Ministry of Education and Science of Ukraine
Kharkiv National University of Radioelectronics

Department of System Techniques

Subject:
«Optimization method and operations»

Practice work №1
“Single-Variable Optimization”

Done by
Avdan O.,
Buriak D.,
Kuzmin A.
Filonova N.,
Iatsenko A.
Group KHT-19-1

Taken by
Vyshniak M.Y.
with the mark «_____»

Kharkiv
2020

1.1 Our team:

Our team consists of 5 participants, among which: Olexandra Avdan, Buriak Daniil, Kuzmin Artem, Filonova Nadiia and Iatsenko Anna, all from the group KHT-19-1.

1.2 Purpose of work:

To explore the realization of methods of optimization single-variable functions.

1.3 Theme for the following practice:

Theme №2 - Golden Section Search

2. Golden Section Search

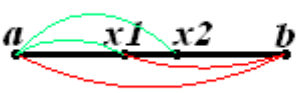
(a)Minimize $f(x) = (10 - x)^2$ over the interval $6 \le x \le 15$, $\varepsilon = 0.01$

(б) Minimize $f(x) = 3x^4 + (x - 1)^2$ over the range [0, 4].

1.4 Algorithm description:

The golden section search algorithm can be used for finding a minimum (or maximum) of a single-variable function *f(x)*. Important is that, *f(x)* should be strictly unimodal function and extremum (min or max) is known to exist.

Assume, the given function is $f(x): [a, b] \to \mathbb{R}, f(x) \in C([a, b])$. . To find an unknown value of *f(x)* on the certain section (maximum or minimum) this section has to be divided according to the proportion of the golden section search into two directions, i.e. two points *x1* and *x2* are chosen. That points are called through-points and could be illustrated like this:



x1 and *x2* must meet the demands below:

$$\frac{b-a}{b-x_1} = \frac{b-a}{x_2-a} = \Phi = \frac{1+\sqrt{5}}{2} = 1.618\ldots \quad (1)$$

Where $\boldsymbol{\Phi}$ – golden section search proportion.

From (1) we can make a conclusion :

$$\begin{aligned} x_1 &= b - \frac{(b-a)}{\Phi} \\ x_2 &= a + \frac{(b-a)}{\Phi} \end{aligned} \quad (2)$$

Point *x1* divides the section *[a, x2]* and point *x2* divides the section *[x1, b]* according to the golden section search proportion.

An *algorithm* of finding minimum/maximum according to the golden search method described below:

Step 1. On the first iteration the section divides by two symmetric about center points.

Step 2. The aim is to decide, which end of segment is closer to the point with maximum value. This end should be thrown back (for calculating minimum).

Step 3. The next iteration we should find only one new point (because we have another point from the previous iteration).

Step 4. Repeating *Step 3* until the needed accuracy would be reached.

Formally this algorithm can also be shown as below:

Step 1. Presetting starting segment borders *a, b* and accuracy *ε.*

Step 2. Calculating starting dividing points :

$x_1 = b - \frac{(b-a)}{\Phi}, \quad x_2 = a + \frac{(b-a)}{\Phi}$ and function value at these points:

$y_1 = f(x_1),\ y_2 = f(x_2)$

If $y_1 \geq y_2$ (in case of calculation minimum) *a = x1*, else *b = x2*.

Step 3. If the needed accuracy ε is reached, $|b - a| < \varepsilon$, then $x = \frac{a+b}{2}$ and further calculations should be stopped. Else repeat *Step 2*.

1.5 Program listing

Program, that realizes Golden Section Search algorithm is written on *Java*, because to our concern this programming language is an optimal combination of convenience and usability. All the code contains comments concerning certain functions and whole classes to make a code more understandable.

1.5.1 Interface Calculator

```
package ua.knt_19_1.drink_for_love.intefaces;

import javax.script.ScriptException;

public interface Calculator {//The interface deals with the processing of the basic formulas of the method.
    boolean checkCondition(Storage storage);//returns true if conditions(method`s) is true

    boolean calculateNewIteration(Storage storage, Writer writer) throws ScriptException;//returns false if we in cycle and calculates new iteration of method
}
```

1.5.2 Interface Storage

```
package ua.knt_19_1.drink_for_love.intefaces;

public interface Storage {//The interface stores all variables needed for method

    void setA(double a);//Setter for this variable
    void setB(double b);//Setter for this variable
    void setE(double e);//Setter for this variable
    void setFunction(String function);//Setter for this variable(set only outer function)
    void setX(double x);//Setter for this variable

    double getA();//Getter for this variable
    double getB();//Getter for this variable
    double getE();//Getter for this variable
    double getX();//Getter for this variable
    String getFunction();//Getter for this variable(return only inner function like 3*x*x*x)

    void takeValues();//read values from console
```

```
    void putValues();//write values to console

}
```

1.5.3 Interface Writer

```
package ua.knt_19_1.drink_for_love.intefaces;

import java.util.List;

public interface Writer {//The interface for writing log to file

    String getFilename();//Getter for this variable
    void setFilename(String filename);//Setter for this variable

    List<String> getOuts();//Getter for this variable
    void setOuts(List<String> outs);//Setter for this variable

    void addString(String iter);//Add String for outputting to file

    boolean flush();//Write first 10 Strings and the last one to file

}
```

1.5.4 Class CalculatorClass

```
package ua.knt_19_1.drink_for_love.classes;

import ua.knt_19_1.drink_for_love.intefaces.Calculator;
import ua.knt_19_1.drink_for_love.intefaces.Storage;
import ua.knt_19_1.drink_for_love.intefaces.Writer;

import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;
import javax.script.ScriptException;

public class CalculatorClass implements Calculator {//The class deals with
the processing of the basic formulas of the method

    static int counter = 0;//counter of cycles
  //static double fi = (1+Math.sqrt(5))/2;//constant Fi for calculations
    static double fi = (Math.sqrt(5)-1)/2;//constant Fi for calculations
    static double x1;
    static double x2;
    static double y1;
    static double y2;

    @Override
    public boolean checkCondition(Storage storage) {
        return Math.abs(storage.getB() - storage.getA()) >
storage.getE();//Checking condition |b-a|>e
    }//Checks method condition

    @Override
   /* public boolean calculateNewIteration(Storage storage, Writer writer)
throws ScriptException {
        if(++counter>30){//Checking loop condition
```

```
            System.out.println("So many iterations!");//Informing the user
about looping
            return false;//Returning false in this condition
        }
        double x1 =storage.getB()-((storage.getB()-
storage.getA())/fi);//Calculating intermediate value
        double x2 =storage.getA()+((storage.getB()-
storage.getA())/fi);//Calculating intermediate value
        double y1 = foo(storage,x1);//Calculating intermediate value
        double y2 = foo(storage,x2);//Calculating intermediate value
        writer.addString(counter + ") a = "+storage.getA()+"; b =
"+storage.getB()+"; x1 = "+x1+"; x2 = "+x2+"; y1 = "+y1+"; y2 =
"+y2+";\n");//Writing new String to file
        if(y1>=y2){//Checking method inner condition f(x1)>=f(x2)
            storage.setA(x1);//Setting new left value of range
        }else {//In other case
            storage.setB(x2);//Setting new right value of range
        }
        return true;//Returning true
    }//Calculates new iteration of method (return false if we more then on 30
iteration else returns true)

    */

    public boolean calculateNewIteration(Storage storage,Writer writer)
throws ScriptException {
        if(++counter>30){//Checking loop condition
            System.out.println("So many iterations!");//Informing the user
about looping
            return false;//Returning false in this condition
        }
        if( counter == 1 ){//Initializing variables
            x1 = storage.getA() + (( storage.getB() - storage.getA() ) * ( 1-
fi ));//Calculating x1 on first iteration a+(b-a)*(1-fi)
            x2 = storage.getA() + (( storage.getB() - storage.getA() ) * fi
);//Calculating x2 on first iteration a+(b-a)*fi
            y1 = foo(storage, x1);//Calculating f(x1) on first iteration
            y2 = foo(storage, x2);//Calculating f(x2) on first iteration
        }
        writer.addString( counter + ") a = " + storage.getA() + "; b = " +
storage.getB() + "; x1 = " +
                x1 + "; x2 = " + x2 + "; f(x1) = " + y1 + "; f(x2) = " + y2 +
";\n");//Adding string to file
        if( y1 > y2){//If f(x1) > f(x2)
            storage.setA(x1);//Setting new a
            x1 = x2;//Setting x1 as prev x2
            y1 = y2;//Setting f(x1) as prev f(x2)
            x2 = storage.getA() + (( storage.getB() - storage.getA() ) * fi
);//Calculating new x2 a+(b-a)*fi
            y2 = foo(storage, x2);//Calculating f(x2)
        }
        else {//If f(x2) > f(x1) or equals
            storage.setB(x2);//Setting new b
            x2 = x1;//Setting x2 as prev x1
            y2 = y1;//Setting f(x2) as prev f(x1)
            x1 = storage.getA() + (( storage.getB() - storage.getA() ) * (1 -
fi));//Calculating new x1 a+(b-a)*(1-fi)
            y1 = foo(storage, x1);//Calculating f(x1)
        }
        return true;//Rerurning true
    }//Calculates new iteration of method (return false if we more then on 30
iteration else returns true)
```

```
    private double foo(Storage storage, double x) throws ScriptException
{//Function returns calculated value of function on this x
        String function =
storage.getFunction().replace("x",String.valueOf(x));//Replacing 'x' in
function to specific value
        ScriptEngineManager manager = new ScriptEngineManager();//Creating
manager for ScriptEngines
        ScriptEngine scriptEngine =
manager.getEngineByName("JavaScript");//Creating JavaScript engine object
        return
Double.parseDouble(scriptEngine.eval(function).toString());//Calculating
value of function on this x and parsing it to Double
    }

}
```

1.5.5 Class StorageClass

```
package ua.knt_19_1.drink_for_love.classes;

import ua.knt_19_1.drink_for_love.intefaces.Storage;

import java.util.Scanner;

public class StorageClass implements Storage {//The class stores all
variables needed for method

     static double a = 0;//left value of range a<=x<=b
     static double b = 0;//right value of range a<=x<=b

     static double e = 0;//accuracy of our calculations
     static String[] function={"",""};//outer(from the user) and inner(for
calculations) function for method
     static double x =0;//minimum extremum point of the function


    @Override
    public void setA(double a) {
    StorageClass.a =a;
    }//Setter for this variable

    @Override
    public void setB(double b) {
    StorageClass.b =b;
    }//Setter for this variable

    @Override
    public void setE(double e) {
    StorageClass.e =e;
    }//Setter for this variable

    @Override
    public double getA() {
        return a;
    }//Getter for this variable

    public  String getFunction() {
        return function[1];
    }//Getter for this variable(return only inner function like 3*x*x*x)
```

```
    public  void setFunction(String function) {
        StorageClass.function[0] = function;
    }//Setter for this variable(set only outer function)

    @Override
    public double getB() {
        return b;
    }//Getter for this variable

    @Override
    public double getE() {
        return e;
    }//Getter for this variable

    @Override
    public void takeValues() {
        Scanner in = new Scanner(System.in);//Creating new Scanner object for
reading from incoming stream System.in which linked with console input
        System.out.print("Input a function: \n");//Outputting String to
console using System.out - output stream witch linked to console output
        function[0] = in.nextLine();//Reading new line from console
        function[1] = changeFunction(function[0]);//Rewrite function from
human-friendly form to JavaScript-friendly form
        System.out.print("Input a arguments for method: a, b and accuracy:
\n");////Outputting String to console
        a = in.nextDouble();//Reading new double from console
        b = in.nextDouble();//Reading new double from console
        e = Double.parseDouble( in.nextLine().replace(",",".") );//Reading new
double from console using "replace" method to avoid differences between '.'
and ','
        System.out.print("Thank you!\n");//Thanks for using our program ;)
    }//Read values for method from console

    @Override
    public  double getX() {
        return x;
    }//Returns X value

    @Override
    public  void setX(double x) {
        StorageClass.x = x;
    }//Setter for X

    @Override
    public void putValues() {
        setX((getA()+getB())/2);//Calculating x before outputting
        System.out.printf("Out x for function %s is %.2f
%n",function[0],x);//Writing function and x to console

    }//Writes x for this function to console

    private String changeFunction(String function){
        int index;//index of first entry of pattern
        while (true){//While endless loop with break option inside
            index = function.indexOf("x^");//Finding "x^" pattern in string
            if(index==-1){//If no such pattern
                break;//Breaking the loop
            }else {//In another case
                char typeOperation = function.charAt(index-1);//Finding type
operation(operation symbol such as '+', '-', '*' and '/' or digit number)
                int count =
Integer.parseInt(String.valueOf(function.charAt(index+2)));//Finding amount
for pow
```

```
                if(typeOperation=='+'||typeOperation=='-
'||typeOperation=='*'||typeOperation=='/'){//If this is operation before
pattern
                    function = function.replace("x^"+count, "x" +
"*x".repeat(Math.max(0, count - 1)));//Rewriting function String by replacing
x^4 to x*x*x*x for example
                }else if(Character.isDigit(typeOperation)){//If this is digit
before pattern
                    function = function.replace("x^"+count,
"*x".repeat(Math.max(0, count)));//Rewriting function String by replacing
3x^4 to 3*x*x*x*x for example
                }else{//In other case
                    System.out.println( "Something went wrong!\n");//Writing
message to user
                }
            }
        }
        while (true){////While endless loop with break option inside
            index = function.indexOf(")^");//Finding "(...)^" pattern in
string
            if(index==-1){//If no such pattern
                break;//Breaking the loop
            }else {//In another case
                StringBuilder str=new StringBuilder(")");//Creating object
for finding brackets pattern
                int count=1;//Counter for finding opened and closed brackets
                for(int i = index-1;i>=0;i--){//Reverse "for" loop for
finding brackets pattern in String from ")^"
                    if(function.charAt(i)==')'){//If this character is closed
brackets
                        count++;//We increasing counter of brackets
                    }else if(function.charAt(i)=='('){//If this character is
opened brackets
                        count--;//We decreasing counter of brackets
                    }
                    str.append(function.charAt(i));//Appending StringBuilder
object. After loop we will have String like ")x-01("
                    if(count==0){//If we found all brackets hierarchy
                        break;//Breaking the loop ahead of time
                    }
                }
                str = str.reverse();//Reversing our brackets String from ")x-
01(" to "(10-x)"
                int count2 =
Integer.parseInt(String.valueOf(function.charAt(index+2)));//Finding amount
for pow
                function = function.replace(str.toString()+"^"+count2, str +
("*" + str.toString()).repeat(count2 - 1));//Rewriting function String by
replacing (10-x)^2 to (10-x)*(10-x) for example
            }
        }


        return function;//Returning new String
    }//Rewrites function from human-friendly form to JavaScript-friendly form
}
```

1.5.6 Class WriterClass

```
package ua.knt_19_1.drink_for_love.classes;

import ua.knt_19_1.drink_for_love.intefaces.Writer;

import java.io.File;
import java.io.FileWriter;
```

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class WriterClass implements Writer {//The class for writing log to
file

    String filename;     //name of file for output
    List<String> outs;   //list of strings to output
    String lastOne = "";      //last added string after 10 already added
strings
    int count = 0;              //counter for added strings

    public WriterClass(String s) {//Constructor with parameter String as
filename
        filename = s;
        outs=new ArrayList<>();//Creating new List object
    }

    @Override
    public String getFilename() {
        return filename;
    }//Getter for this value

    @Override
    public void setFilename(String filename) {
        this.filename = filename;
    }//Setter for this value

    @Override
    public List<String> getOuts() {
        return outs;
    }//Getter for this value

    @Override
    public void setOuts(List<String> outs) {
        this.outs=outs; //Add all outs
    }//Setter for this value

    @Override
    public void addString(String iter) {//Add String for outputting to file
        if(count<10){//Add String for outputting to file
            outs.add(iter);//Add String to list using List`s method "add"
            count++;//Increasing the counter
        }else {//If we already have 10 Strings
            lastOne = iter;//Rewrite lastOne with this String
        }

    }

    @Override
    public boolean flush() {//Write first 10 Strings and the last one to file
        File file = new File(filename);//Creating new File`s object for
writing Strings to it

        try (//Start of "try" block
                FileWriter fileWriter = new FileWriter(file)//Using "()"
after "try" to autoclose FileWriter and creating FileWriter object to control
outputting stream to file
        ) {
            for (String s: outs) {//In cycle "for each" we address to each of
10 Strings
                fileWriter.write(s);//Writing each of 10 String to file
```

```
            }
            fileWriter.write(lastOne);//Writing the last one before flushing
String
            lastOne="";//Reset value of this variable
            outs.clear();//Reset list for strings
        } catch (IOException e) {//End of "try" block and start of "catch"
clause
            e.printStackTrace();//Printing all error branch
            return false;//Returns false when we catches error
        }

        return true;//Returns true if we don't have errors
    }
}
```

1.5.7 Class GoldenSearchMethod

```
package ua.knt_19_1.drink_for_love;

import ua.knt_19_1.drink_for_love.classes.CalculatorClass;
import ua.knt_19_1.drink_for_love.classes.StorageClass;
import ua.knt_19_1.drink_for_love.classes.WriterClass;
import ua.knt_19_1.drink_for_love.intefaces.Calculator;
import ua.knt_19_1.drink_for_love.intefaces.Storage;
import ua.knt_19_1.drink_for_love.intefaces.Writer;

import javax.script.ScriptException;

public class GoldenSearchMethod {//The main class for our program

    private static Writer writer;//Variable for Writer interface
    private static Calculator calculator;//Variable for Calculator interface
    private static Storage storage;//Variable for Storage interface

    public static void main(String[] args) throws ScriptException {//The main
method for our method

        writer = new WriterClass("log.txt");//Creating new WriterClass object
with file name
        storage = new StorageClass();//Creating new StorageClass object
        calculator = new CalculatorClass();//Creating new CalculatorClass
object

        storage.takeValues();//Read values from console and put it to
variables in StorageClass object

        do{}//Using empty cycle "do while" because we calculate new iteration
in "while" part
        while(calculator.calculateNewIteration(storage, writer) &&
calculator.checkCondition(storage));//Checking looping in iterations and
method conditions

        storage.putValues();//Write values from StorageClass object to
console

       writer.flush();//Write all log Strings to file
    }
}
```

1.6 Results of the program's work:

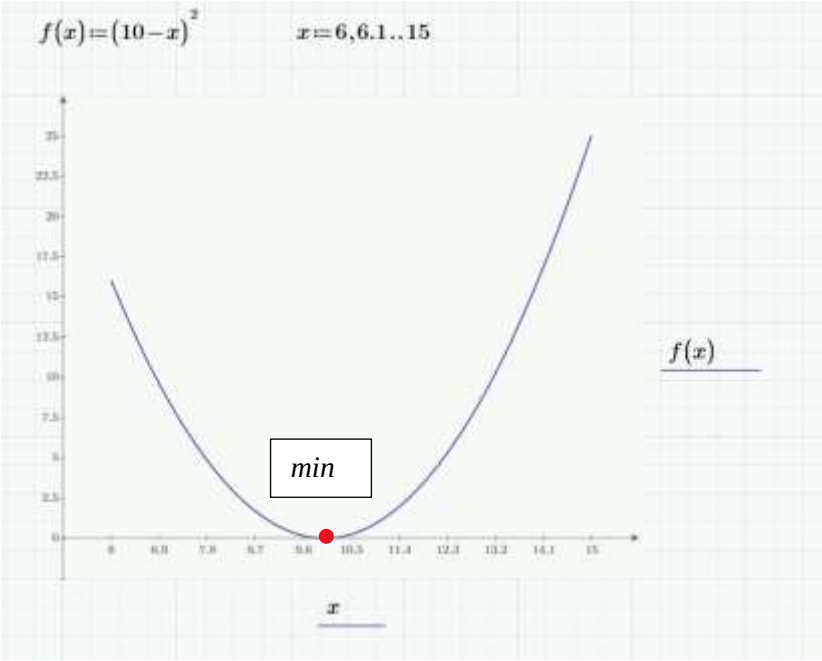
Solution of an example ***a)***

```
Input a function:
(10-x)^2
Input a arguments for method: a, b and accuracy:
6 15 0.01
Thank you!
Out x for function (10-x)^2 is 10,00

Process finished with exit code 0
```

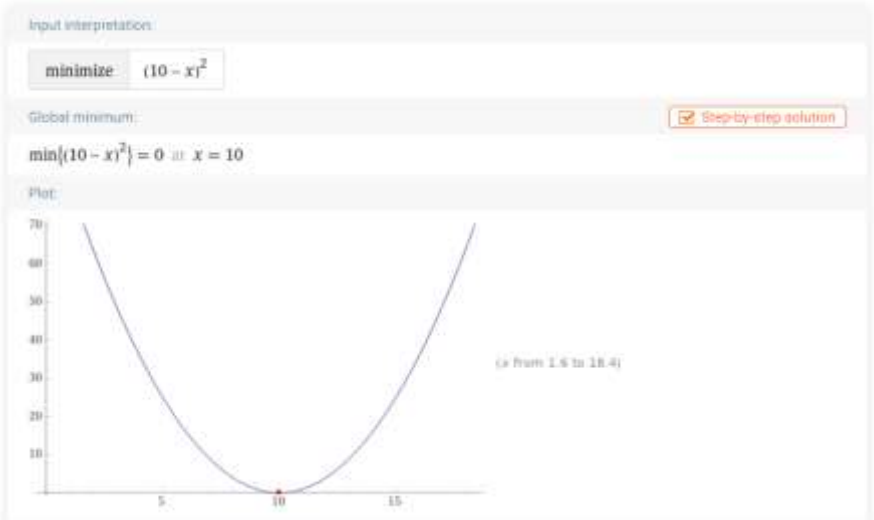
On the 15th iteration a minimum was found.

```
Out x for function (10-x)^2 and accuracy 1,0000000 is 9,8434588 Amount of iterations is 5
Out x for function (10-x)^2 and accuracy 0,1000000 is 10,0210597 Amount of iterations is 10
Out x for function (10-x)^2 and accuracy 0,0100000 is 9,9984472 Amount of iterations is 15
Out x for function (10-x)^2 and accuracy 0,0010000 is 9,9997073 Amount of iterations is 19
Out x for function (10-x)^2 and accuracy 0,0001000 is 10,0000048 Amount of iterations is 24
Out x for function (10-x)^2 and accuracy 0,0000100 is 9,9999985 Amount of iterations is 29
Out x for function (10-x)^2 and accuracy 0,0000010 is 10,0000002 Amount of iterations is 34
Out x for function (10-x)^2 and accuracy 0,0000001 is 10,0000000 Amount of iterations is 39
```

To visualize the result, a graph in Mathcad was built:



Also to decide the answer right we used Wolphram Alpha



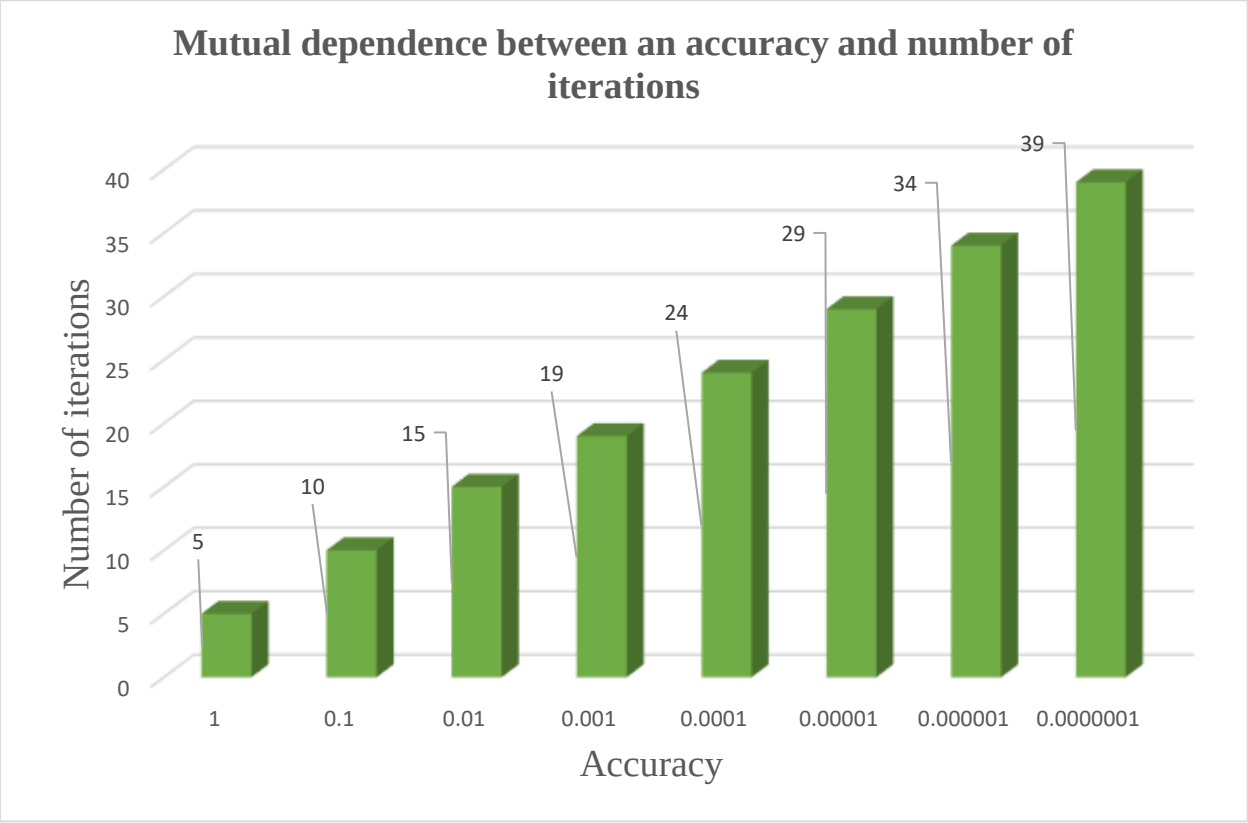
Here we can see the result we've already got during the program's work. So, our program works correctly.

Let's figure out the dependence between an accuracy and number of iterations.

Accuracy	Number of iterations
1	5
0.1	10

0.01	15
0.001	19
0.0001	24
0.00001	29
0.000001	34
0.0000001	39

Having these values let's build a graph for better visualization:



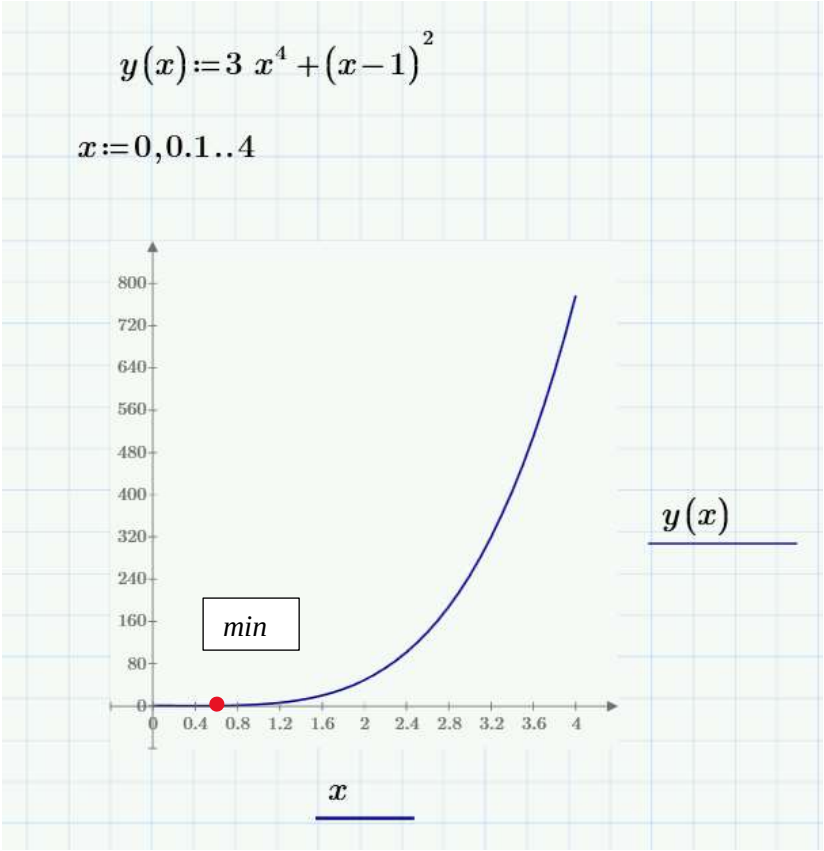
From this graph we can make a conclusion, that the more accurate result need to be, the more iterations have to be done.

Solution of an example ***b)***

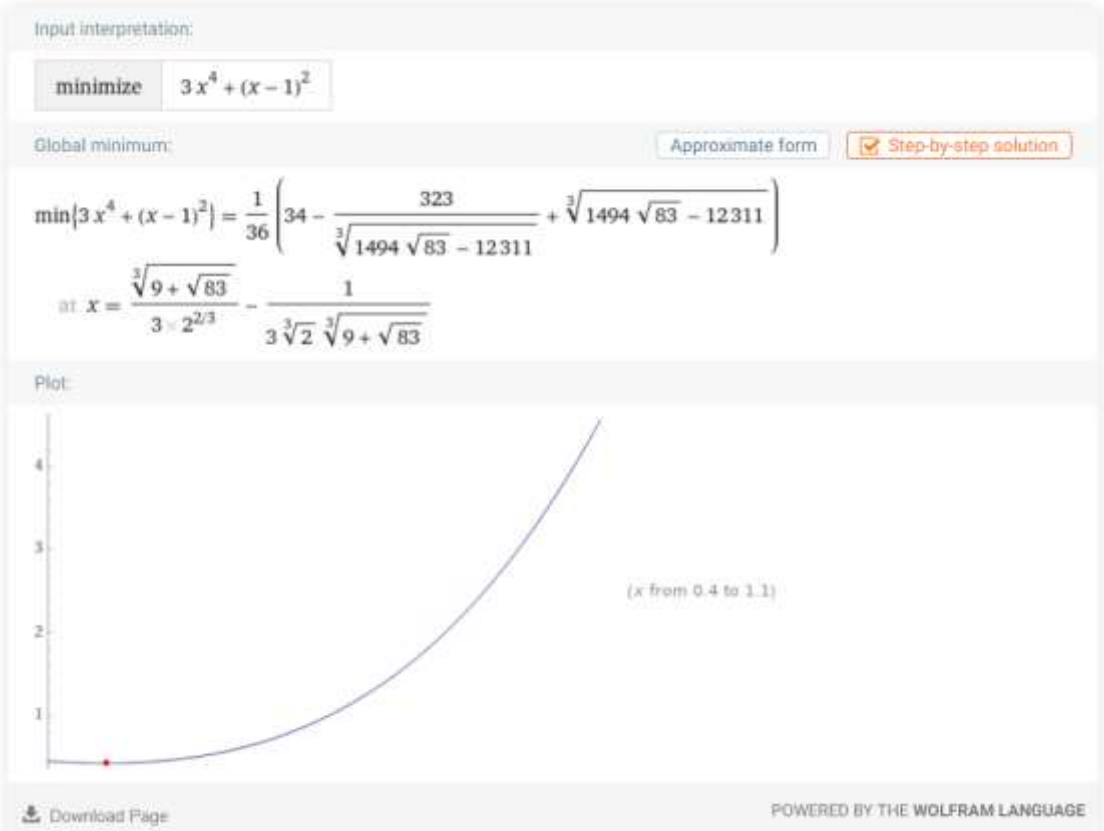
```
Input a function:
3x^4+(x-1)^2
Input a arguments for method: a, b and accuracy:
0 4 0.01
Thank you!
Out x for function 3x^4+(x-1)^2 is 0,45

Process finished with exit code 0
```

```
Out x for function 3x^4+(x-1)^2 and accuracy 1,0000000 is 0,4721360 Amount of iterations is 3
Out x for function 3x^4+(x-1)^2 and accuracy 0,1000000 is 0,4558747 Amount of iterations is 8
Out x for function 3x^4+(x-1)^2 and accuracy 0,0100000 is 0,4496635 Amount of iterations is 13
Out x for function 3x^4+(x-1)^2 and accuracy 0,0010000 is 0,4506514 Amount of iterations is 18
Out x for function 3x^4+(x-1)^2 and accuracy 0,0001000 is 0,4507019 Amount of iterations is 23
Out x for function 3x^4+(x-1)^2 and accuracy 0,0000100 is 0,4506991 Amount of iterations is 27
Out x for function 3x^4+(x-1)^2 and accuracy 0,0000010 is 0,4506989 Amount of iterations is 32
Out x for function 3x^4+(x-1)^2 and accuracy 0,0000001 is 0,4506988 Amount of iterations is 37
```

To visualize the result, a graph in Mathcad was built:



Also to decide the answer right we used Wolphram Alpha:

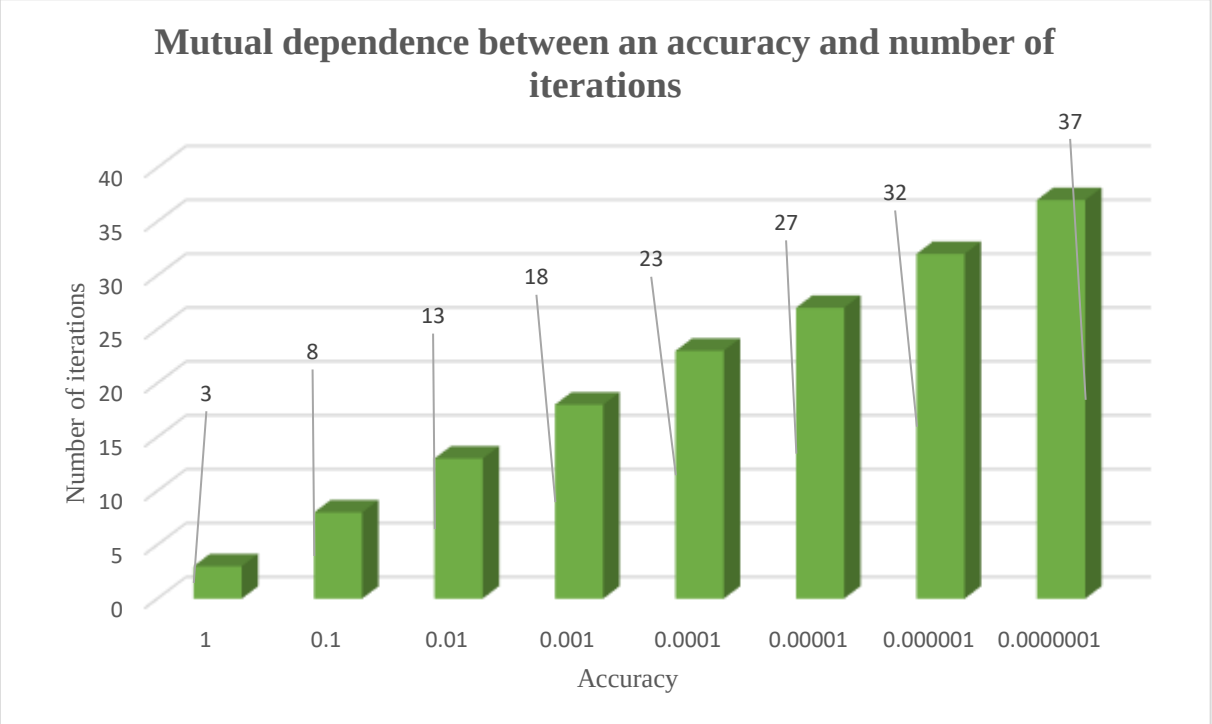


Here we can see the result we've already got during the program's work. So, our program works correctly.

Let's figure out the dependence between an accuracy and number of iterations.

Accuracy	Number of iterations
1	3
0.1	8
0.01	13
0.001	18
0.0001	23
0.00001	27
0.000001	32
0.0000001	37

Having these values let's build a graph for better visualization:



From this graph we can make a conclusion, that the more accurate result need to be, the more iterations have to be done.

1.7 Conclusion

During this practice work the realization of methods of optimization single-variable functions was explored. The program, which calculates the minimum of a single – variable function was written. All the results of a program were checked and decided the right ones. Also, to visualize found minimum graphs in Mathcad were built.