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

Бишкек 2016

**Содержание**

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

[**Часть №1** 3](#_Toc469472968)

[**Часть №2** 3](#_Toc469472969)

[**Раздел №1** 3](#_Toc469472970)

[**Раздел №2** 3](#_Toc469472971)

[**Раздел №3** 4](#_Toc469472972)

[**Раздел №4** 4](#_Toc469472973)

[**Раздел №5** 8](#_Toc469472974)

[**Раздел №6** 9](#_Toc469472975)

[**Практическая работа №2** 16](#_Toc469472976)

[**Часть №1** 16](#_Toc469472977)

[**Часть №2** 16](#_Toc469472978)

[**Раздел №1** 16](#_Toc469472979)

[**Раздел №2** 16](#_Toc469472980)

[**Раздел №3** 17](#_Toc469472981)

[**Раздел №4** 18](#_Toc469472982)

[**Раздел №5** 21](#_Toc469472983)

[**Раздел №6** 24](#_Toc469472984)

[**Практическая работа №3** 30](#_Toc469472985)

[**Часть №1** 30](#_Toc469472986)

[**Часть №2** 30](#_Toc469472987)

[**Раздел №1** 30](#_Toc469472988)

[**Раздел №2** 30](#_Toc469472989)

[**Раздел №3** 30](#_Toc469472990)

[**Раздел №4** 31](#_Toc469472991)

[**Раздел №5** 34](#_Toc469472992)

[**Раздел №6** 37](#_Toc469472993)

[**Практическая работа №4** 42](#_Toc469472994)

[**Часть №1** 42](#_Toc469472995)

[**Часть №2** 42](#_Toc469472996)

[**Раздел №1** 42](#_Toc469472997)

[**Раздел №2** 42](#_Toc469472998)

[**Раздел №3** 42](#_Toc469472999)

[**Раздел №4** 44](#_Toc469473000)

[**Раздел №5** 48](#_Toc469473001)

[**Раздел №6** 50](#_Toc469473002)

# **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**

## **Раздел №1**

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

## **Раздел №2**

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.

## **Раздел №3**

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

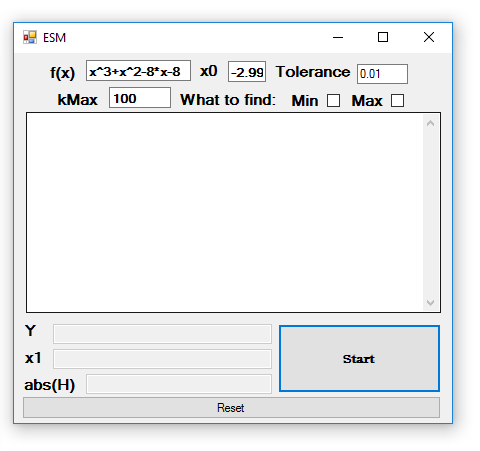
## **Раздел №4**

**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 |

## 

## **Раздел №5**

Стадия №3: Код 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);

}

//x^3+x^2-8\*x-8 max in point: x1\*= –2.0; min in point: x2\*= 1.33; find a minimum;

++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 = "";

}

}

}

## 

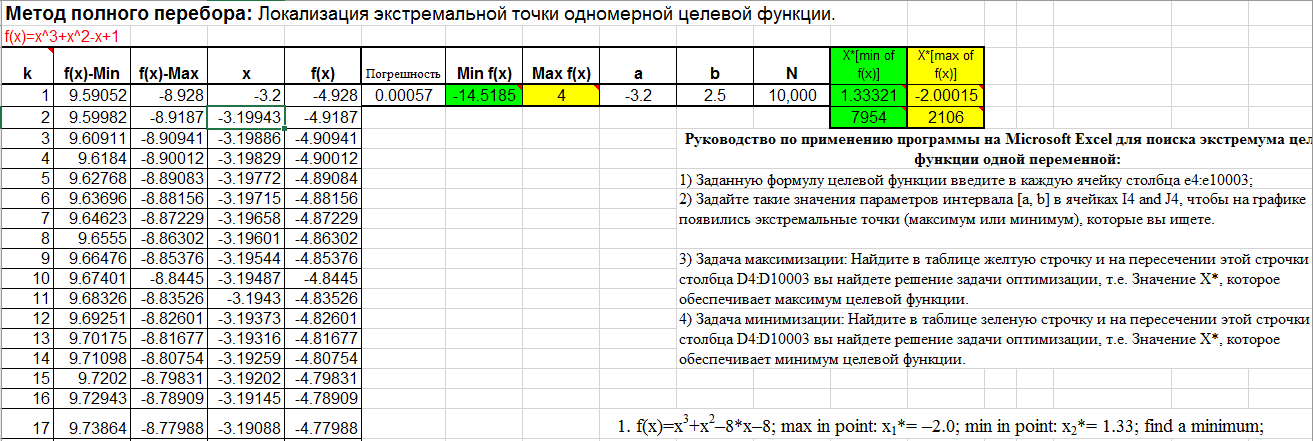
## **Раздел №6**

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

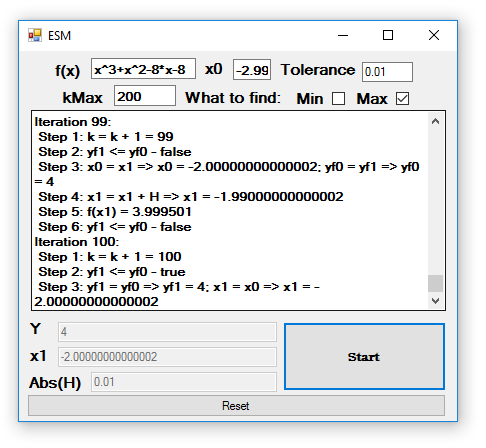
Тесты проводятся на основе программы, реализующей метод полного перебора в Excel MO\_LookingForOneOptPoint\_02.04.2012, разработанной преподавателем (приложено).

Тест №1

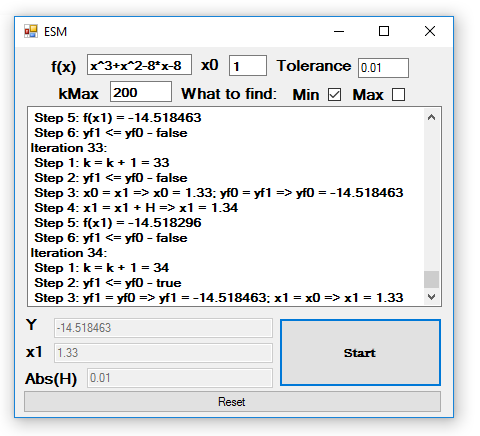
Эталон:



max:

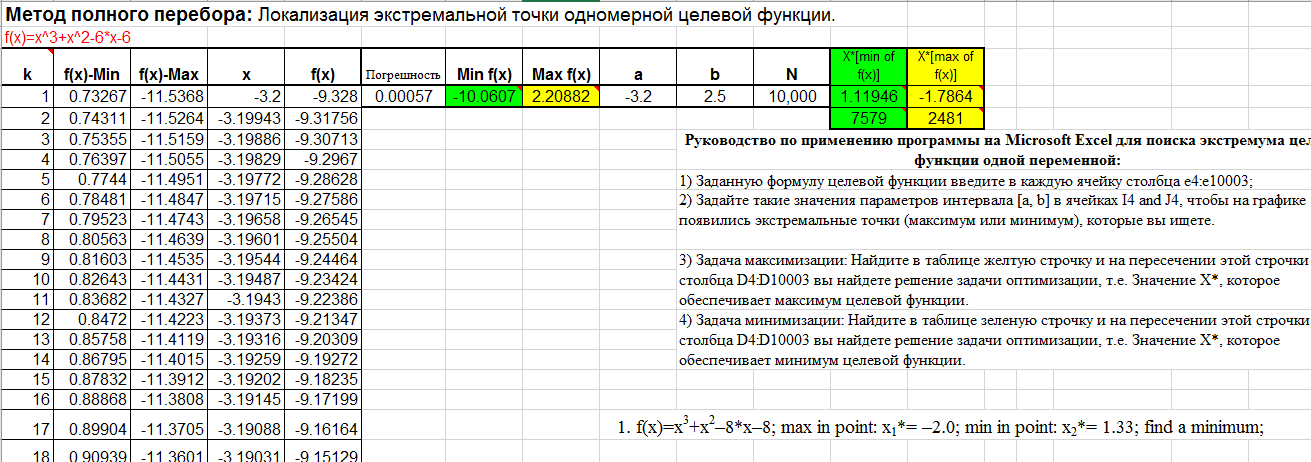


min:

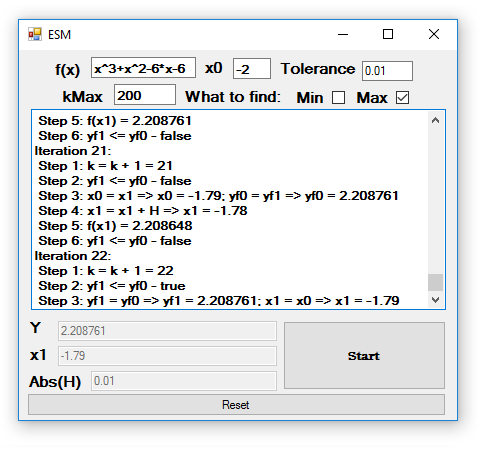


Тест №2

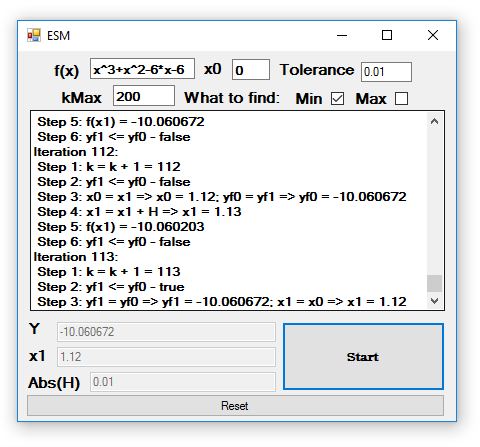
Эталон:



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**

## **Раздел №1**

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

## **Раздел №2**

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.

## **Раздел №3**

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

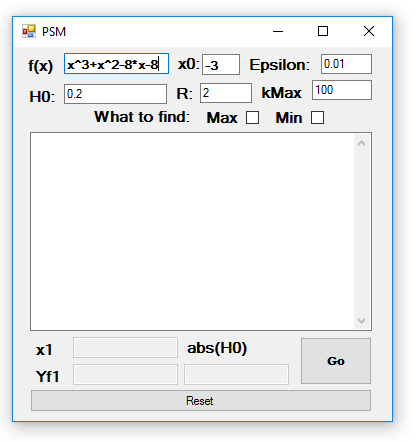
## **Раздел №4**

**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 |
| Button2 | Design (Name) | ButtonClear |
| Button2 | Appearance (Text) | Очистить |
| ProgressBar1 | Appearance (Text) | ProgressBar1 |
| Label7 | Appearance (Text) | X1 |
| Label7 | Design (Name) | ResultXLabel |
| Label8 | Appearance (Text) | F1 |
| Label8 | Design (Name) | fResultLabel |
| Label9 | Appearance (Text) | AmountIterations |
| Label9 | Design (Name) | AmountIterationsLabel |
| Label10 | Appearance (Text) | Time |
| Label10 | Design (Name) | TimeLabel |
| ProgressBar1 | Behavior (Visible) | False |
| ProgressBar1 | Design (Name) | ProgressBar1 |
| Textbox7 | Design (Name) | X1Box |
| Textbox7 | **Behavior (ReadOnly)** | True |
| Textbox8 | Design (Name) | F1Box |
| Textbox8 | **Behavior (ReadOnly)** | True |
| Textbox9 | Design (Name) | AmountIterationsBox |
| Textbox9 | **Behavior (ReadOnly)** | True |
| Textbox10 | Design (Name) | TimeBox |
| Textbox10 | **Behavior (ReadOnly)** | True |

## **Раздел №5**

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

Стадия №3: Код программы на C#.

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;

using System.Diagnostics;

namespace PoketSearch

{

public partial class PoketSearch : Form

{

Stopwatch time = new Stopwatch();

public double Function(double x1, string functio)

{

ExpressionParser parser = new ExpressionParser(); // объявляем парсер переменную

parser.Values.Add("x", x1);

// ((DoubleValue)parser.Values["x"]).Value = x1;

return parser.Parse(functio); // получаем значение функции из текст. поля

}

public PoketSearch()

{

InitializeComponent();

}

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

{

string function; // функция

double k\_max, tol, x0, b, r, x1, f1 = 0, f0 = 0, Time, eps, delta, tolerance, h0, h1, k = 0, flag = 0;

if (FuncBox.Text == "") MessageBox.Show("Введите функцию");

else if (X0Box.Text == "") MessageBox.Show("Введите начальную точку");

else if (h0Box.Text == "") MessageBox.Show("Введите параметр H(0)");

else if (RBox.Text == "") MessageBox.Show("Введите параметр R");

else if (ToleranceBox.Text == "" || Double.Parse(ToleranceBox.Text) <= Double.Parse("1e-16")) MessageBox.Show("Вы не ввели погрешность, или ввели погрешность меньшую 1е-15");

else if (kmaxBox.Text == "") MessageBox.Show("Введите количество итераций");

else

{

time.Start();

ExpressionParser parser = new ExpressionParser(); // объявляем парсер переменную

progressBar1.Value = 0;

progressBar1.Visible = true;

// progressBar1.value += 1;

function = FuncBox.Text;

x0 = Double.Parse(X0Box.Text.Trim().Replace(".", ",")); // получаем значение левой границы из текст. поля,преобразуем

tol = Double.Parse(ToleranceBox.Text.Trim().Replace(".", ",")); // получаем значение погрешности и текст. поля, преобразуем

k\_max = Double.Parse(kmaxBox.Text);

tolerance = Double.Parse(ToleranceBox.Text);

h0 = Double.Parse(h0Box.Text.Trim().Replace(".", ","));

r = Double.Parse(RBox.Text);

eps = Double.Parse("1e-15");

if (MinradioButton.Checked == true)

{

f0 = Function(x0, function);

h1 = h0;

x1 = x0 + h1;

f1 = Function(x1, function);

k = 0;

do

{

k += 1;

progressBar1.Increment(1);

if (f1 >= f0)

{

if (Math.Abs(h0) < tol / r)

{

h1 = h0;

x1 = x0;

f1 = f0;

flag = 1;

}

else

{

h1 = -h0 / r;

h0 = h1;

h1 = h0;

x0 = x1;

f0 = f1;

x1 = x0 + h1;

f1 = Function(x1, function);

}

}

else

{

h1 = h0;

x0 = x1;

f0 = f1;

x1 = x0 + h1;

f1 = Function(x1, function);

}

delta = Math.Abs(x1 - x0);

}

while (k < k\_max && flag != 1); //delta х1-х0

// tolerance - из текст поля

//

progressBar1.Value = progressBar1.Maximum;

X1Box.Text = x1.ToString();

F1Box.Text = f1.ToString();

AmountIterationsBox.Text = k.ToString();

time.Stop();

TimeSpan tm = time.Elapsed;

TimeBox.Text = String.Format("{0000}", tm.Milliseconds);

MessageBox.Show("Решение найдено!");

progressBar1.Visible = false;

}

if (MaxradioButton.Checked == true)

{

f0 = Function(x0, function);

h1 = h0;

x1 = x0 + h1;

f1 = Function(x1, function);

do

{

k += 1;

progressBar1.Increment(1);

if (f1 <= f0)

{

if (Math.Abs(h0) < tol / r)

{

h1 = h0;

x1 = x0;

f1 = f0;

flag = 1;

}

else

{

h1 = -h0 / r;

h0 = h1;

h1 = h0;

x0 = x1;

f0 = f1;

x1 = x0 + h1;

f1 = Function(x1, function);

}

}

else

{

h1 = h0;

x0 = x1;

f0 = f1;

x1 = x0 + h1;

f1 = Function(x1, function);

}

delta = Math.Abs(x1 - x0);

}

while ((k < k\_max) && flag != 1);//(Math.Abs(delta - tolerance) < eps)); //delta х1-х0

// tolerance - из текст поля

//

progressBar1.Value = progressBar1.Maximum;

X1Box.Text = x1.ToString();

F1Box.Text = f1.ToString();

AmountIterationsBox.Text = k.ToString();

time.Stop();

TimeSpan tm = time.Elapsed;

TimeBox.Text = String.Format("{0000}", tm.Milliseconds);

MessageBox.Show("Решение найдено!");

progressBar1.Visible = false;

}

}

}

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

{

progressBar1.Value = 0;

X1Box.Text = "";

F1Box.Text = "";

AmountIterationsBox.Text = "";

TimeBox.Text = "";

time.Reset();

}

}

}

## **Раздел №6**

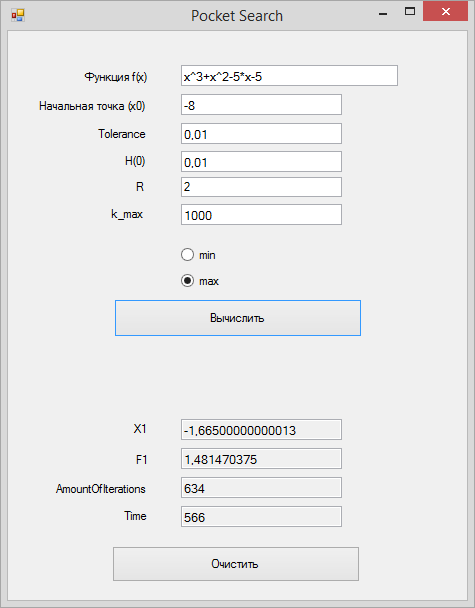
**Тесты для проверки ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, РЕАЛИЗУЮЩЕГО МЕТОД ПОРАЗРЯДНОГО ПРИБЛИЖЕНИЯ**

Тесты проводятся на основе вопросно-ответной веб-системы **Wolfram|Alpha**, имеющей математический раздел, где есть возможность находить максимум, или минимум.

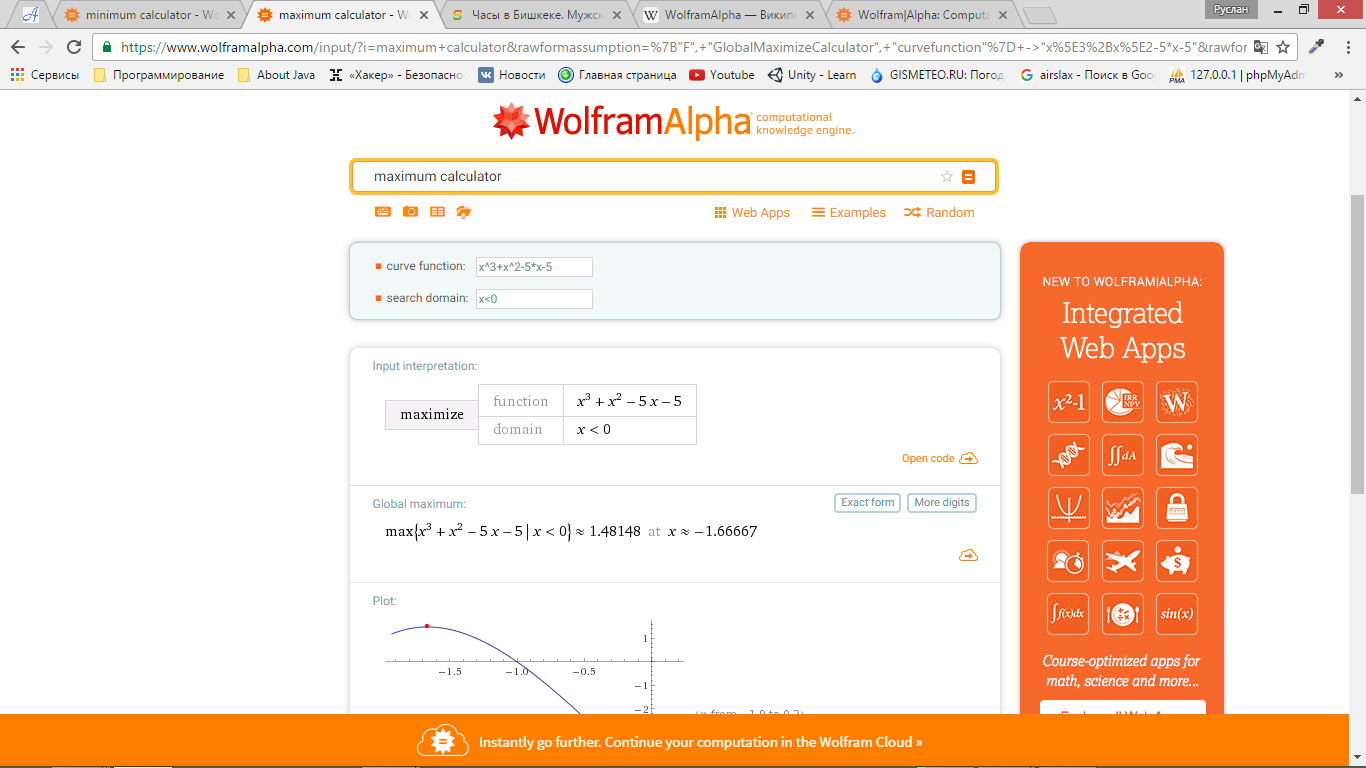
Максимум калькулятор – https://www.wolframalpha.com/input/?i=maximum+calculator

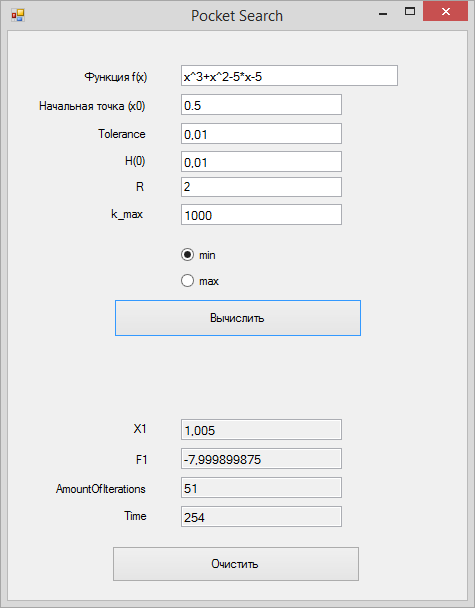
Минимум калькулятор – https://www.wolframalpha.com/input/?i=minimum+calculator

Тест №1:

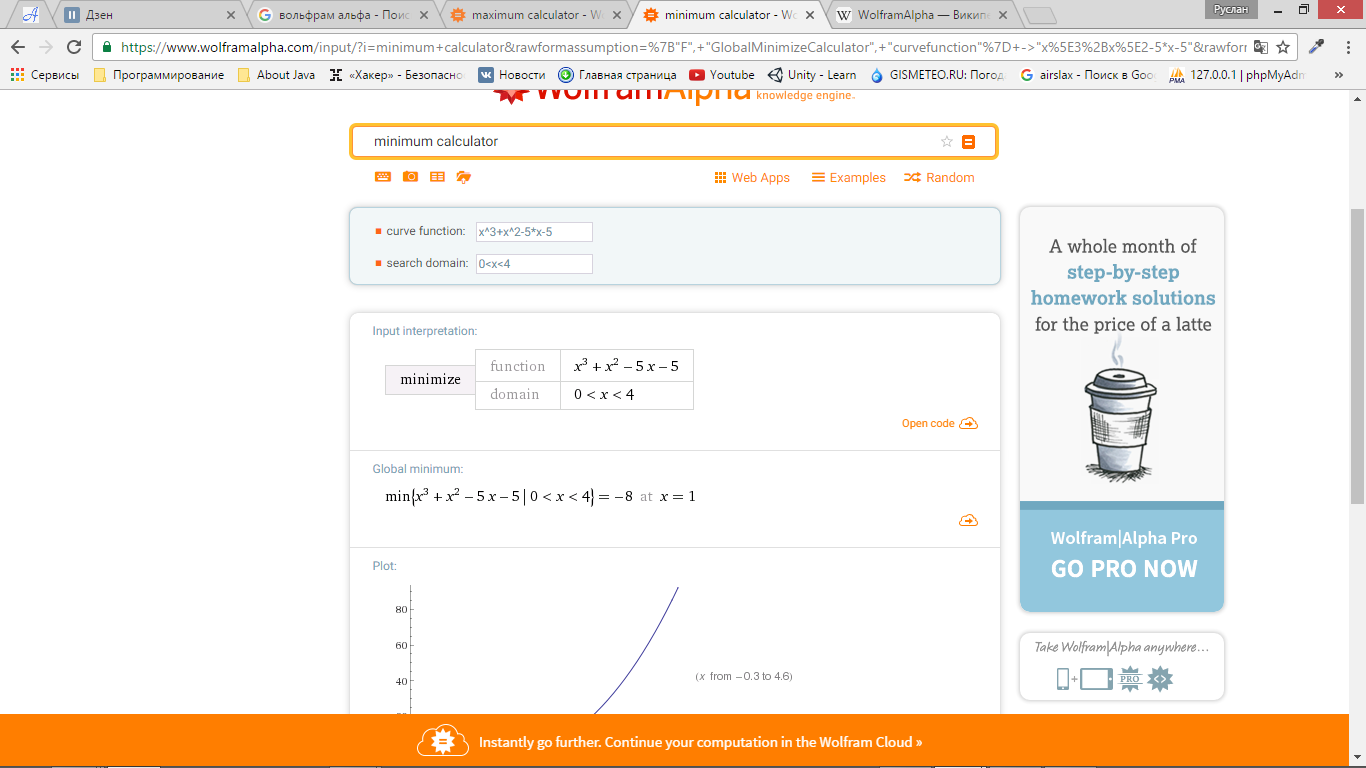


max:

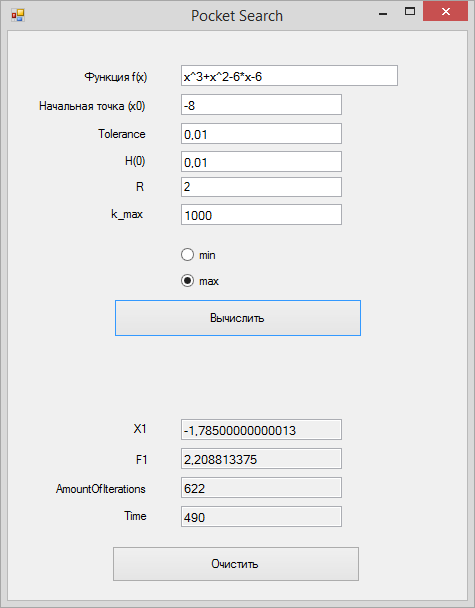




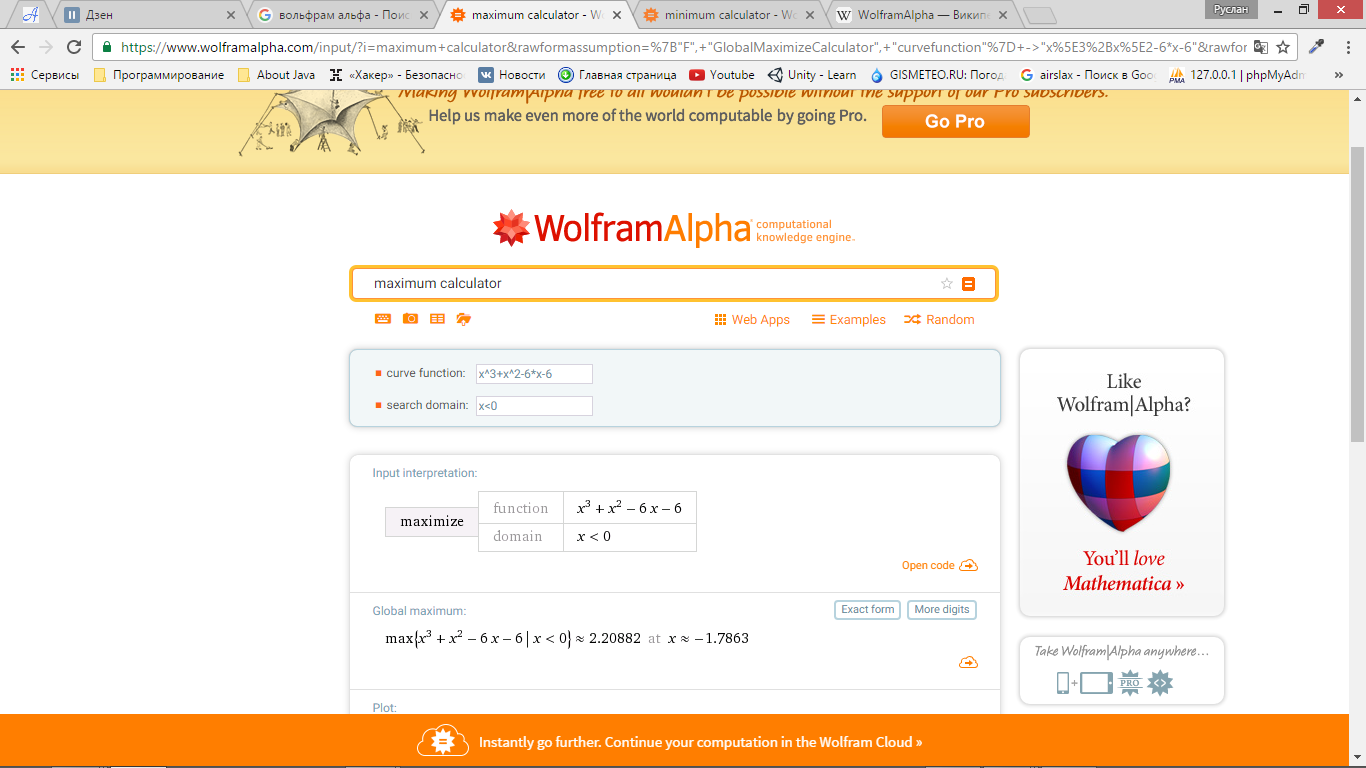
min:

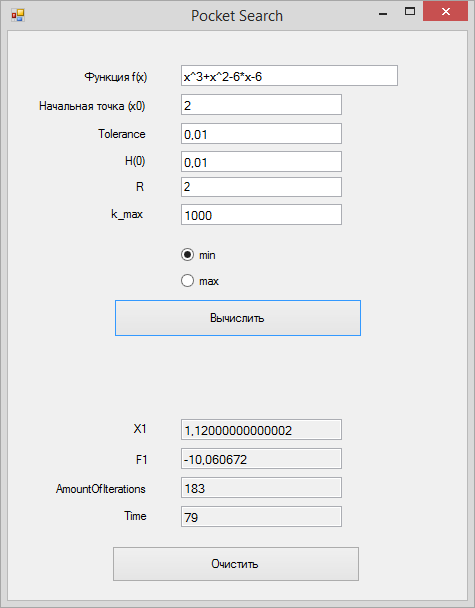


Тест №2:

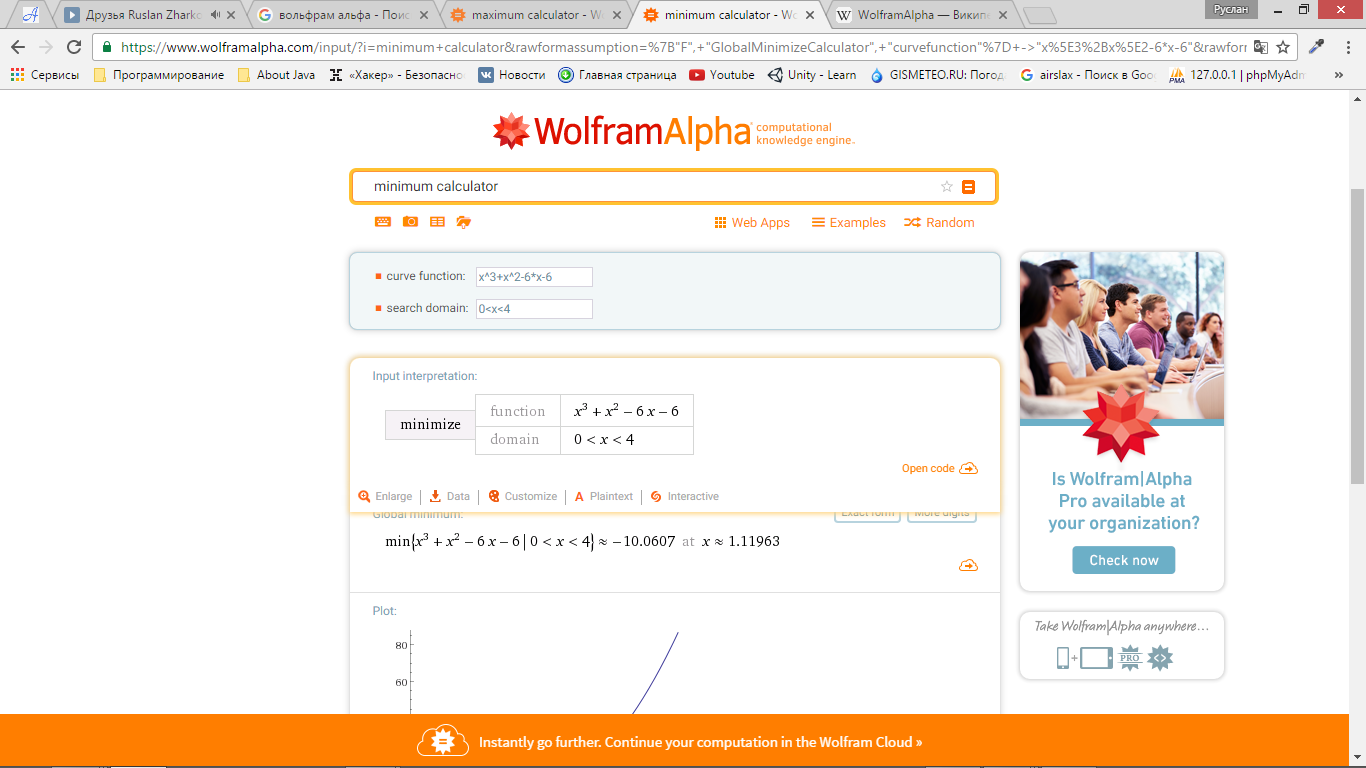


max:

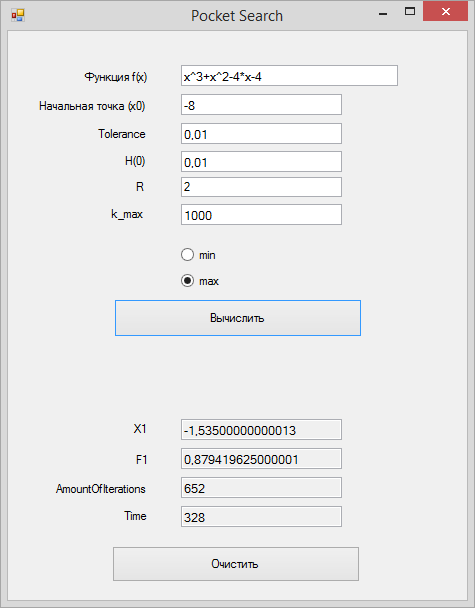




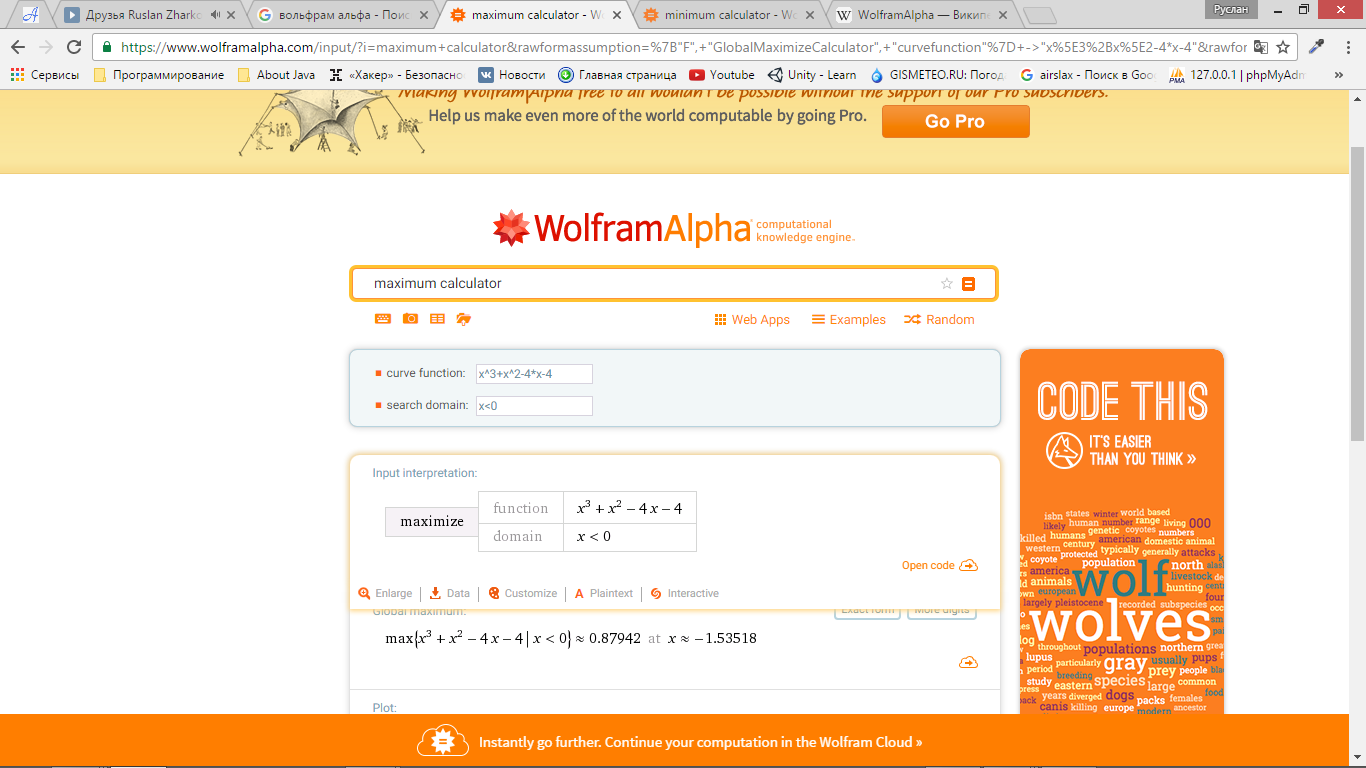
min:

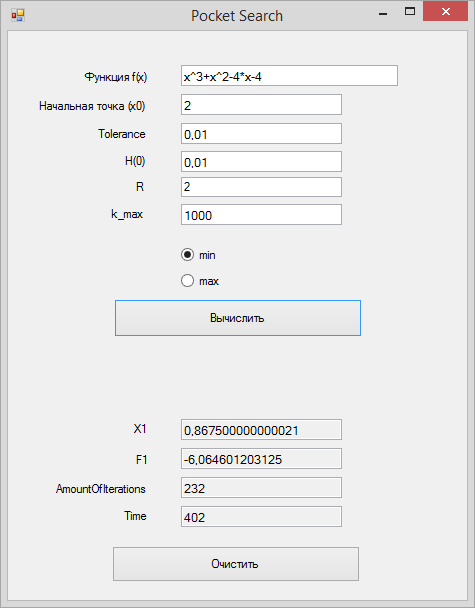


Тест №3:

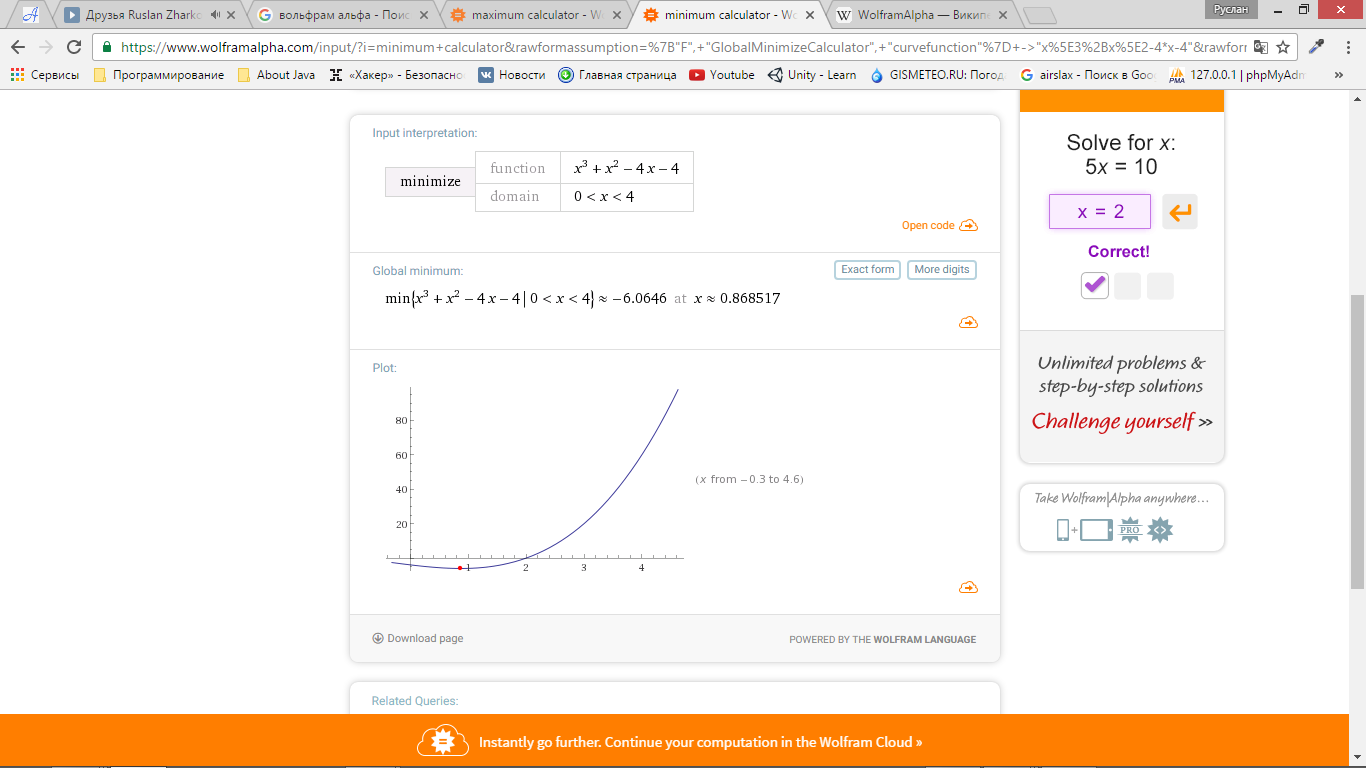


max:





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**

## **Раздел №1**

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

## **Раздел №2**

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.

## **Раздел №3**

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.

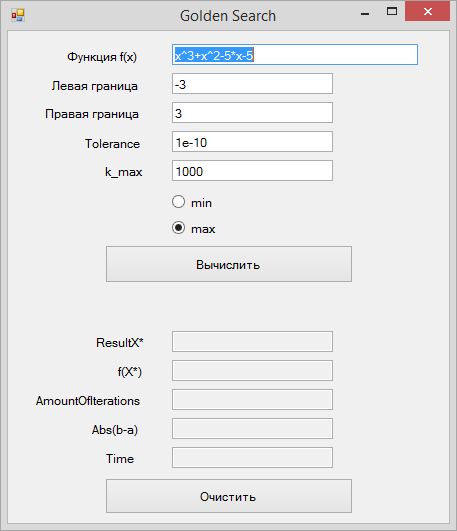
## **Раздел №4**

**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) | FuncBox |
| Textbox1 | Appearance (Text) | x^3+x^2-5\*x-5 |
| TextBox2 | Desing (Name) | LeftEndPointBox |
| TextBox2 | Appearance (Text) | -3 |
| TextBox3 | Design (Name) | RightEndPointBox |
| TextBox3 | Appearance (Text) | 3 |
| TextBox4 | Design (Name) | ToleranceBox |
| TextBox4 | Appearance (Text) | 1e-10 |
| TextBox5 | Design (Name) | kmaxBox |
| TextBox5 | Appearance (Text) | 1000 |
| Label1 | Design (Name) | FuncLabel |
| Label1 | Appearance (Text) | Функция f(x) |
| Label2 | Design (Name) | LeftEndPointLabel |
| Label2 | Appearance (Text) | Левая граница |
| Label3 | Design (Name) | RightEndPointlabel |
| Label3 | Appearance (Text) | Правая граница |
| Label4 | Design (Name) | ToleranceLabel |
| Label4 | Appearance (Text) | Tolerance |
| Label5 | Design (Name) | k\_maxLabel |
| Label5 | Appearance (Text) | k\_max |
| RadioButton1 | Design (Name) | MinradioButton |
| RadioButton1 | Appearance (Text) | min |
| RadioButton1 | Checked | false |
| RadioButton2 | Design (Name) | MaxradioButton |
| RadioButton2 | Appearance (Text) | max |
| RadioButton2 | Checked | true |
| Button1 | Design (Name) | ButtonRunGoldenMethod |
| Button1 | Appearance (Text) | Вычислить |
| Button2 | Design (Name) | ButtonClear |
| Button2 | Appearance (Text) | Очистить |
| ProgressBar1 | Appearance (Text) | ProgressBar1 |
| Label6 | Appearance (Text) | ResultX\* |
| Label6 | Design (Name) | ResultXLabel |
| Label7 | Appearance (Text) | f(X\*) |
| Label7 | Design (Name) | fResultLabel |
| Label8 | Appearance (Text) | AmountIterations |
| Label8 | Design (Name) | AmountIterationsLabel |
| Label9 | Appearance (Text) | Abs(b-a) |
| Label9 | Design (Name) | AbsLabel |
| Label10 | Appearance (Text) | Time |
| Label10 | Design (Name) | TimeLabel |
| ProgressBar1 | Behavior (Visible) | False |
| ProgressBar1 | Design (Name) | ProgressBar1 |
| Textbox6 | Design (Name) | ResultBox |
| Textbox6 | **Behavior (ReadOnly)** | True |
| Textbox7 | Design (Name) | fResultBox |
| Textbox7 | **Behavior (ReadOnly)** | True |
| Textbox8 | Design (Name) | AmountIterationsBox |
| Textbox8 | **Behavior (ReadOnly)** | True |
| TextBox9 | Design (Name) | AbsBox |
| TextBox10 | **Behavior (ReadOnly)** | True |
| Textbox10 | Design (Name) | TimeBox |
| Textbox10 | **Behavior (ReadOnly)** | True |

## **Раздел №5**

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

Стадия №3: Код программы на C#, ассоцированный с интерфейсной формой

“Form1.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;

using System.Diagnostics;

namespace GoldenSearch

{

public partial class GoldenSearch : Form

{

Stopwatch time = new Stopwatch();

string fucntion; // функция

double k\_max, tol, a, b, r, x1, x2, f1 = 0 , f2 = 0, Time, k = 0, abs = 0;

public double Function(double x1)

{

ExpressionParser parser = new ExpressionParser(); // объявляем парсер переменную

parser.Values.Add("x", x1);

// ((DoubleValue)parser.Values["x"]).Value = x1;

return parser.Parse(fucntion); // получаем значение функции из текст. поля

}

public GoldenSearch()

{

InitializeComponent();

}

private void Bisection\_Load(object sender, EventArgs e)

{

}

private void ButtonClear\_Click(object sender, EventArgs e) // очистка полей с решением

{

progressBar1.Value = 0;

ResultBox.Text = "";

fResultBox.Text = "";

AmountIterationsBox.Text = "";

AbsBox.Text = "";

TimeBox.Text = "";

time.Reset();

}

private void ButtonRunGoldenMethod\_Click(object sender, EventArgs e) // решение задачи

{

if (FuncBox.Text == "") MessageBox.Show("Введите функцию");

else if (LeftEndPointBox.Text == "") MessageBox.Show("Введите левую границу");

else if (RightEndPointBox.Text == "") MessageBox.Show("Введите правую границу");

else if (ToleranceBox.Text == "" || Double.Parse(ToleranceBox.Text) <= Double.Parse("1e-16")) MessageBox.Show("Вы не ввели погрешность, или ввели погрешность меньшую 1е-15");

else if (kmaxBox.Text == "") MessageBox.Show("Введите количество итераций");

else

{

time.Start();

ExpressionParser parser = new ExpressionParser(); // объявляем парсер переменную

progressBar1.Value = 0;

progressBar1.Visible = true;

// progressBar1.value += 1;

fucntion = FuncBox.Text;

a = Double.Parse(LeftEndPointBox.Text); // получаем значение левой границы из текст. поля,преобразуем

b = Double.Parse(RightEndPointBox.Text); // получаем значение правой границы из текст. поля, преобразуем

tol = Double.Parse(ToleranceBox.Text); // получаем значение погрешности и текст. поля, преобразуем

k\_max = Double.Parse(kmaxBox.Text);

r = (Math.Sqrt(5) - 1) / 2;

if (a > b) MessageBox.Show("Левая граница не должна быть больше правой");

else if (a == b) MessageBox.Show("Левая граница не должна быть равна правой");

else

{

if (MinradioButton.Checked == true)

{

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

f1 = Function(x1);

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

f2 = Function(x2);

k = 0;

do {

k += 1;

progressBar1.Increment(1);

if (f1 > f2)

{

a = x1;

x1 = x2;

f1 = f2;

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

f2 = Function(x2);

}

else

{

b = x2;

x2 = x1;

f2 = f1;

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

f1 = Function(x1);

}

}

while (k < k\_max && Math.Abs(b - a) > tol );

progressBar1.Value = progressBar1.Maximum;

abs = Math.Abs(b - a);

ResultBox.Text = x1.ToString();

fResultBox.Text = f1.ToString();

AmountIterationsBox.Text = k.ToString();

AbsBox.Text = abs.ToString("0e0");

time.Stop();

TimeSpan tm = time.Elapsed;

TimeBox.Text = String.Format("{0000}", tm.Milliseconds);

MessageBox.Show("Решение найдено!");

progressBar1.Visible = false;

}

if (MaxradioButton.Checked == true)

{

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

f1 = Function(x1);

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

f2 = Function(x2);

k = 0;

do

{

k += 1;

if (f1 < f2)

{

a = x1;

x1 = x2;

f1 = f2;

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

f2 = Function(x2);

}

else

{

b = x2;

x2 = x1;

f2 = f1;

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

f1 = Function(x1);

}

}

while (k < k\_max && Math.Abs(b - a) > tol);

progressBar1.Value = progressBar1.Maximum;

abs = Math.Abs(b - a);

ResultBox.Text = x1.ToString();

fResultBox.Text = f1.ToString();

AmountIterationsBox.Text = k.ToString();

AbsBox.Text = abs.ToString("0e0");

time.Stop();

TimeSpan tm = time.Elapsed;

TimeBox.Text = String.Format("{0000}", tm.Milliseconds);

MessageBox.Show("Решение найдено!");

progressBar1.Visible = false;

}

}

}

}

private void progressBar1\_Click(object sender, EventArgs e)

{

}

}

}

## **Раздел №6**

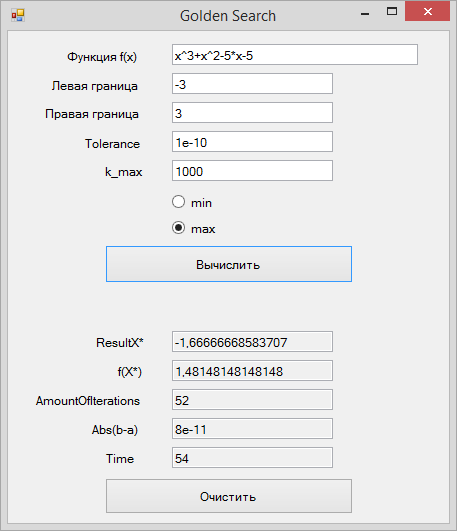
**Тесты для проверки ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, РЕАЛИЗУЮЩЕГО Метод золотого сечения.**

Тесты проводятся на основе вопросно-ответной веб-системы **Wolfram|Alpha**, имеющей математический раздел, где есть возможность находить максимум, или минимум.

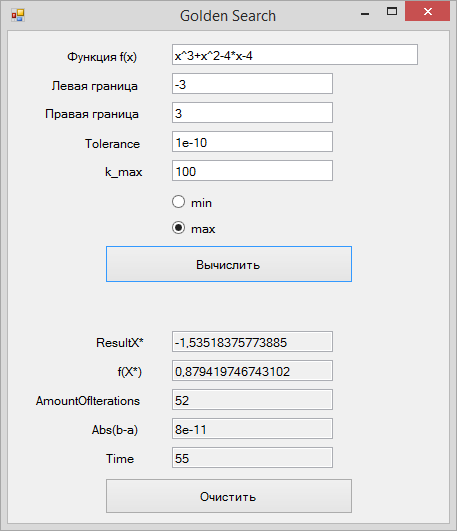
Максимум калькулятор – https://www.wolframalpha.com/input/?i=maximum+calculator

Минимум калькулятор – https://www.wolframalpha.com/input/?i=minimum+calculator

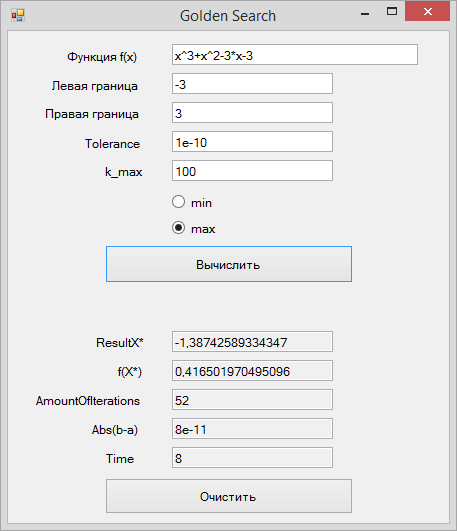
Тест №1.



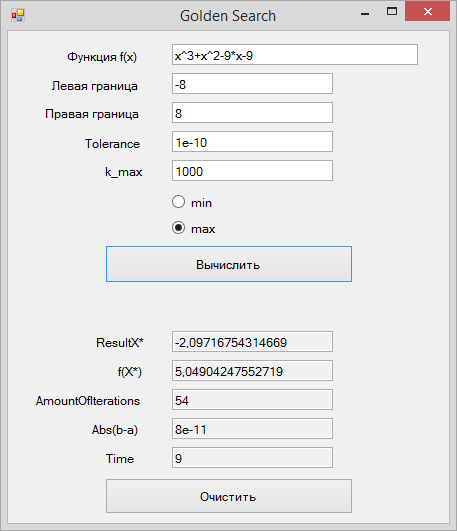
Тест №2.



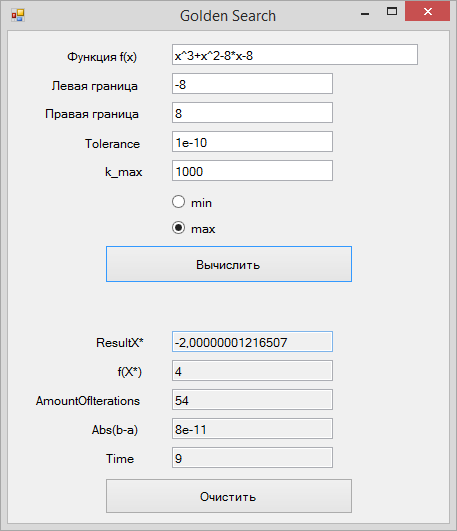
Тест №3



Тест №4



Тест №5



# **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**

## **Раздел №1**

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

## **Раздел №2**

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.

## **Раздел №3**

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.

## **Раздел №4**

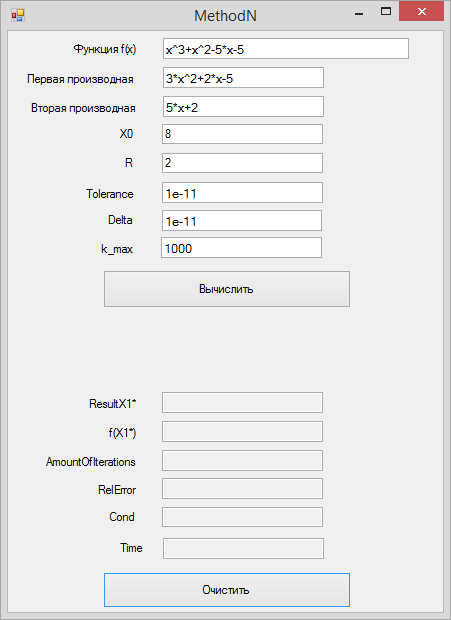
**Стадии *проектирования* системы для поиска минимума нелинейной функции , реализующей метод Ньютона:**

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) | FuncBox |
| TextBox1 | Appearance (Text) | x^3+x^2-5\*x-5 |
| TextBox2 | Desing (Name) | FirstDerivativeBox |
| TextBox2 | Appearance (Text) | 3\*x^2+2\*x-5 |
| TextBox3 | Desing (Name) | SecondDerivativeBox |
| TextBox3 | Appearance (Text) | 5\*x+2 |
| TextBox4 | Design (Name) | x0Box |
| TextBox4 | Appearance (Text) | -8 |
| TextBox5 | Design (Name) | RBox |
| TextBox5 | Appearance (Text) | 2 |
| TextBox6 | Design (Name) | ToleranceBox |
| TextBox6 | Appearance (Text) | 1e-11 |
| TextBox7 | Desing (Name) | DeltaBox |
| TextBox7 | Appearance (Text) | 1e-11 |
| TextBox8 | Design (Name) | kmaxBox |
| TextBox8 | Appearance (Text) | 1000 |
| Label1 | Design (Name) | FuncLabel |
| Label1 | Appearance (Text) | Функция f(x) |
| Label2 | Design (Name) | FirstDerivativeLabel |
| Label2 | Appearance (Text) | Первая производная |
| Label3 | Design (Name) | SecondDerivativeLabel |
| Label3 | Appearance (Text) | Вторая производная |
| Label4 | Design (Name) | x0Label |
| Label4 | Appearance (Text) | X0 |
| Label5 | Design (Name) | RLabel |
| Label5 | Appearance (Text) | R |
| Label6 | Design (Name) | ToleranceLabel |
| Label6 | Appearance (Text) | Tolerance |
| Label7 | Design (Name) | DeltaLabel |
| Label7 | Appearance (Text) | Delta |
| Label8 | Design (Name) | k\_maxLabel |
| Label8 | Appearance (Text) | k\_max |
| Button1 | Design (Name) | ButtonRunNMethod |
| Button1 | Appearance (Text) | Вычислить |
| Button2 | Design (Name) | ButtonClear |
| Button2 | Appearance (Text) | Очистить |
| ProgressBar1 | Appearance (Text) | ProgressBar1 |
| Label8 | Appearance (Text) | ResultX1\* |
| Label8 | Design (Name) | ResultXLabel |
| Label9 | Appearance (Text) | f(X1\*) |
| Label9 | Design (Name) | fResultLabel |
| Label10 | Appearance (Text) | AmountIterations |
| Label10 | Design (Name) | AmountIterationsLabel |
| Label11 | Appearance (Text) | RelError |
| Label11 | Design (Name) | RelLabel |
| Label12 | Appearance (Text) | Cond |
| Label12 | Design (Name) | CondLabel |
| Label13 | Appearance (Text) | Time |
| Label13 | Design (Name) | TimeLabel |
| ProgressBar1 | Behavior (Visible) | False |
| ProgressBar1 | Design (Name) | ProgressBar1 |
| Textbox9 | Design (Name) | ResultBox |
| Textbox9 | **Behavior (ReadOnly)** | True |
| Textbox10 | Design (Name) | fResultBox |
| Textbox10 | **Behavior (ReadOnly)** | True |
| Textbox11 | Design (Name) | AmountIterationsBox |
| Textbox11 | **Behavior (ReadOnly)** | True |
| Textbox12 | Design (Name) | RelErrorBox |
| Textbox12 | **Behavior (ReadOnly)** | True |
| Textbox11 | Design (Name) | CondBox |
| Textbox11 | **Behavior (ReadOnly)** | True |
| Textbox12 | Design (Name) | TimeBox |
| Textbox12 | **Behavior (ReadOnly)** | True |

## **Раздел №5**

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

Стадия №.3: Код программы на C#.

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;

using System.Diagnostics;

namespace MethodN

{

public partial class NMethod : Form

{

Stopwatch time = new Stopwatch();

public double Function(double x0, string func)

{

ExpressionParser parser = new ExpressionParser(); // объявляем парсер переменную

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

// ((DoubleValue)parser.Values["x"]).Value = x1;

return parser.Parse(func); // получаем значение функции из текст. поля

}

public double FirstDerivative(double x0, string deriv) // для первой производной

{

ExpressionParser parser = new ExpressionParser(); // объявляем парсер переменную

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

// ((DoubleValue)parser.Values["x"]).Value = x1;

return parser.Parse(deriv); // получаем значение функции из текст. поля

}

public double SecondDerivative(double x0, string derivv) // для второй производной

{

ExpressionParser parser = new ExpressionParser(); // объявляем парсер переменную

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

// ((DoubleValue)parser.Values["x"]).Value = x1;

return parser.Parse(derivv); // получаем значение функции из текст. поля

}

public NMethod()

{

InitializeComponent();

}

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

{

string function, dfx0, ddfx0; // функция

double k\_max, tol, r, x0, x1, fx0 = 0, fx1 = 0, k = 0, cond = 0, dfx, ddfx, dfx1, Epsilon, QND, DP=0, DP0=0, RelError, Delta, flag=0;

if (FuncBox.Text == "") MessageBox.Show("Введите функцию");

else if (FirstDerivativeBox.Text == "") MessageBox.Show("Введите первую производную");

else if (SecondDerivativeBox.Text == "") MessageBox.Show("Введите вторую производную");

else if (x0Box.Text == "") MessageBox.Show("Введите х0");

else if (RBox.Text == "") MessageBox.Show("Введите параметр R");

else if (ToleranceBox.Text == "" || Double.Parse(ToleranceBox.Text) <= Double.Parse("1e-16")) MessageBox.Show("Вы не ввели погрешность, или ввели погрешность меньшую 1е-15");

else if (kmaxBox.Text == "") MessageBox.Show("Введите количество итераций");

else

{

time.Start();

ExpressionParser parser = new ExpressionParser(); // объявляем парсер переменную

progressBar1.Value = 0;

progressBar1.Visible = true;

// progressBar1.value += 1;

function = FuncBox.Text;

dfx0 = FirstDerivativeBox.Text; // получаем значение левой границы из текст. поля,преобразуем

ddfx0 = SecondDerivativeBox.Text; // получаем значение правой границы из текст. поля, преобразуем

x0 = Double.Parse(x0Box.Text.Trim().Replace(".", ",")); // получаем значение х0

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

r = Double.Parse(RBox.Text); // получаем значение r

tol = Double.Parse(ToleranceBox.Text.Trim().Replace(".", ",")); // получаем значение погрешности и текст. поля, преобразуем

Delta = Double.Parse(DeltaBox.Text.Trim().Replace(".", ","));

k\_max = Double.Parse(kmaxBox.Text);

Epsilon = tol;

fx0 = Function(x0, function);

dfx = FirstDerivative(x0, dfx0);

ddfx = SecondDerivative(x0, ddfx0);

k = 0;

do

{

k += 1;

progressBar1.Increment(1);

if (Math.Abs(ddfx) <= Epsilon)

{

cond = 1;

QND = 0;

}

else

{

DP = dfx / ddfx;

}

if (k == 1)

{

DP0 = DP;

}

if (Math.Sign(DP0) == Math.Sign(DP))

{

x1 = x0 - DP;

}

else

{

x1 = x0 - DP / r;

}

DP0 = DP;

fx0 = Function(x1, function);

dfx1 = FirstDerivative(x1, dfx0);

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

if (RelError < Delta && cond == 1)

{

cond = 2;

}

if (Math.Abs((x1 - x0) / x0) < Epsilon) flag = 1;

x0 = x1;

dfx = dfx1;

}

while ((k < k\_max) && flag != 1) ;

progressBar1.Value = progressBar1.Maximum;

ResultBox.Text = x1.ToString();

fResultBox.Text = fx0.ToString();

AmountIterationsBox.Text = k.ToString();

RelErrorBox.Text = RelError.ToString();

CondBox.Text = cond.ToString();

time.Stop();

TimeSpan tm = time.Elapsed;

TimeBox.Text = String.Format("{0000}", tm.Milliseconds);

MessageBox.Show("Решение найдено!");

progressBar1.Visible = false;

}

}

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

{

progressBar1.Value = 0;

ResultBox.Text = "";

fResultBox.Text = "";

AmountIterationsBox.Text = "";

RelErrorBox.Text = "";

CondBox.Text = "";

TimeBox.Text = "";

time.Reset();

}

}

}

## **Раздел №6**

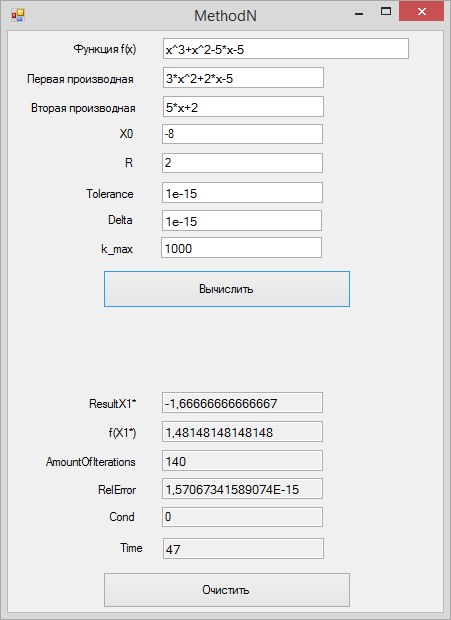
**Тесты для проверки ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, РЕАЛИЗУЮЩЕГО МЕТОД НЬЮТОНА**

Тесты проводятся на основе вопросно-ответной веб-системы **Wolfram|Alpha**, имеющей математический раздел, где есть возможность находить максимум, или минимум.

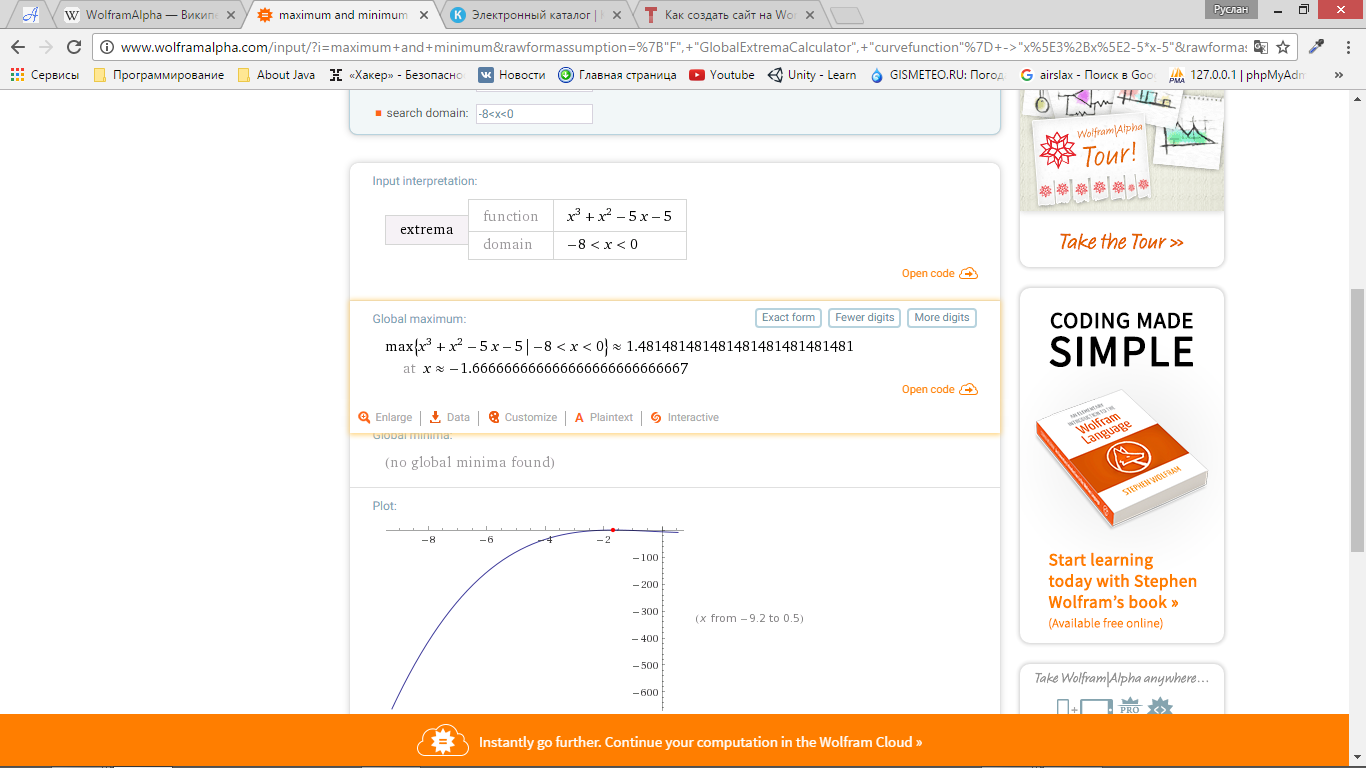
Максимум калькулятор – <https://www.wolframalpha.com/input/?i=maximum+calculator>

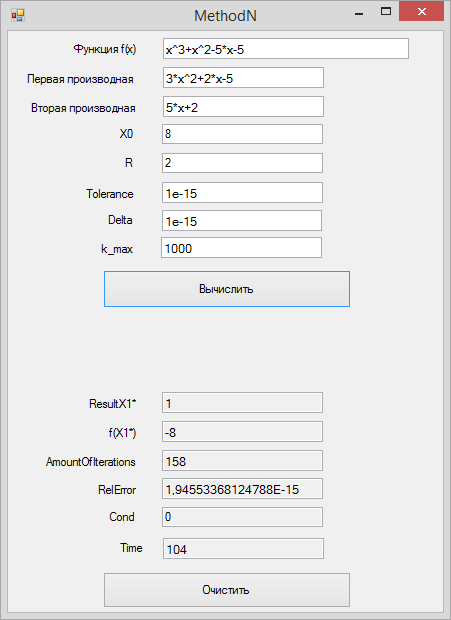
Минимум калькулятор – <https://www.wolframalpha.com/input/?i=minimum+calculator>

Тест №1

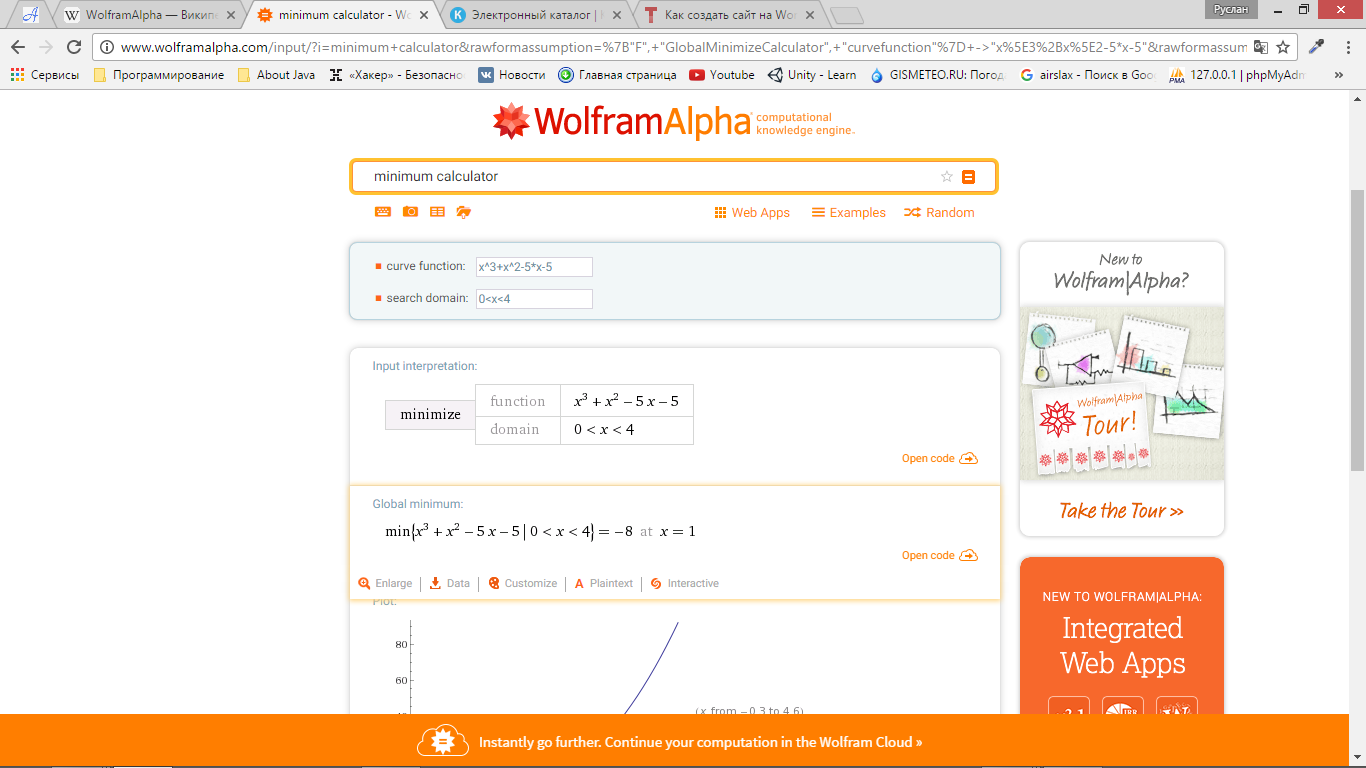


max:

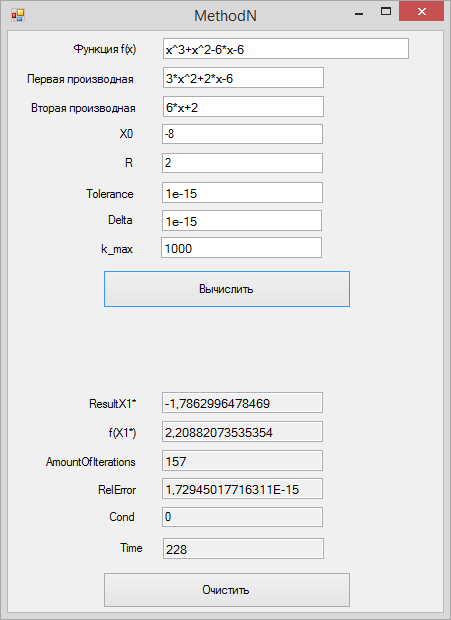




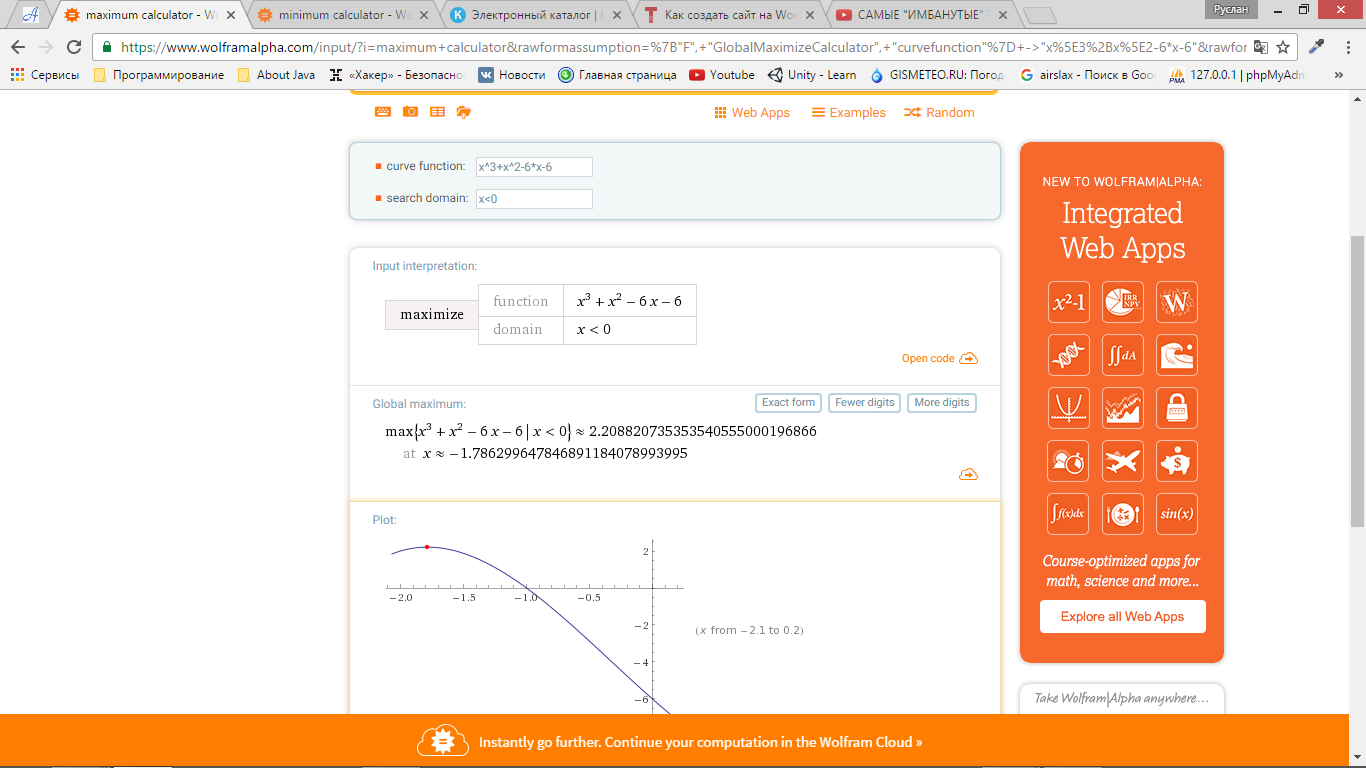
min:

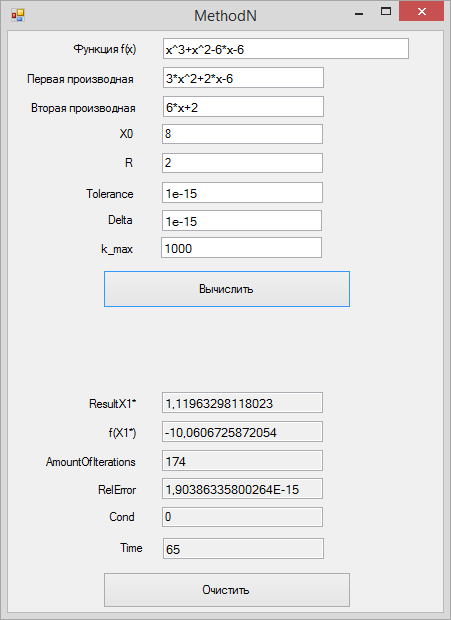


Тест №2

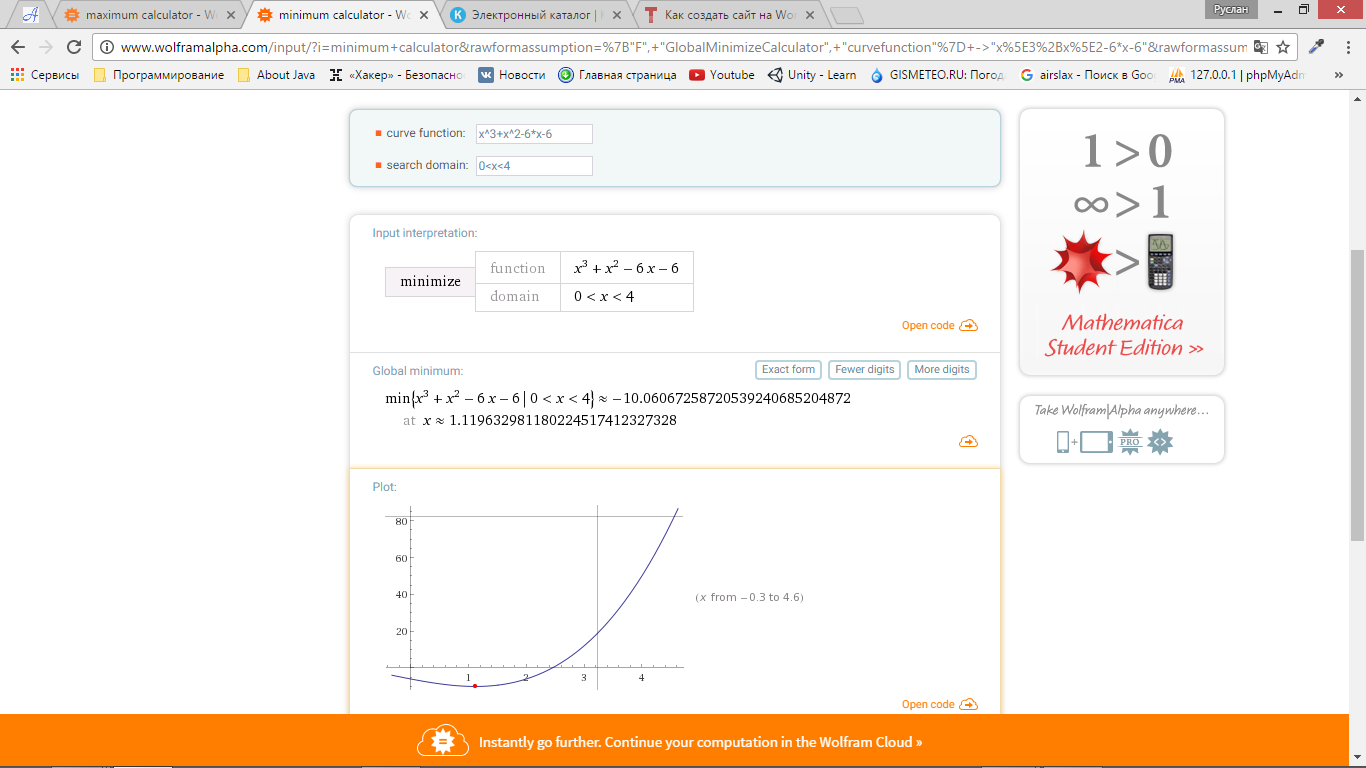


max:

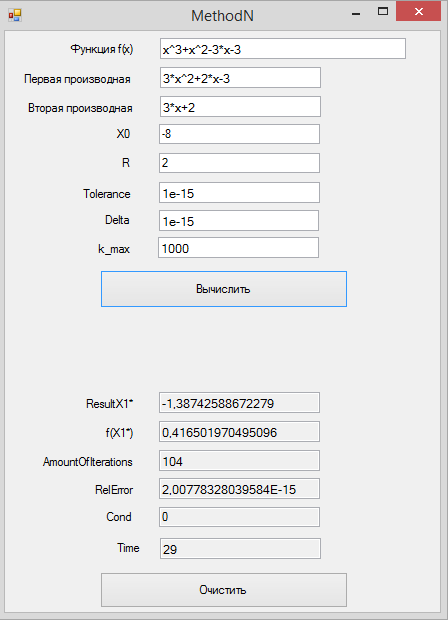




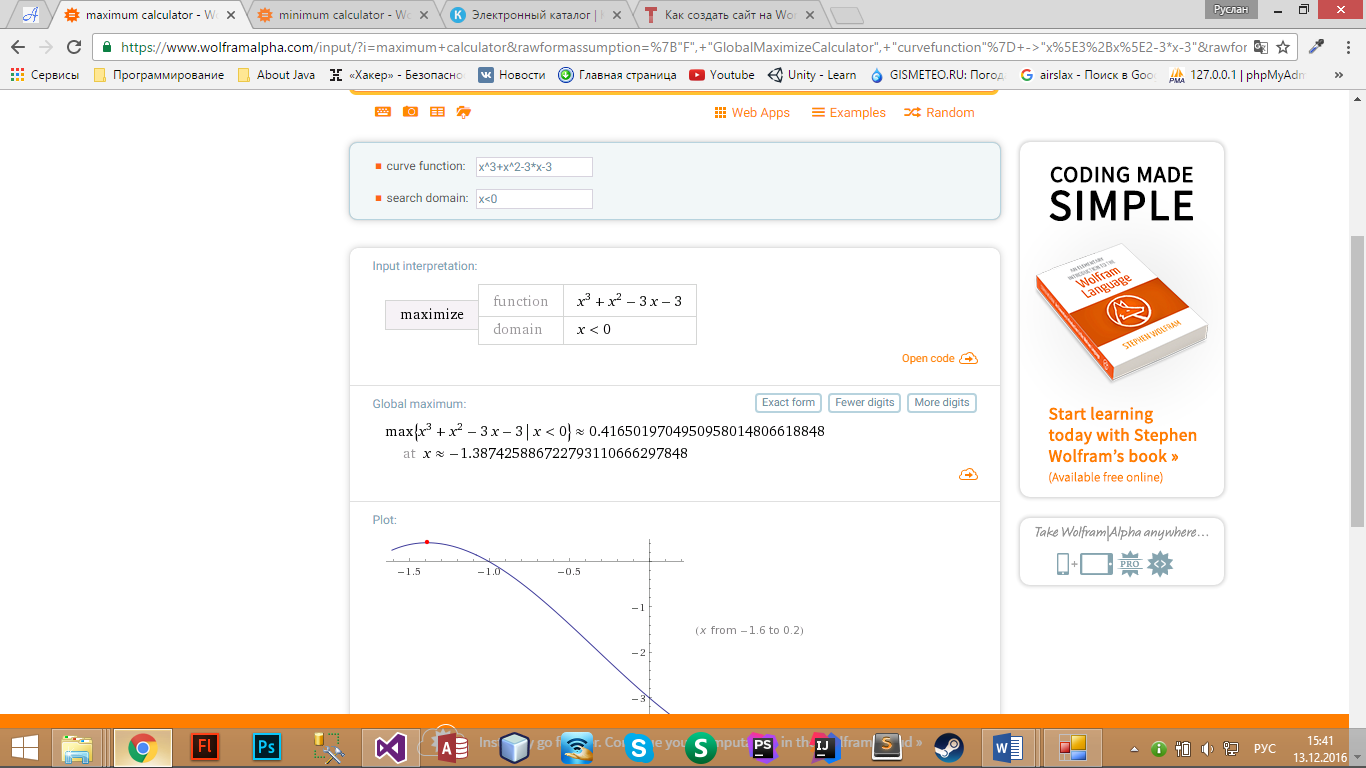
min:

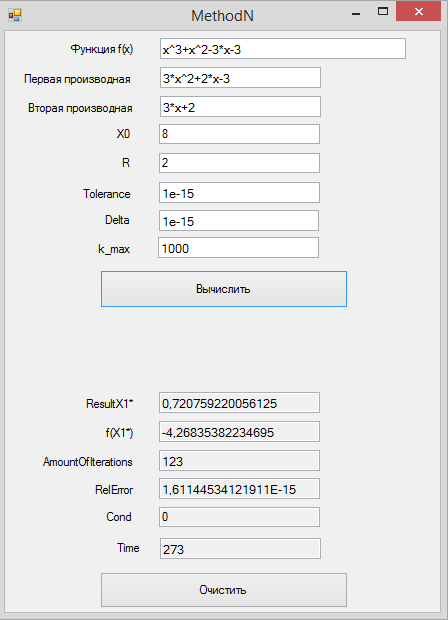


Тест №3



max:





min:

