values.Add("x", 0);

}

public double func(double x)

{

parser.Values.Remove("x");

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

return parser.Parse(FuncBox.Text);

}

public bool getError()

{

bool err = false;

Calcs.Text = "";

if ((!checkBoxMax.Checked && !checkBoxMin.Checked) || (checkBoxMax.Checked && checkBoxMin.Checked))

{

Calcs.AppendText("Choose what to find (min or max)" + Environment.NewLine);

err = true;

}

if (FuncBox.Text == "")

{

Calcs.AppendText("There are no function to solve (f(x))" +Environment.NewLine);

err = true;

}

if (X0Box.Text == "")

{

Calcs.AppendText("Enter first point (x0)" + Environment.NewLine);

err = true;

}

if (ToleranceBox.Text == "")

{

Calcs.AppendText("Enter tolerance value (tol)" + Environment.NewLine);

err = true;

}

if (kmaxBox.Text == "")

{

Calcs.AppendText("Enter maximum value of iterations (kMax)" + Environment.NewLine);

err = true;

}

return err;

}

private void Start\_Btn\_Click(object sender, EventArgs e)

{

if (!getError())

{

double x0 = double.Parse(X0Box.Text, System.Globalization.NumberStyles.Any), fx0 = func(x0), tol = double.Parse(ToleranceBox.Text, System.Globalization.NumberStyles.Any), H = tol, fx1, x1;

int k = 0; bool solved = false;

int kMax = int.Parse(kmaxBox.Text);

Calcs.AppendText("Step 1: H = tol => H = " + H);

Calcs.AppendText(Environment.NewLine);

fx0 = func(x0);

Calcs.AppendText("Step 2: f(x0) = " + fx0);

Calcs.AppendText(Environment.NewLine);

x1 = x0 + H;

Calcs.AppendText("Step 3: x1 = x0 + H => x1 = " + x0 + " + " + H + " = " + x1);

Calcs.AppendText(Environment.NewLine);

fx1 = func(x1);

Calcs.AppendText("Step 4: f(x1) = " + fx1);

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText("Step 5: k = 0");

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(Environment.NewLine);

if (checkBoxMin.Checked)

{

do

{

if (k > 0)

{

Calcs.AppendText("false");

Calcs.AppendText(Environment.NewLine);

}

//x^3+x^2-8\*x-8

Calcs.AppendText("Iteration " + ++k + ":");

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 1: k = k + 1 = " + k);

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 2: yf1 <= yf0 - ");

if (fx1 >= fx0)

{

Calcs.AppendText("true");

Calcs.AppendText(Environment.NewLine);

fx1 = fx0;

Calcs.AppendText(" Step 3: yf1 = yf0 => yf1 = " + fx1);

x1 = x0;

Calcs.AppendText("; x1 = x0 => x1 = " + x1);

solved = true;

}

else

{

Calcs.AppendText("false"); Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 3: x0 = x1 => x0 = " + x1 + "; yf0 = yf1 => yf0 = " + fx1);

x0 = x1; fx0 = fx1;

Calcs.AppendText(Environment.NewLine);

x1 = x1 + H;

Calcs.AppendText(" Step 4: x1 = x1 + H => x1 = " + x1);

Calcs.AppendText(Environment.NewLine);

fx1 = func(x1);

Calcs.AppendText(" Step 5: f(x1) = " + fx1);

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 6: yf1 <= yf0 - ");

}

} while ((!solved) && k < kMax);

}

else if (checkBoxMax.Checked)

{

do

{

if (k > 0)

{

Calcs.AppendText("false");

Calcs.AppendText(Environment.NewLine);

}

++k;

Calcs.AppendText("Iteration " + k + ":");

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 1: k = k + 1 = " + k);

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 2: yf1 <= yf0 - ");

if (fx1 <= fx0)

{

Calcs.AppendText("true");

Calcs.AppendText(Environment.NewLine);

fx1 = fx0;

Calcs.AppendText(" Step 3: yf1 = yf0 => yf1 = " + fx1);

x1 = x0;

Calcs.AppendText("; x1 = x0 => x1 = " + x1);

solved = true;

}

else

{

Calcs.AppendText("false");

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 3: x0 = x1 => x0 = " + x1 + "; yf0 = yf1 => yf0 = " + fx1);

x0 = x1; fx0 = fx1;

Calcs.AppendText(Environment.NewLine);

x1 = x1 + H;

Calcs.AppendText(" Step 4: x1 = x1 + H => x1 = " + x1); Calcs.AppendText(Environment.NewLine);

fx1 = func(x1);

Calcs.AppendText(" Step 5: f(x1) = " + fx1); Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 6: yf1 <= yf0 - ");

}

} while ( (!solved) && k < kMax);

}

SolY.Text = fx1.ToString();

SolX.Text = x1.ToString();

AbsH.Text = String.Format(Math.Abs(H).ToString(), "0e0");

}

}

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

{

SolY.Text = "";

SolX.Text = "";

AbsH.Text = "";

}

}

}

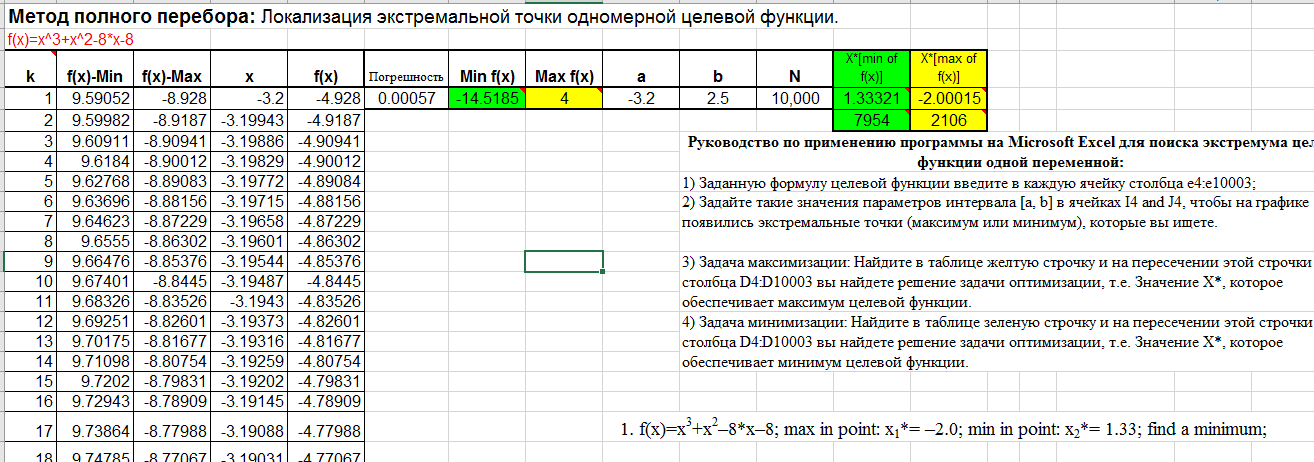
## 

**Тесты для проверки ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, РЕАЛИЗУЮЩЕГО МЕТОД РАВНОМЕРНОГО ПОИСКА**

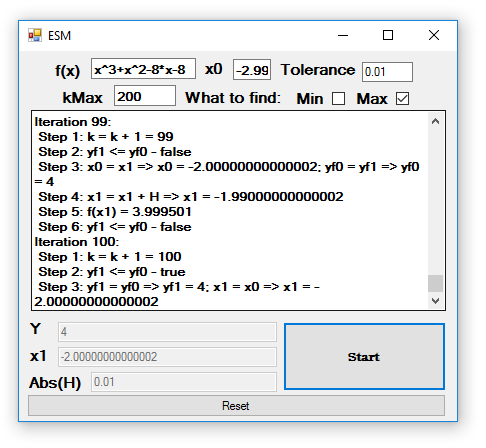
Тесты проводятся на основе программы, реализующей метод полного перебора в Excel MO\_LookingForOneOptPoint\_02.04.2012, разработанной преподавателем (приложено).

Тест №1

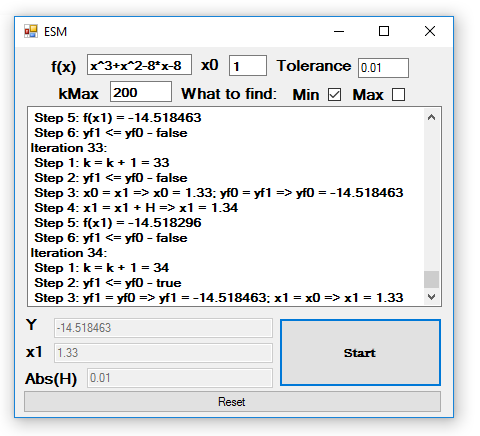
Эталон:



max:

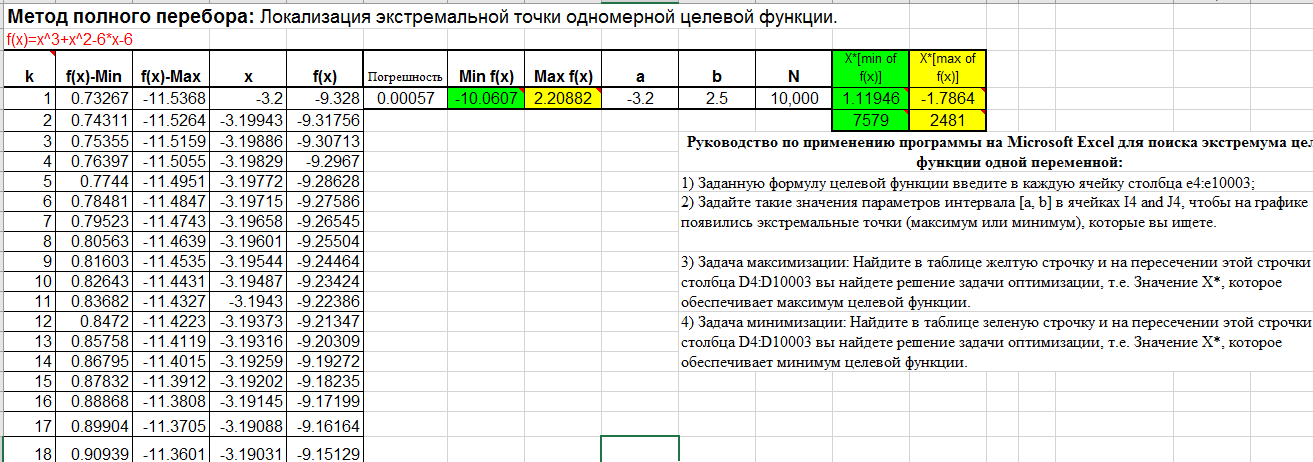


min:

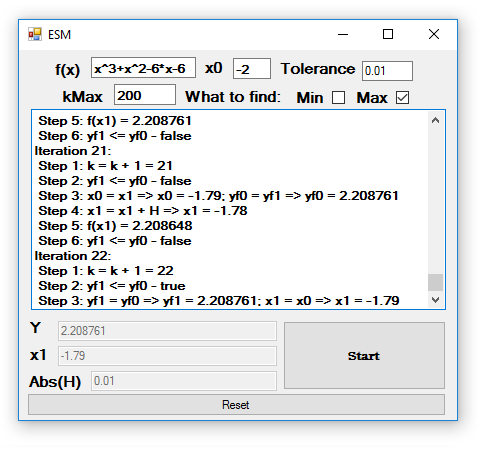


Тест №2

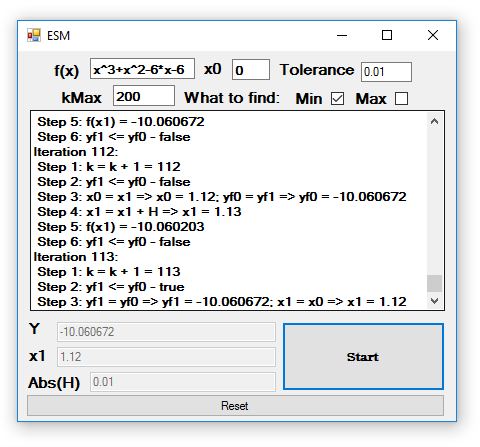
Эталон:



max:



min:



# **Практическая работа №2**

# **Часть №1**

«Разработка ПО для поиска ***минимума функций***

на основе *метода поразрядного приближения»*

**Что дано:**

* ***Спецификация проблемы*** поиска минимума произвольной нелинейной функции ;
* ***Спецификация*** метода *поразрядного приближения* для нахождения минимума нелинейной функции;
* ***Блок-схема*** метода *поразрядного приближения*;
* ***Интерфейсная форма*** системы поиска минимума нелинейной функции, реализующей метод *поразрядного приближения;*
* ***Тесты*** для проверки ПО.

**Что требуется:**

* Разработать ***проект*** ПО для решения произвольной оптимизационной задачи с произвольной погрешностью без ограничений методом поразрядного приближения
* Сконструировать систему, реализующую метод поразрядного приближения на основе использования парсера;
* Разработать ***код*** ПО для поиска минимума произвольной нелинейной функции для произвольно заданной допустимой погрешности.
* Провести валидацию системы – доказать идентичность результатов решения задач с помощью разработанного ПО заданным тестам.

# **Часть №2**

**Наименование работы –** Нахождение минимума нелинейной функции *методом поразрядного приближения*

**СПЕЦИФИКАЦИЯ ПРОБЛЕМЫ №2: Нахождение МИНИМУМА нелинейной функции методом поразрядного приближения**

* Найти минимум произвольной нелинейной функции

с заданной допустимой погрешностью ***Tolerance*** методом поразрядного приближения. Нелинейная функция имеет ***произвольный*** аналитический вид, составленный из математических функций (полиномов различных степеней, тригонометрических – sin(x), cos(x), exp(x), ln(x), log(x) и. т. д.), который имеет математический смысл, и для которой существует хотя бы одно решение задачи.

**СПЕЦИФИКАЦИЯ (Описание) метода поразрядного приближения**

Метод является усовершенствованным вариантом метода перебора. В этом методе перебор точек интервала неопределенности происходит с шагом h, i = 0, 1, … до тех пор, пока количество итераций не станет больше заданного или h будет меньше чем точность/параметр R.

После этого шаг уменьшается в несколько -R раз, и производится перебор точек в противоположном направлении (с новым шагом) до тех пор, пока значения f(x) не перестанут уменьшаться.

Процедура уменьшения шага и смены направления перебора на противоположное повторяется несколько раз. Поиск прекращается, если текущий шаг дискретизации при последнем проходе алгоритма не превосходит заданной точности.

**Описание алгоритма решения проблемы в виде пошаговой итерационной процедуры**

1. Задать нелинейную функцию.
2. Задать начальную точку поиска и точность ε.
3. Задать начальный шаг дискретизации h и параметр R
4. Положить x0.
5. Вычислить значение функции  f(x0).
6. Определить точку xi = xi-1+Δ и значение функции  f(xi).
7. Если (Abs(h1) >= Epsilon/R) то i=i+1 иначе перейти на шаг 10
8. Если fx0 > fxi, то присвоить x0 значение xi а fx0=fxi иначе присвоить x0 значение xi а fx0=fxi и поменять шаг поиска h=-(h/R)
9. Присвоить xi+1=xi+h затем вычислить функцию.
10. Вывести x0 и fx0

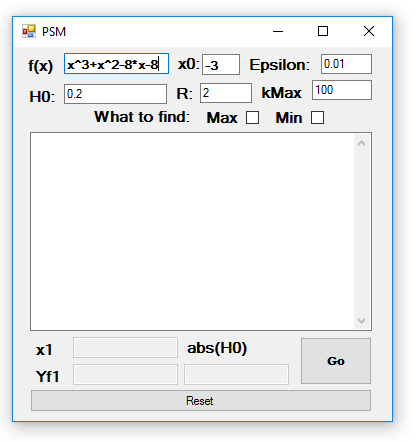
Поиск завершен

**Стадии *проектирования* системы для поиска минимума нелинейной функции , реализующей метод поразрядного приближения:**

Стадия №.1: Разработка блок-схемы метода поразрядного приближения



Стадия №.2: Проектирование интерфейса системы, реализующей метод поразрядного приближения.



Документирование процесса задания свойств элементов интерфейсной формы системы.

|  |  |  |
| --- | --- | --- |
| **Control** | **Property** | **Setting** |
| TextBox1 | Desing (Name) | inFx |
| TextBox1 | Appearance (Text) | x^3+x^2-8\*x-8 |
| TextBox2 | Desing (Name) | inX0 |
| TextBox2 | Appearance (Text) | -3 |
| TextBox3 | Design (Name) | inEpsilon |
| TextBox3 | Appearance (Text) | 0,01 |
| TextBox4 | Design (Name) | inH0 |
| TextBox4 | Appearance (Text) | 0,2 |
| TextBox5 | Design (Name) | inR |
| TextBox5 | Appearance (Text) | 2 |
| TextBox6 | Desing (Name) | inKMax |
| TextBox6 | Appearance (Text) | 100 |
| Label1 | Design (Name) | FuncLabel |
| Label1 | Appearance (Text) | f(x) |
| Label2 | Design (Name) | X0Label |
| Label2 | Appearance (Text) | x0: |
| Label3 | Design (Name) | EpsilonLabel |
| Label3 | Appearance (Text) | Epsilon: |
| Label4 | Design (Name) | H0Label |
| Label4 | Appearance (Text) | H0: |
| Label5 | Design (Name) | RLabel |
| Label5 | Appearance (Text) | R: |
| Label6 | Design (Name) | KmaxLabel |
| Label6 | Appearance (Text) | kMax |
| Label7 | Design (Name) | MaxMinLabel |
| Label7 | Appearance (Text) | What to find: |
| CheckBox1 | Design (Name) | maxCheckBox |
| CheckBox1 | Appearance (Text) | Max |
| CheckBox2 | Design (Name) | MinCheckBox |
| CheckBox2 | Appearance (Text) | Min |
| TextBox7 | Design (Name) | Calcs |
| TextBox7 | Behavior (Multiline) | True |
| Button1 | Design (Name) | Start\_Btn |
| Button1 | Appearance (Text) | Go |
| Button2 | Design (Name) | Reset\_Btn |
| Button2 | Appearance (Text) | Reset |
| Label8 | Appearance (Text) | x1 |
| Label8 | Design (Name) | OutX1Label |
| Label9 | Appearance (Text) | Yf1 |
| Label9 | Design (Name) | OutYf1Label |
| Label10 | Appearance (Text) | abs(H0) |
| Label10 | Design (Name) | OutAbsH0Label |
| Textbox8 | Design (Name) | OutX1 |
| Textbox8 | **Behavior (Enabled)** | False |
| Textbox9 | Design (Name) | OutYf1 |
| Textbox9 | **Behavior (Enabled)** | False |
| Textbox10 | Design (Name) | OutAbsH0 |
| Textbox10 | **Behavior (Enabled)** | False |

**Стадия *конструирования* системы для поиска минимума нелинейной функции , реализующей метод поразрядного приближения:**

Стадия №3: Код программы на C#. (PSM.cs)

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using info.lundin.math;

namespace FinalSolution

{

public partial class PSM : Form

{

public ExpressionParser parser = new ExpressionParser();

public void Form1\_Load(object sender, EventArgs e)

{

parser.Values.Add("x", 0);

}

public double func(double x)

{

parser.Values.Remove("x");

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

return parser.Parse(inFx.Text);

}

public PSM()

{

InitializeComponent();

}

public bool getError()

{

bool err = false;

Calcs.Text = "";

if ((!minCheckBox.Checked && !maxCheckBox.Checked) || (minCheckBox.Checked && maxCheckBox.Checked))

{

Calcs.AppendText("Choose what to find (min or max)" + Environment.NewLine);

err = true;

}

if (inFx.Text == "")

{

Calcs.AppendText("There are no function to solve (f(x))" + Environment.NewLine);

err = true; }

if (inX0.Text == "")

{

Calcs.AppendText("Enter first point (x0)" + Environment.NewLine);

err = true;

}

if (inH0.Text == "") {

Calcs.AppendText("Enter H0" + Environment.NewLine);

err = true; }

if (inR.Text == "") {

Calcs.AppendText("Enter R parameter" + Environment.NewLine);

err = true; }

if (inEpsilon.Text == "") {

Calcs.AppendText("Enter Epsilon" + Environment.NewLine); err = true; }

if (inKMax.Text == "") {

Calcs.AppendText("Enter maximum value of iterations (kMax)" + Environment.NewLine); err = true; }

return err; }

private void Start\_Btn\_Click(object sender, EventArgs e) {

if (!getError()) {

double x0 = double.Parse(inX0.Text, System.Globalization.NumberStyles.Any), fx0 = func(x0), fx1, x1, Epsilon = double.Parse(inEpsilon.Text, System.Globalization.NumberStyles.Any), H0 = double.Parse(inH0.Text, System.Globalization.NumberStyles.Any), H1 = H0, R = double.Parse(inR.Text, System.Globalization.NumberStyles.Any);

int k = 0, kMax = int.Parse(inKMax.Text);

Calcs.Text = "";

Calcs.AppendText("Step 1: f(x0) = " + fx0);

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText("Step 2 H1 = H0 => H1 = " + H1);

Calcs.AppendText(Environment.NewLine);

x1 = x0 + H1;

Calcs.AppendText("Step 3 x1 = x0 + H1 => x1 = " + x0 + " + " + H1 + " = " + x1);

Calcs.AppendText(Environment.NewLine);

fx1 = func(x1);

Calcs.AppendText("Step 4 Function(x1) = " + fx1);

Calcs.AppendText(Environment.NewLine);

k = 0;

Calcs.AppendText("Step 5 k = " + k);

Calcs.AppendText(Environment.NewLine);

if (!minCheckBox.Checked)

do

{

if (k > 0)

{

Calcs.AppendText("true"); Calcs.AppendText(Environment.NewLine);

}

// Step 1

Calcs.AppendText("Iteration " + ++k); Calcs.AppendText(Environment.NewLine);

Calcs.AppendText("Step 1 k = k + 1 = " + k); Calcs.AppendText(Environment.NewLine);

// Step 2

Calcs.AppendText("Step 2 yf1 <= yf0 => " + fx1 + " <= " + fx0 + " - ");

if (fx1 <= fx0)

{

Calcs.AppendText("true"); Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 3 ABS(H0) < Epsilon/R => " + Math.Abs(H0) + " < " + Epsilon + "/" + R + " - ");

if (Math.Abs(H0) < (Epsilon / R))

{

Calcs.AppendText("true"); Calcs.AppendText(Environment.NewLine);

H1 = H0;

Calcs.AppendText(" Step 4 H1 = H0 => H1 = " + H0); Calcs.AppendText(Environment.NewLine);

x1 = x0;

Calcs.AppendText(" Step 5 x1 = x0 => x1 = " + x1); Calcs.AppendText(Environment.NewLine);

fx1 = fx0;

Calcs.AppendText(" Step 6 yf1 = yf0 => yf1 = " + fx1); Calcs.AppendText(Environment.NewLine);

break;

}

else

{

Calcs.AppendText("false");

Calcs.AppendText(Environment.NewLine);

H1 = -H0 / R;

Calcs.AppendText(" Step 4 H1 = -H0 / R => H1 = " + -H0 + " / " + R + " = " + H1);

Calcs.AppendText(Environment.NewLine);

H0 = H1;

Calcs.AppendText(" Step 5 H0 = H1 => H0 = " + H0);

Calcs.AppendText(Environment.NewLine);

x0 = x1; fx0 = fx1;

Calcs.AppendText(" Step 6 x0 = x1 => x0 = " + x0 + ";\t yf0 = yf1 => yf0 = " + fx0);

Calcs.AppendText(Environment.NewLine);

x1 = x0 + H1;

Calcs.AppendText(" Step 7 x1 = x0 + H1 => x1 = " + x0 + " + " + H1 + " = " + x1);

Calcs.AppendText(Environment.NewLine);

fx1 = func(x1);

Calcs.AppendText(" Step 8 Function(x1) = " + fx1);

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 9 ");//if(k < k\_Max)\*/

}

}

else

{

Calcs.AppendText("false");

Calcs.AppendText(Environment.NewLine);

H1 = H0;

Calcs.AppendText(" Step 3 H1 = H0 => H1 = " + H1);

Calcs.AppendText(Environment.NewLine);

x0 = x1;

fx0 = fx1;

Calcs.AppendText(" Step 4 x0 = x1 => x0 = " + x0 + "; yf0 = yf1 => yf0 = " + fx0);

Calcs.AppendText(Environment.NewLine);

x1 = x0 + H1;

Calcs.AppendText(" Step 5 x1 = x0 + H1 => x1 = " + x0 + " + " + H1 + " = " + x1);

Calcs.AppendText(Environment.NewLine);

fx1 = func(x1);

Calcs.AppendText(" Step 6 Function(x1) = " + fx1);

Calcs.AppendText(" Step 7 "); }

} while (k < kMax);

else {

do {

if (k > 0) {

Calcs.AppendText("true");

Calcs.AppendText(Environment.NewLine); }

Calcs.AppendText("Iteration " + ++k);

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText("Step 1 k = k + 1 = " + k);

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText("Step 2 yf1 >= yf0 => " + fx1 + " <= " + fx0 + " - ");

if (fx1 >= fx0) {

Calcs.AppendText("true");

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 3 ABS(H0) < Epsilon/R => " + Math.Abs(H0) + " < " + Epsilon + "/" + R + " - ");

if (Math.Abs(H0) < (Epsilon / R))

{

Calcs.AppendText("true");

Calcs.AppendText(Environment.NewLine);

H1 = H0;

Calcs.AppendText(" Step 4 H1 = H0 => H1 = " + H0);

Calcs.AppendText(Environment.NewLine);

x1 = x0;

Calcs.AppendText(" Step 5 x1 = x0 => x1 = " + x1);

Calcs.AppendText(Environment.NewLine);

fx1 = fx0;

Calcs.AppendText(" Step 6 yf1 = yf0 => yf1 = " + fx1);

Calcs.AppendText(Environment.NewLine);

break;

}

else {

Calcs.AppendText("false");

Calcs.AppendText(Environment.NewLine);

H1 = -H0 / R;

Calcs.AppendText(" Step 4 H1 = -H0 / R => H1 = " + -H0 + " / " + R + " = " + H1);

Calcs.AppendText(Environment.NewLine);

H0 = H1;

Calcs.AppendText(" Step 5 H0 = H1 => H0 = " + H0);

Calcs.AppendText(Environment.NewLine);

x0 = x1; fx0 = fx1;

Calcs.AppendText(" Step 6 x0 = x1 => x0 = " + x0 + ";\t yf0 = yf1 => yf0 = " + fx0);

Calcs.AppendText(Environment.NewLine);

x1 = x0 + H1;

Calcs.AppendText(" Step 7 x1 = x0 + H1 => x1 = " + x0 + " + " + H1 + " = " + x1);

Calcs.AppendText(Environment.NewLine);

fx1 = func(x1);

Calcs.AppendText(" Step 8 Function(x1) = " + fx1);

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 9 ");//if(k < k\_Max)\*/ } }

else {

Calcs.AppendText("false");

Calcs.AppendText(Environment.NewLine);

H1 = H0;

Calcs.AppendText(" Step 3 H1 = H0 => H1 = " + H1);

Calcs.AppendText(Environment.NewLine);

x0 = x1;

fx0 = fx1;

Calcs.AppendText(" Step 4 x0 = x1 => x0 = " + x0 + "; yf0 = yf1 => yf0 = " + fx0);

Calcs.AppendText(Environment.NewLine);

x1 = x0 + H1;

Calcs.AppendText(" Step 5 x1 = x0 + H1 => x1 = " + x0 + " + " + H1 + " = " + x1);

Calcs.AppendText(Environment.NewLine);

fx1 = func(x1);

Calcs.AppendText(" Step 6 Function(x1) = " + fx1);

Calcs.AppendText(" Step 7 "); }

} while (k < kMax); }

outX1.Text = x1.ToString();

outAbsH0.Text = Math.Abs(H0).ToString();

outYf1.Text = fx1.ToString(); } }

private void Reset\_Btn\_Click(object sender, EventArgs e) {

outX1.Text = "";

outAbsH0.Text = "";

outYf1.Text = "";

Calcs.Text = "";

minCheckBox.Checked = false;

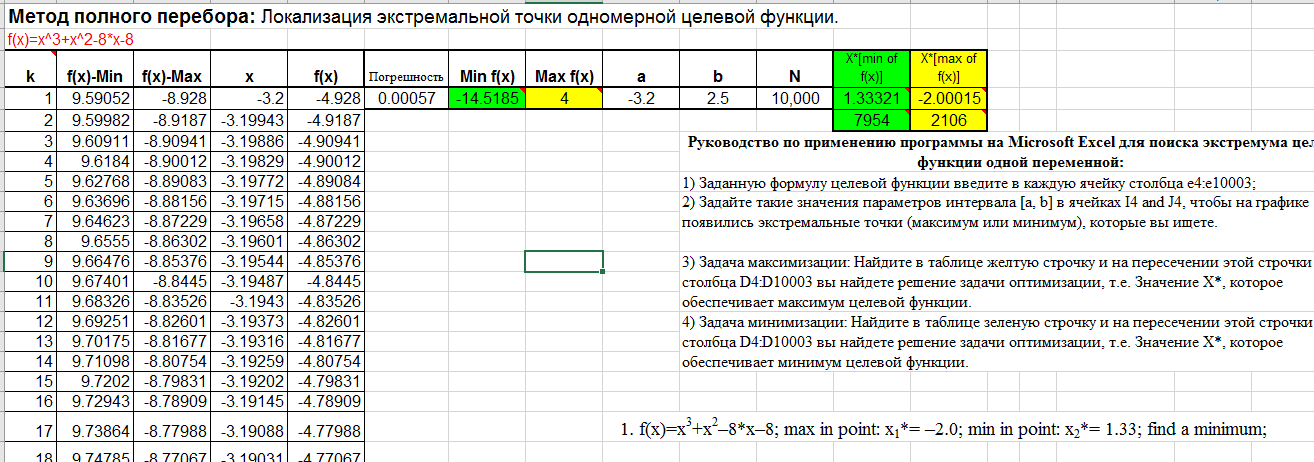
}

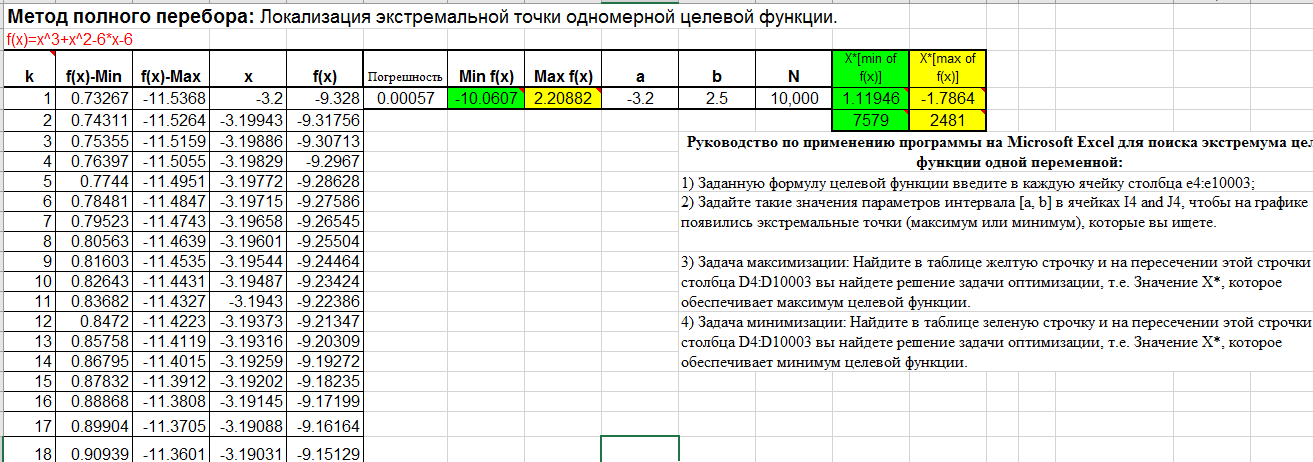
}

}

**Тесты для проверки ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, РЕАЛИЗУЮЩЕГО МЕТОД ПОРАЗРЯДНОГО ПРИБЛИЖЕНИЯ**

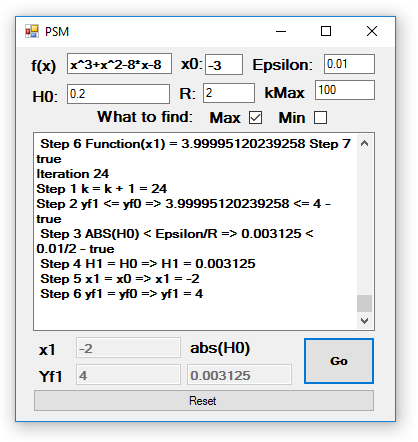
Эталон:



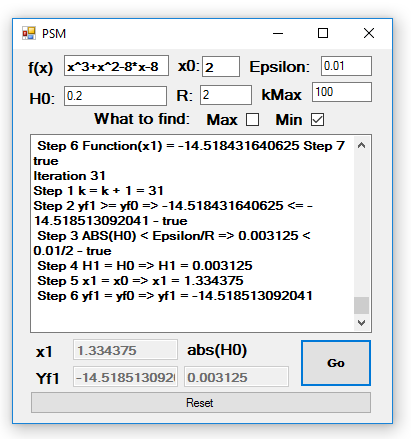


Тест №1:

max:

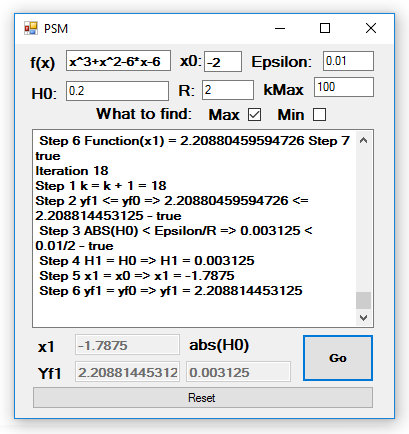


min:

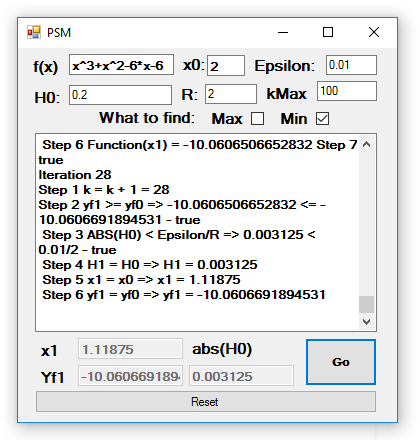


Тест №2:

max:



min:



# **Практическая работа №3**

# **Часть №1**

«Разработка ПО для поиска минимума ***нелинейной унимодальной функции***

на основе итерационного метода: ***Golden section method***

**Что дано:**

* ***Спецификация проблемы*** поиска минимума ***нелинейной унимодальной функции*** f(x);
* ***Спецификация*** итерационного метода (***Golden section method***) для нахождения минимума ***нелинейной унимодальной функции***
* ***Блок-схема*** Golden section method;
* ***Интерфейсная форма*** системы поиска минимума ***нелинейной унимодальной функции***, реализующей итерационный метод —***Golden section method***;
* ***Тесты*** для проверки ПО.

**Что требуется:**

* Разработать ***проект*** ПО для поиска минимума произвольной ***нелинейной унимодальной функции*** ***f(x)*** для произвольной заданной допустимой погрешности;
* Сконструировать систему, реализующую итерационный метод Golden section Method на основе использования парсера;
* Разработать ***код*** ПО для поиска минимума произвольной нелинейной функции f(x) для произвольной заданной допустимой погрешности;
* Провести валидацию системы – доказать идентичность результатов решения задач с помощью разработанного ПО заданным тестам.

# **Часть №2**

**Наименование работы –** Нахождение минимума нелинейной функции *методом золотого сечения*

**СПЕЦИФИКАЦИЯ ПРОБЛЕМЫ №2: Нахождение МИНИМУМА нелинейной функции методом золотого сечения**

* Найти минимум произвольной нелинейной функции

с заданной допустимой погрешностью ***Tolerance*** методом золотого сечения. Нелинейная функция имеет ***произвольный*** аналитический вид, составленный из математических функций (полиномов различных степеней, тригонометрических – sin(x), cos(x), exp(x), ln(x), log(x) и. т. д.), который имеет математический смысл, и для которой существует хотя бы одно решение задачи.

**СПЕЦИФИКАЦИЯ (Описание) метода золотого сечения**

Метод относится к последовательным стратегиям и является одним из вариантов метода исключения отрезков. Алгоритм опирается на анализ значений функции в двух точках, являющихся точками золотого сечения текущего интервала неопределенности. Исключение отрезка в данном случае выполняется так же, как и в методе дихотомии. При этом с учетом свойств золотого сечения на каждой итерации, кроме первой, требуется только одно новое вычисление функции.

**Описание алгоритма решения проблемы в виде пошаговой итерационной процедуры**

1. **Шаг 1.** Задаются начальные границы отрезка a,\;b\! и точность \varepsilon\!.
2. **Шаг 2.** Рассчитывают начальные точки деления: x_1 = b-\frac{(b-a)}{\varphi},\quad x_2 = a+\frac{(b-a)}{\varphi}\! и значения в них [целевой функции](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F): y_1=f(x_1),\;y_2=f(x_2)\!.
   * Если y_1 \ge y_2\! (для поиска max изменить неравенство на y_1 \le y_2\!), то  a=x_1, x_1 = x_2, x_2 = a+\frac{(b-a)}{\varphi}\!
   * Иначе b=x_2, x_2 = x_1, x_1 = b-\frac{(b-a)}{\varphi}\!.
3. **Шаг 3.**
   * Если |b-a|<\varepsilon \!, то x=\frac{a+b}{2}\! и останов.
   * Иначе возврат к шагу 2.

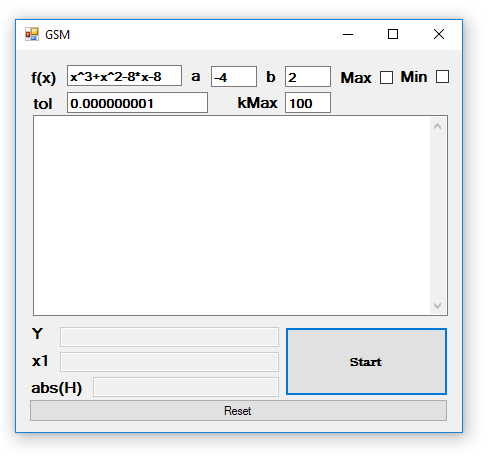
Алгоритм взят из источника: *Джон Г.Мэтьюз, Куртис Д.Финк.* "Численные методы. Использование MATLAB". — М, СПб: "Вильямс", 2001. — 716 с.

**Стадии *проектирования* системы для поиска минимума нелинейной функции , реализующей метод золотого сечения:**

Стадия №.1: Разработка блок-схемы метода золотого сечения



Стадия №.2: Проектирование интерфейса системы, реализующей метод золотого сечения



Документирование процесса задания свойств элементов интерфейсной формы системы.

|  |  |  |
| --- | --- | --- |
| **Control** | **Property** | **Setting** |
| TextBox1 | Desing (Name) | InFx |
| TextBox1 | Appearance (Text) | x^3+x^2-8\*x-8 |
| TextBox2 | Desing (Name) | InA |
| TextBox2 | Appearance (Text) | -4 |
| TextBox3 | Design (Name) | InB |
| TextBox3 | Appearance (Text) | 2 |
| TextBox4 | Design (Name) | InTol |
| TextBox4 | Appearance (Text) | 0.000000001 |
| TextBox5 | Design (Name) | InKMax |
| TextBox5 | Appearance (Text) | 100 |
| Label1 | Design (Name) | FuncLabel |
| Label1 | Appearance (Text) | f(x) |
| Label2 | Design (Name) | InALabel |
| Label2 | Appearance (Text) | a |
| Label3 | Design (Name) | InBLabel |
| Label3 | Appearance (Text) | b |
| Label4 | Design (Name) | InTolLabel |
| Label4 | Appearance (Text) | tol |
| Label5 | Design (Name) | InKMaxLabel |
| Label5 | Appearance (Text) | kMax |
| CheckBox1 | Design (Name) | maxCheckBox |
| CheckBox1 | Appearance (Text) | Max |
| CheckBox2 | Design (Name) | minCheckBox |
| CheckBox2 | Appearance (Text) | Min |
| Button1 | Design (Name) | Start\_Btn |
| Button1 | Appearance (Text) | Start |
| Button2 | Design (Name) | Reset\_Btn |
| Button2 | Appearance (Text) | Reset |
| TextBox6 | Design (Name) | Calcs |
| Label6 | Appearance (Text) | x1 |
| Label6 | Design (Name) | OutX1Label |
| Label7 | Appearance (Text) | Y |
| Label7 | Design (Name) | OutYLabel |
| Label8 | Appearance (Text) | abs(b-a) |
| Label8 | Design (Name) | OutAbsBALabel |
| Textbox7 | Design (Name) | OutY |
| Textbox7 | **Behavior (Enabled)** | False |
| Textbox8 | Design (Name) | OutX |
| Textbox8 | **Behavior (Enabled)** | False |
| TextBox9 | Design (Name) | OutAbsBA |
| TextBox9 | **Behavior (Enabled)** | False |

**Стадии *конструирования* системы для поиска минимума нелинейной функции , реализующей метод золотого сечения:** “GSM.cs”

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using info.lundin.math;

namespace FinalSolution

{

public partial class GSM : Form

{

public ExpressionParser parser = new ExpressionParser();

public void GSM\_Load(object sender, EventArgs e)

{

parser.Values.Add("x", 0);

}

public double func(double x)

{

parser.Values.Remove("x");

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

return parser.Parse(InFx.Text);

}

//x^3+x^2-8\*x-8

public GSM()

{

InitializeComponent();

}

public bool getError()

{

bool err = false;

Calcs.Text = "";

if ((!minCheckBox.Checked && !maxCheckBox.Checked) || (minCheckBox.Checked && maxCheckBox.Checked))

{

Calcs.AppendText("Choose what to find (min or max)" + Environment.NewLine);

err = true;

}

if (InFx.Text == "")

{

Calcs.AppendText("There are no function to solve (f(x))" + Environment.NewLine);

err = true;

}

if (InA.Text == "")

{

Calcs.AppendText("Enter left point (A)" + Environment.NewLine);

err = true;

}

if (InB.Text == "")

{

Calcs.AppendText("Enter right point (B)" + Environment.NewLine);

err = true;

}

if (InTol.Text == "")

{

Calcs.AppendText("Enter tolerance value (tol)" + Environment.NewLine);

err = true;

}

if(InKMax.Text == "")

{

Calcs.AppendText("Enter maximum value of iterations (kMax)" + Environment.NewLine);

err = true;

}

return err;

}

private void Start\_Btn\_Click(object sender, EventArgs e)

{

if (!getError())

{

double a = double.Parse(InA.Text, System.Globalization.NumberStyles.Any), b = double.Parse(InB.Text, System.Globalization.NumberStyles.Any), tol = double.Parse(InTol.Text, System.Globalization.NumberStyles.Any), x1, x2, fx1, fx2, r;

int k = 0, kMax = int.Parse(InKMax.Text);

Calcs.Text = "";

r = (Math.Sqrt(5.0) - 1) / 2;

Calcs.AppendText(" Step 1 r = (5^1/2 - 1) / 2 = " + r);

Calcs.AppendText(Environment.NewLine);

x1 = a + (1 - r) \* (b - a);

Calcs.AppendText(" Step 2 x1 = a + (1 - r) \* (b - a) => x1 = " + x1);

Calcs.AppendText(Environment.NewLine);

fx1 = func(x1);

Calcs.AppendText(" Step 3 f(x1) = " + fx1);

Calcs.AppendText(Environment.NewLine);

x2 = a + r \* (b - a);

Calcs.AppendText(" Step 4 x2 = a + r \* (b - a) => x2 = " + x2);

Calcs.AppendText(Environment.NewLine);

fx2 = func(x2);

Calcs.AppendText(" Step 5 f(x2) = ");

k = 0;

Calcs.AppendText(Environment.NewLine);

if (!minCheckBox.Checked)

{

do

{

if (k > 0)

{

Calcs.AppendText("true");

Calcs.AppendText(Environment.NewLine);

}

// Step 1

Calcs.AppendText("Iteration " + ++k);

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 1 k = k + 1 = " + k);

Calcs.AppendText(Environment.NewLine);

// Step 2

Calcs.AppendText(" Step 2 yf1 < yf2 => " + fx1 + " < " + fx2 + " - ");

if (fx1 < fx2) {

Calcs.AppendText("true");

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 3 a = x1 => a = " + x1);

a = x1;

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 4 x1 = x2 => x1 = " + x2);

x1 = x2;

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 5 f1 = f2 => f1 = " + fx2);

fx1 = fx2;

Calcs.AppendText(Environment.NewLine);

x2 = a + r \* (b - a);

Calcs.AppendText(" Step 6 x2 = a + r \* (b - a) => x2 = " + x2);

Calcs.AppendText(Environment.NewLine);

fx2 = func(x2);

}

else {

Calcs.AppendText("false");

Calcs.AppendText(Environment.NewLine);

b = x2;

Calcs.AppendText(" Step 3 b = x2 => b = " + b);

Calcs.AppendText(Environment.NewLine);

x2 = x1;

Calcs.AppendText(" Step 4 x2 = x1 => x2 = " + x2);

Calcs.AppendText(Environment.NewLine);

fx2 = fx1;

Calcs.AppendText(" Step 5 f2 = f1 => f2 = " + fx2);

Calcs.AppendText(Environment.NewLine);

x1 = a + (1 - r) \* (b - a);

Calcs.AppendText(" Step 6 x1 = a + r \* (b - a) => x2 = " + x1);

Calcs.AppendText(Environment.NewLine);

fx1 = func(x1);

Calcs.AppendText(" Step 7 f(x1) = " + fx1);

Calcs.AppendText(Environment.NewLine);

}

} while ((Math.Abs((b - a)) > tol) && k < kMax);

outX1.Text = x1.ToString();

outAbsBA.Text = Math.Abs(b - a).ToString();

SolY.Text = fx1.ToString();

}

else {

do

{

if (k > 0)

{

Calcs.AppendText("true");

Calcs.AppendText(Environment.NewLine);

}

// Step 1

Calcs.AppendText("Iteration " + ++k);

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 1 k = k + 1 = " + k);

Calcs.AppendText(Environment.NewLine);

// Step 2

Calcs.AppendText(" Step 2 yf1 < yf2 => " + fx1 + " < " + fx2 + " - ");

if (fx1 > fx2)

{

Calcs.AppendText("true");

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 3 a = x1 => a = " + x1);

a = x1;

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 4 x1 = x2 => x1 = " + x2);

x1 = x2;

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText(" Step 5 f1 = f2 => f1 = " + fx2);

fx1 = fx2;

Calcs.AppendText(Environment.NewLine);

x2 = a + r \* (b - a);

Calcs.AppendText(" Step 6 x2 = a + r \* (b - a) => x2 = " + x2);

Calcs.AppendText(Environment.NewLine);

fx2 = func(x2);

}

else

{

Calcs.AppendText("false");

Calcs.AppendText(Environment.NewLine);

b = x2;

Calcs.AppendText(" Step 3 b = x2 => b = " + b);

Calcs.AppendText(Environment.NewLine);

x2 = x1;

Calcs.AppendText(" Step 4 x2 = x1 => x2 = " + x2);

Calcs.AppendText(Environment.NewLine);

fx2 = fx1;

Calcs.AppendText(" Step 5 f2 = f1 => f2 = " + fx2);

Calcs.AppendText(Environment.NewLine);

x1 = a + (1 - r) \* (b - a);

Calcs.AppendText(" Step 6 x1 = a + r \* (b - a) => x2 = " + x1);

Calcs.AppendText(Environment.NewLine);

fx1 = func(x1);

Calcs.AppendText(" Step 7 f(x1) = " + fx1);

Calcs.AppendText(Environment.NewLine);

}

} while ((Math.Abs((b - a)) > tol) && k < kMax);

outX1.Text = x1.ToString();

outAbsBA.Text = Math.Abs(b - a).ToString();

SolY.Text = fx1.ToString();

}

}

}

private void button2\_Click\_1(object sender, EventArgs e)

{

minCheckBox.Checked = false;

maxCheckBox.Checked = false;

Calcs.Text = "";

outX1.Text = "";

outAbsBA.Text = "";

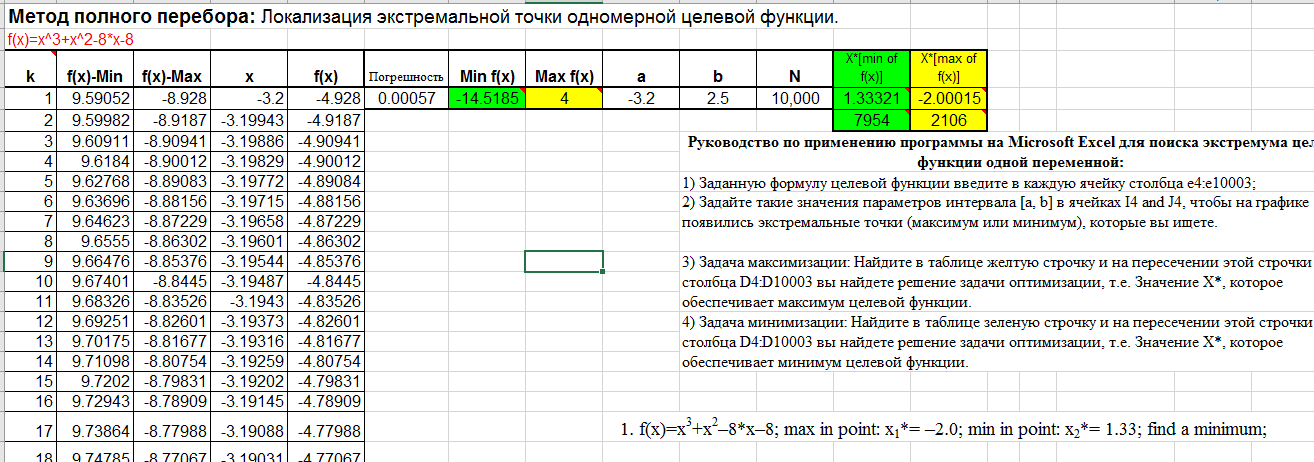
SolY.Text = "";

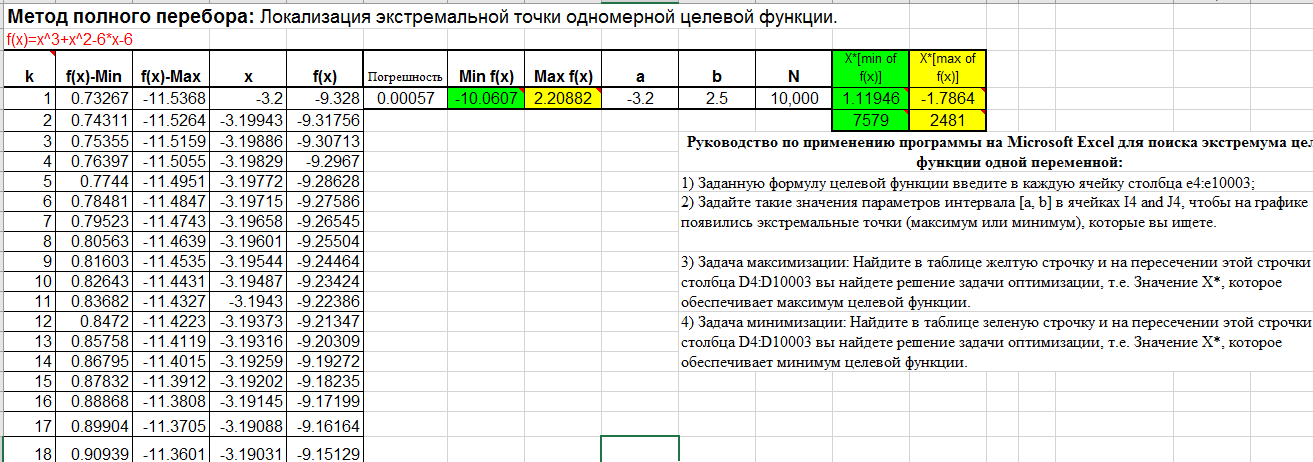
}

}

**Тесты для проверки ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, РЕАЛИЗУЮЩЕГО Метод золотого сечения.**

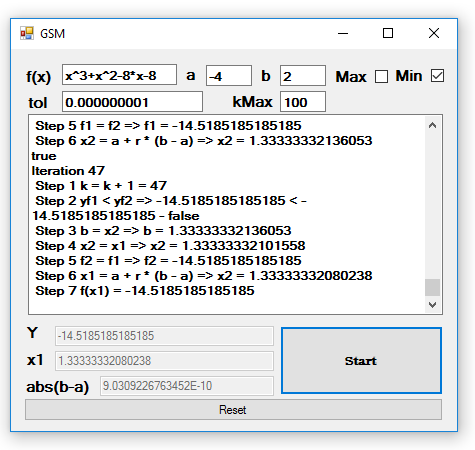
Эталон:



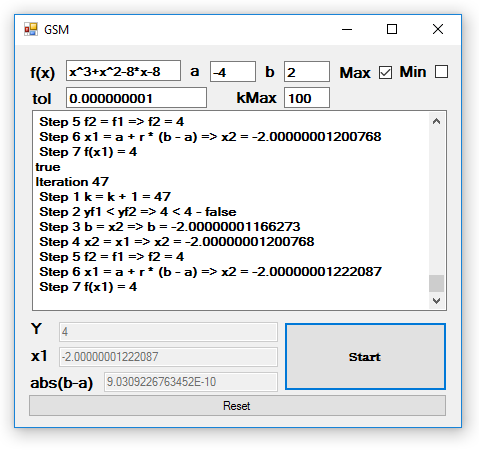


Тест №1

min

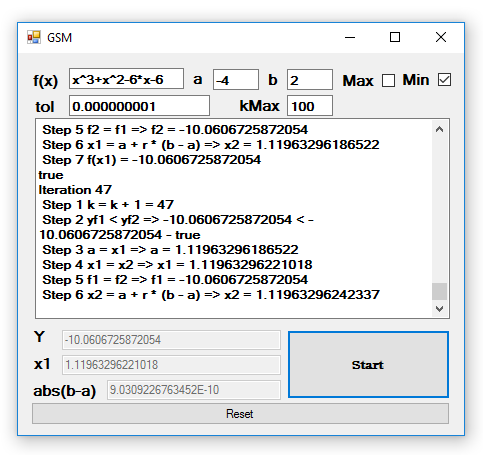


max

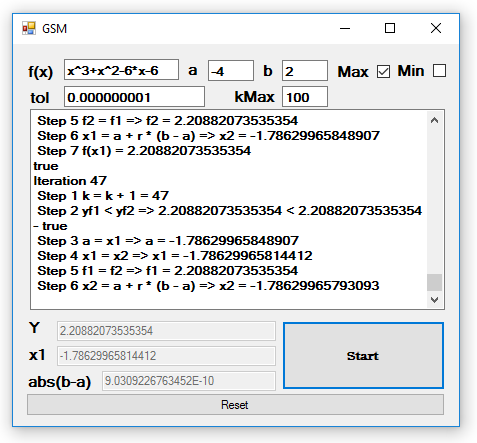


Тест №2

min



max



# **Практическая работа №4**

# **Часть №1**

«Разработка ПО для поиска минимума ***нелинейной унимодальной функции***

на основе итерационного метода: ***Newton method***

**Что дано:**

* ***Спецификация проблемы*** поиска минимума нелинейной функции f(x);
* ***Спецификация*** итерационного метода (***Newton method***) для нахождения минимума ***нелинейной унимодальной функции***
* ***Блок-схема*** Newton method;
* ***Интерфейсная форма*** системы поиска минимума ***нелинейной унимодальной функции***, реализующей итерационный метод — ***Newton method***;
* ***Тесты*** для проверки ПО.

**Что требуется:**

* Разработать ***проект*** ПО для поиска минимума произвольной ***нелинейной унимодальной функции*** ***f(x)*** для произвольной заданной допустимой погрешности;
* Сконструировать систему, реализующую итерационный метод Newton method на основе использования парсера;
* Разработать ***код*** ПО для поиска минимума произвольной нелинейной функции f(x) для произвольной заданной допустимой погрешности;
* Провести валидацию системы – доказать идентичность результатов решения задач с помощью разработанного ПО заданным тестам.

# **Часть №2**

**Наименование работы –** Нахождение минимума нелинейной функции *методом Ньютона*

**СПЕЦИФИКАЦИЯ ПРОБЛЕМЫ №2: Нахождение МИНИМУМА нелинейной функции методом НЬЮТОНА**

* Найти минимум произвольной нелинейной функции

с заданной допустимой погрешностью ***Tolerance*** методом Ньютона. Нелинейная функция имеет ***произвольный*** аналитический вид, составленный из математических функций (полиномов различных степеней, тригонометрических – sin(x), cos(x), exp(x), ln(x), log(x) и. т. д.), который имеет математический смысл, и для которой существует хотя бы одно решение задачи.

**СПЕЦИФИКАЦИЯ (Описание) метода НЬЮТОНА**

Метод Ньютона относится к *методам второго порядка*, поскольку для вычисления каждой итерации требуется знание второй производной функции *f*. Метод Ньютона, с одной стороны, может сходиться с более высоким чем градиентный метод порядком, а, с другой стороны, для его сходимости требуются достаточно хорошие начальные приближения. Метод Ньютона требует большего объема вычислительной работы, поскольку приходится вычислять вторые производные функции *f*.

**Описание алгоритма решения проблемы в виде пошаговой итерационной процедуры**

1. Задается начальное приближение x_0
2. Чтобы численно решить уравнение f(x)=0\![методом простой итерации](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%B9_%D0%B8%D1%82%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8), его необходимо привести к следующей форме: x=\varphi(x)\!, где \varphi\! —[сжимающее отображение](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B8%D0%BC%D0%B0%D1%8E%D1%89%D0%B5%D0%B5_%D0%BE%D1%82%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5).
3. Для наилучшей [сходимости](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82%D1%8C_%D1%81%D1%85%D0%BE%D0%B4%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8) метода в точке очередного приближения x^*\! должно выполняться условие \varphi'(x^*)=0\!.
4. Решение данного уравнения ищут в виде \varphi(x)=x+\alpha(x)f(x)\!, тогда:

\varphi'(x^*)=1+\alpha'(x^*)f(x^*)+\alpha(x^*) f'(x^*)=0.\!

1. В предположении, что точка приближения «достаточно близка» к корню \tilde{x}\!, и что заданная функция [непрерывна](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BE%D1%82%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) (f(x^*)\approx f(\tilde{x})=0)\!, окончательная формула для \alpha(x)\! такова:

\alpha(x)=-\frac{1}{f'(x)}.\!

1. С учётом этого [функция](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) \varphi(x)\! определяется:

\varphi(x)=x-\frac{f(x)}{f'(x)}.\!

1. При некоторых условиях эта функция в окрестности корня осуществляет сжимающее отображение[[1]](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD%D0%B0#cite_note-1), и алгоритм нахождения численного решения уравнения f(x)=0\! сводится к итерационной процедуре вычисления:

x_{n+1}=x_{n}-\frac{f(x_n)}{f'(x_n)}.\!

Поиск завершен.

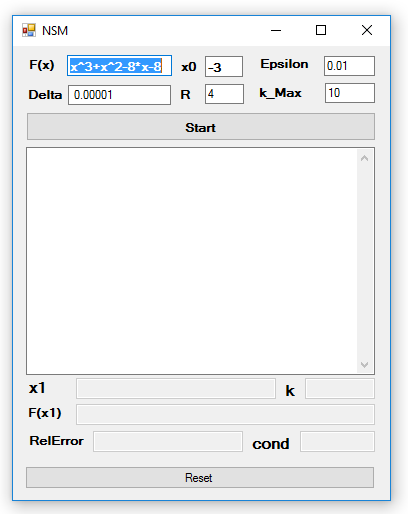
**Стадии *проектирования* системы для поиска минимума нелинейной функции , реализующей метод Ньютона:**

Стадия №.1: Разработка блок-схемы метода Ньютона





Стадия №.2: Проектирование интерфейса системы, реализующей метод Ньютона



Документирование процесса задания свойств элементов интерфейсной формы системы.

|  |  |  |
| --- | --- | --- |
| **Control** | **Property** | **Setting** |
| TextBox1 | Desing (Name) | InFx |
| TextBox1 | Appearance (Text) | x^3+x^2-8\*x-8 |
| TextBox2 | Desing (Name) | InX0 |
| TextBox2 | Appearance (Text) | -3 |
| TextBox3 | Desing (Name) | InEpsilon |
| TextBox3 | Appearance (Text) | 0.01 |
| TextBox4 | Design (Name) | InDelta |
| TextBox4 | Appearance (Text) | 0.00001 |
| TextBox5 | Design (Name) | InR |
| TextBox5 | Appearance (Text) | 4 |
| TextBox6 | Design (Name) | InKMax |
| TextBox6 | Appearance (Text) | 100 |
| Label1 | Design (Name) | InFxLabel |
| Label1 | Appearance (Text) | f(x) |
| Label2 | Design (Name) | InX0Label |
| Label2 | Appearance (Text) | x0 |
| Label3 | Design (Name) | InEpsilonLabel |
| Label3 | Appearance (Text) | Epsilon |
| Label4 | Design (Name) | InDeltaLabel |
| Label4 | Appearance (Text) | Delta |
| Label5 | Design (Name) | InRLabel |
| Label5 | Appearance (Text) | R |
| Label6 | Design (Name) | InKMaxLabel |
| Label6 | Appearance (Text) | k\_Max |
| Button1 | Design (Name) | StartBtn |
| Button1 | Appearance (Text) | Start |
| Button2 | Design (Name) | ResetBtn |
| Button2 | Appearance (Text) | Reset |
| TextBox7 | Design (Name) | Calcs |
| Label8 | Appearance (Text) | x1 |
| Label8 | Design (Name) | OutX1Label |
| Label9 | Appearance (Text) | F(x1) |
| Label9 | Design (Name) | OutFx1Label |
| Label10 | Appearance (Text) | RelError |
| Label10 | Design (Name) | OutRelErrorLabel |
| Label11 | Appearance (Text) | k |
| Label11 | Design (Name) | OutKLabel |
| Label12 | Appearance (Text) | cond |
| Label12 | Design (Name) | OutCondLabel |
| Textbox9 | Design (Name) | OutX1 |
| Textbox9 | **Behavior (Enabled)** | False |
| Textbox10 | Design (Name) | OutFx1 |
| Textbox10 | **Behavior (Enabled)** | False |
| Textbox11 | Design (Name) | OutRelError |
| Textbox11 | **Behavior (Enabled)** | False |
| Textbox12 | Design (Name) | OutRelError |
| Textbox12 | **Behavior (Enabled)** | False |
| Textbox11 | Design (Name) | OutCond |
| Textbox11 | **Behavior (Enabled)** | False |
| Textbox12 | Design (Name) | OutK |
| Textbox12 | **Behavior (Enabled)** | False |
| Textbox13 | Design (Name) | OutDer1 |
| Textbox13 | **Behavior (Enabled)** | False |
| Textbox14 | Design (Name) | OutDer2 |
| Textbox14 | **Behavior (Enabled)** | False |

**Стадии *конструирования* системы для поиска минимума нелинейной функции , реализующей метод Ньютона: ”NSM.cs”**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Windows.Forms;

using info.lundin.math;

using aziretParser;

namespace FinalSolution

{

public partial class NSM : Form

{

public ExpressionParser parser = new ExpressionParser();

public void GSM\_Load(object sender, EventArgs e)

{

parser.Values.Add("x", 0);

}

public double diff\_func(double x)

{

parser.Values.Remove("x");

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

String DerStrFunc = ParserDecimal.ReturnDerivative(inFx.Text);

return parser.Parse(DerStrFunc); }

public double double\_diff\_func(double x)

{

parser.Values.Remove("x");

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

String DerStrFunc = ParserDecimal.ReturnDerivative(inFx.Text);

DerStrFunc = ParserDecimal.ReturnDerivative(DerStrFunc);

return parser.Parse(DerStrFunc);

}

public void NewLine()

{ Calcs.AppendText(Environment.NewLine);

}

public double func(double x)

{

parser.Values.Remove("x");

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

return parser.Parse(inFx.Text);

}

public bool getError() {

bool err = false;

Calcs.Text = "";

if (inFx.Text == "") {

Calcs.AppendText("There are no function to solve (f(x))" + Environment.NewLine);

err = true; }

if (inDelta.Text == "") {

Calcs.AppendText("Enter enter Delta" + Environment.NewLine);

err = true; }

if (inX0.Text == "") {

Calcs.AppendText("Enter first point (x0)" + Environment.NewLine);

err = true; }

if (inR.Text == "") {

Calcs.AppendText("Enter R parameter" + Environment.NewLine);

err = true; }

if(inEpsilon.Text == "") {

Calcs.AppendText("Enter Epsilon" + Environment.NewLine);

err = true; }

if (InKMax.Text == "")

{

Calcs.AppendText("Enter maximum value of iterations (kMax)" + Environment.NewLine);

err = true;

}

return err;

}

//x^3+x^2-8\*x-8

public NSM()

{

InitializeComponent();

}

public bool sign(double num1, double num2)

{

return num1 >= 0 && num2 >= 0 || num1 < 0 && num2 < 0;

}

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

{

if (!getError())

{

double RelError, DP = 0, DPO = 0, dfx1, ddfx1, x0 = double.Parse(inX0.Text, System.Globalization.NumberStyles.Any), R = double.Parse(inR.Text), Delta = double.Parse(inDelta.Text), fx1, fx0 = func(x0), dfx0, ddfx0, Epsilon = double.Parse(inEpsilon.Text, System.Globalization.NumberStyles.Any), x1;

int k = 0, cond = 0, step = 0, d = 0, k\_Max = int.Parse(InKMax.Text);

Calcs.AppendText("Step 1 cond = 0");

Calcs.AppendText(Environment.NewLine);

Calcs.AppendText("Step 2 yfx0 = f(x0) = " + fx0);

NewLine();

dfx0 = diff\_func(x0);

Calcs.AppendText("Step 3 dfx0 = f'(x0) = " + dfx0);

NewLine();

ddfx0 = double\_diff\_func(x0);

Calcs.AppendText("Step 4 ddfx0 = f''(x0) = " + ddfx0);

NewLine();

k = 0;

do{

if (k > 0) Calcs.AppendText("true");

Calcs.AppendText("Iteration " + (k + 1));

NewLine();

Calcs.AppendText(" Step " + ++step + " k = k + 1 => k = " + k + " + 1 = " + ++k);

NewLine();

Calcs.AppendText(" Step 2 abs(ddfx0) <= Epsilon => " + Math.Abs(ddfx0) + " <= " + Epsilon + " - ");

if (Math.Abs(ddfx0) <= Epsilon) {

Calcs.AppendText("true");

NewLine();

Calcs.AppendText(" Step 3 cond = 1");

NewLine();

cond = 1;

} else {

Calcs.AppendText("false");

NewLine();

DP = dfx0 / ddfx0;

Calcs.AppendText(" Step 3 DP = dfx0 / ddfx0 => DP = " + dfx0 + " / " + ddfx0 + " = " + DP);

NewLine(); }

Calcs.AppendText(" Step 4 k == 1 => " + k + " = 1 - ");

if (k == 1) {

Calcs.AppendText("true");

NewLine();

d = 1;

DPO = DP;

Calcs.AppendText(" Step 5 DPO = DP => DPO = " + DP);

NewLine();

} else {

Calcs.AppendText("false");

NewLine(); }

Calcs.AppendText(" Step " + (4 + d) + " sign(DPO) == sign(DP) => sign(" + DPO + ") == sign(" + DP + ") - ");

if (sign(DPO, DP)) {

Calcs.AppendText("true");

NewLine();

x1 = x0 - DP;

Calcs.AppendText(" Step " + (5 + d) + " x1 = x0 - DP => " + "x1 = " + x0 + " - " + DP + " = " + x1);

NewLine();

} else { Calcs.AppendText("false");

NewLine();

x1 = x0 - DP / R;

Calcs.AppendText(" Step " + (6 + d) + " x1 = x0 - DP / R => " + "x1 = " + x0 + " - " + DP + " / " + R + " = " + x1);

NewLine(); }

DPO = DP;

Calcs.AppendText(" Step " + (7 + d) + " DPO = DP => DPO = " + DPO);

NewLine();

fx1 = func(x1);

Calcs.AppendText(" Step " + (8 + d) + " yf1 = f(x1) = " + fx1);

dfx1 = diff\_func(x1);

Calcs.AppendText(" Step " + (9 + d) + " dfx1 = f'(x1) = " + dfx1);

NewLine();

ddfx1 = double\_diff\_func(x1);

Calcs.AppendText(" Step " + (10 + d) + " ddfx1 = f''(x1) = " + ddfx1);

NewLine();

RelError = 2 \* Math.Abs(DP) / (Math.Abs(x1) + Epsilon);

Calcs.AppendText(" Step " + (11 + d) + " RelError = 2 \* abs(DP) / (abs(x1) + Epsilon) => RelError = 2 \* abs(" + DP + ") / (abs(" + x1 + ") + " + Epsilon + ") = " + RelError);

NewLine();

Calcs.AppendText(" Step " + (12 + d) + " RelError < Delta => " + RelError + " < " + Delta + " - "); if (RelError < Delta)

{

Calcs.AppendText("true");

NewLine();

d++;

Calcs.AppendText(" Step " + (12 + d) + " cond <> 1 => " + cond + " <> 1 - ");

if (cond != 1)

{

Calcs.AppendText("true");

NewLine();

d++;

Calcs.AppendText(" Step " + (12 + d) + " cond = 2");

NewLine();

cond = 2;

}

else

{

Calcs.AppendText("false");

NewLine();

}

}

else

{

Calcs.AppendText("false");

NewLine();

}

Calcs.AppendText(" Step " + (13 + d) + " x0 = x1 => x0 = " + x1 + "; dfx0 = dfx1 => dfx0 = " + dfx1 + "; ddfx0 = ddfx1 => ddfx0 = " + ddfx1 + ";");

NewLine();

x0 = x1; dfx0 = dfx1; ddfx0 = ddfx1;

Calcs.AppendText(" Step " + (14 + d) + " k < k\_Max & cond == 0 => " + k + " < " + k\_Max + " & " + cond + " == 0 - ");

NewLine();

} while (k < k\_Max && cond == 0);

Calcs.AppendText("false");

NewLine();

NoutX1.Text = x1.ToString();

outYf1.Text = fx1.ToString();

outK.Text = k.ToString();

outRelError.Text = String.Format(RelError.ToString(), "0e0");

outCond.Text = cond.ToString();

String DerStrFunc = ParserDecimal.ReturnDerivative(inFx.Text);

OutDer1.Text = erStrFunc;

OutDer2.Text = ParserDecimal.ReturnDerivative(DerStrFunc); } }

private void Reset\_Btn\_Click(object sender, EventArgs e) {

outCond.Text = "";

outK.Text = "";

outRelError.Text = "";

outYf1.Text = "";

NoutX1.Text = "";

Calcs.Text = "";

OutDer1.Text = "";

OutDer2.Text = "";

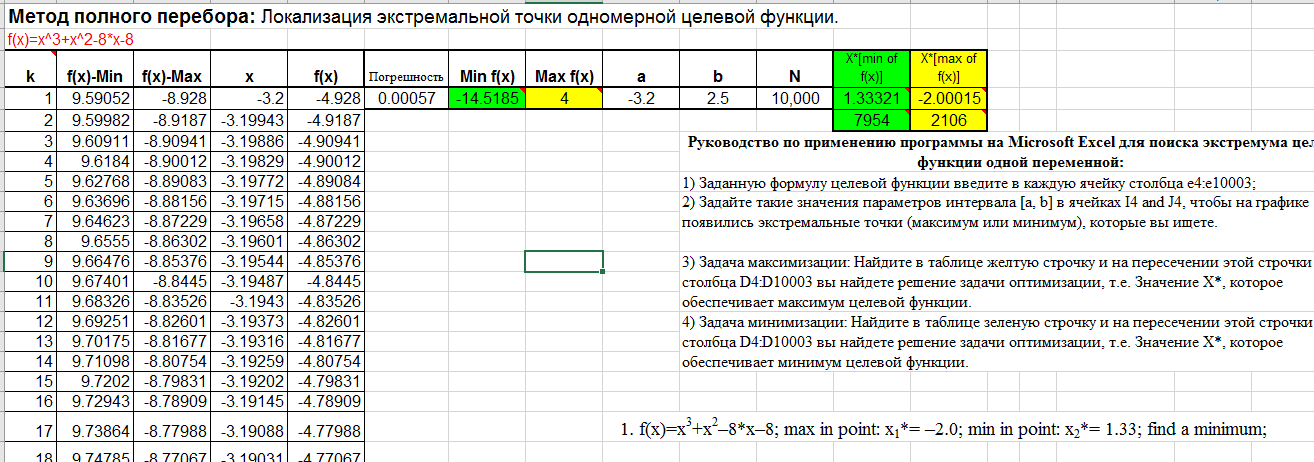
}

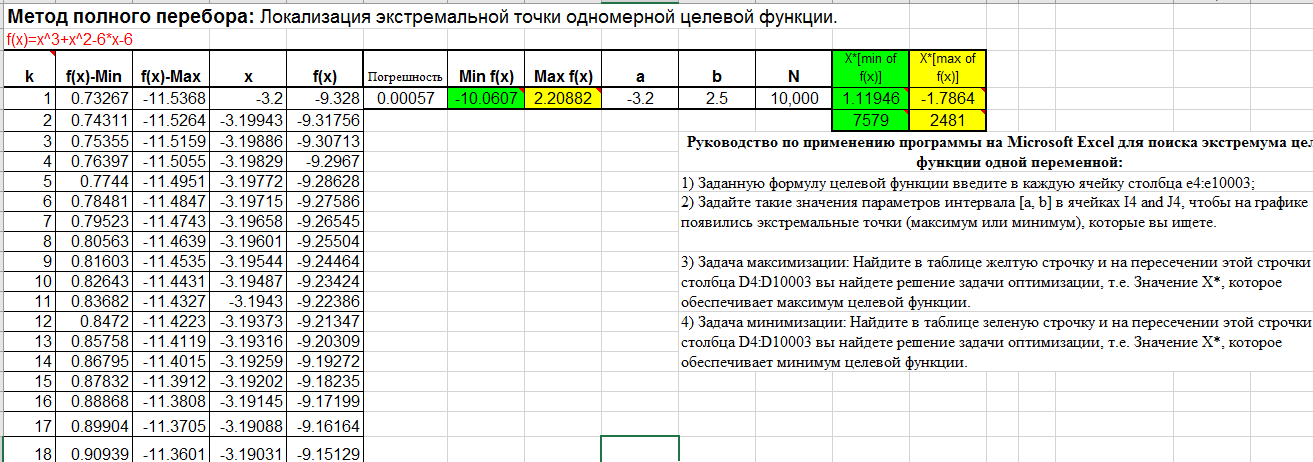
}

}

**Тесты**

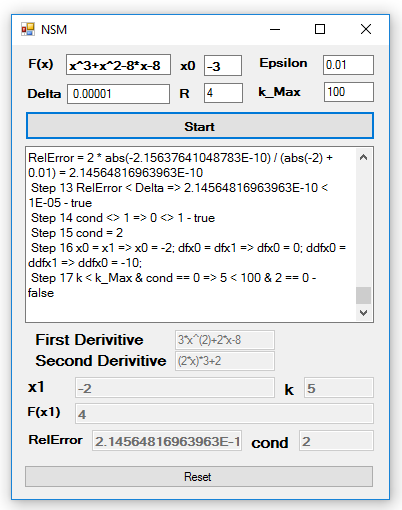
Эталон:



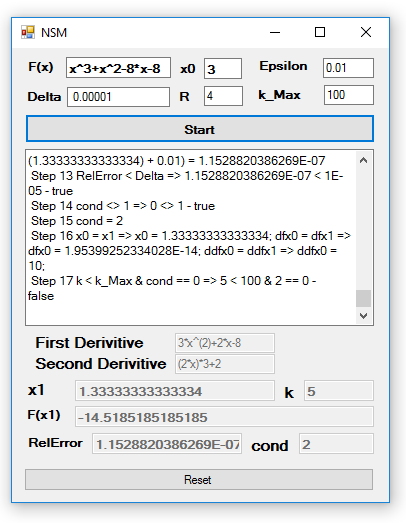


Тест №1

max:

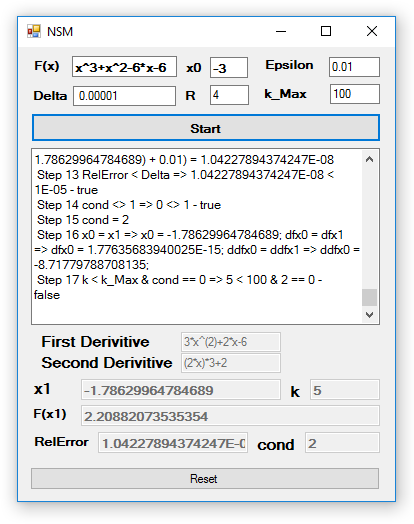


min:

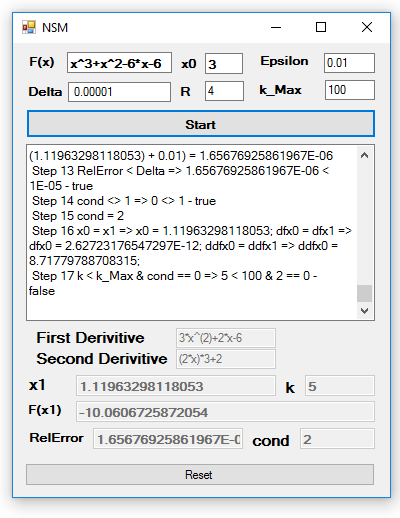


Тест №2

max:



min:



Подключение info.lundin.math; для парсирования функций

Зайдите на сайт info.lundin.com и скачайте парсер. Для подключения парсера к проекту необходимо прописать ссылку (Reference) на этот файл. Для этого зайдем в Обозреватель проекта, затем правой кнопкой по пункту References вызовем контекстное меню и в нем выберем пункт Add references. В появившемся окне нажмем кнопку Browse… и там находим скаченный info.lundin.dll файл.

AziretParser для нахождения производных я прилагаю (на одном уровне с отчетом). Подключается он ровно таким же способом…….