Mapua University

School of Electrical, Electronics

and

Computer Engineering

COE60/C1

Machine Problem 1

User Manual

Sardina, Kent Johnric M.

2014150748

COE60/C1

Prof. Carlos V. Hortinela IV

1. Regula Falsi Method.

The Regula-Falsi Method (sometimes called the False Position Method) is a method used to find a numerical estimate of an equation.

This method attempts to solve an equation of the form f(x)=0. (This is very common in most numerical analysis applications.) Any equation can be written in this form. The converge process in the bisection method is very slow. It depends only on the choice of end points of the interval [a, b]. The function f(x) does not have any role in finding the point c (which is just the mid-point of a and. One can try for a better convergence-rate, at the risk of a worse one, or none.

Most numerical equation-solving methods usually converge faster than Bisection. The price for that is that some of them (e.g. Newton's method and Secant) can fail to converge at all, and all of them can sometimes converge much slower than Bisection—sometimes prohibitively slowly. None can guarantee Bisection's reliable and steady guaranteed convergence rate. Regula Falsi, like Bisection, always converges, usually considerably faster than Bisection—but sometimes much slower than Bisection. This algorithm requires a function f(x) and two points a and b for which f(x) is positive for one of the values and negative for the other.

Regula Falsi's Calculated Solution-Estimate Method:

The idea for the Regula-Falsi method is to connect the points (a, f(a)) and (b, f(b)) with a straight line.

Since linear equations are the simplest equations to solve for find the regula-falsi point (x_{rfp}) which is the solution to the linear equation connecting the endpoints.

Equation of the line:

$$y - f(a) = \frac{f(b) - f(a)}{b - a}(x - a)$$

Solving for X_{xfp}

$$0 - f(a) = \frac{f(b) - f(a)}{b - a}\left(x_{rfp} - a\right)$$

$$\frac{-f(a)(b-a)}{f(b) - f(a)} = x_{rfp} - a$$

$$x_{rfp} = a - \frac{f(a)(b-a)}{f(b) - f(a)}$$

Regula Falsi assumes that f(x) is linear—even though these methods are needed only when f(x) is *not* linear, and usually work well anyway.

The ratio of the change in x, to the resulting change in y is:

$$\frac{x_2 - x_1}{y_2 - y_1}$$

Because y, most recently, is y_2, and we want y to be 0, then the change that we want in y is:

0 - y_2

Of course, that's equal to –y2.

The latest value of x plus the product of the desired change in y and the expected ratio of change in x to change in y:

$$x_3 = x_2 + (-y_2)\frac{x_2 - x_1}{y_2 - y_1}$$

Not only is this form more simple and symmetrical, but it has a computational advantage:

As a solution is approached, x1 and x2 will be very close together, and nearly always of the same sign. Such a subtraction can lose significant digits.

Because y2 and y1 are always of opposite sign the "subtraction" in the numerator of the improved formula is effectively an addition (as is the subtraction in the denominator too).

Parts of the program:

I. Numerical method selection of MP 1 (Main Window)

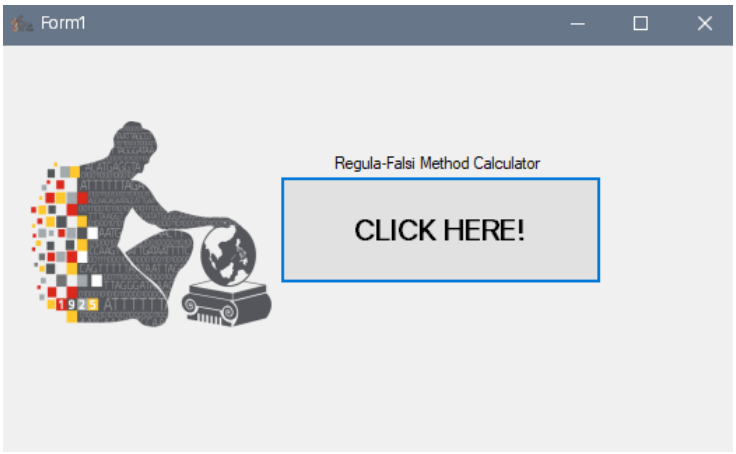


Figure 1. Main Window

This depicts what available numerical method to be used by the user. For Machine Problem 1, Regula-Falsi method is available.

II. Regular-Falsi Method Main Window

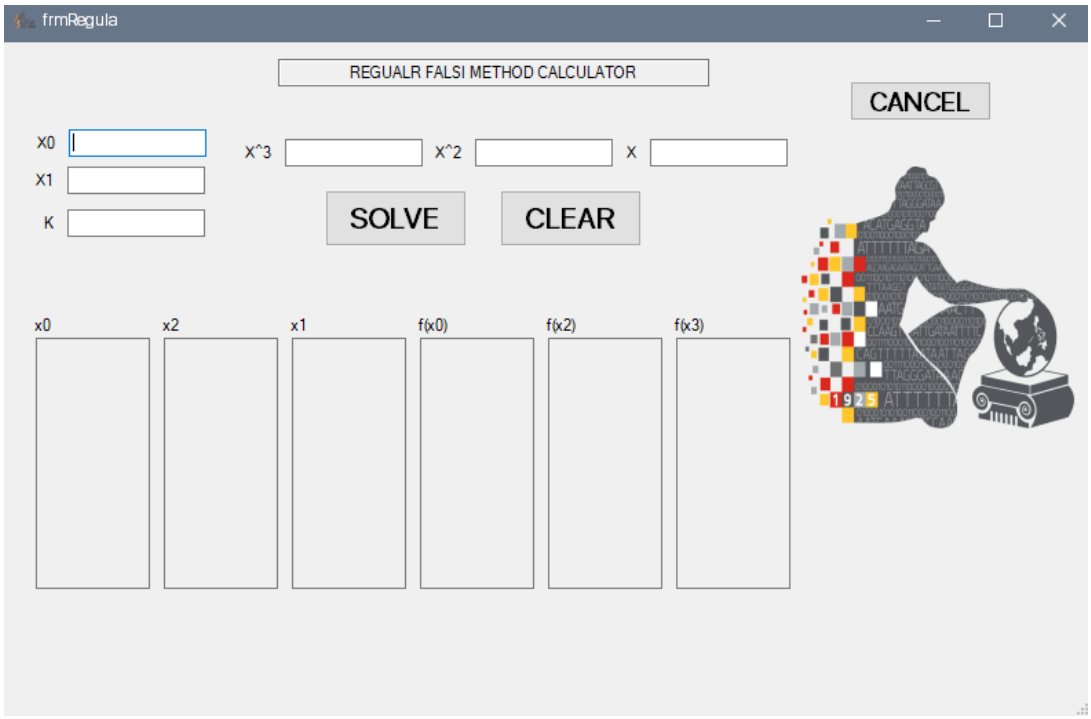
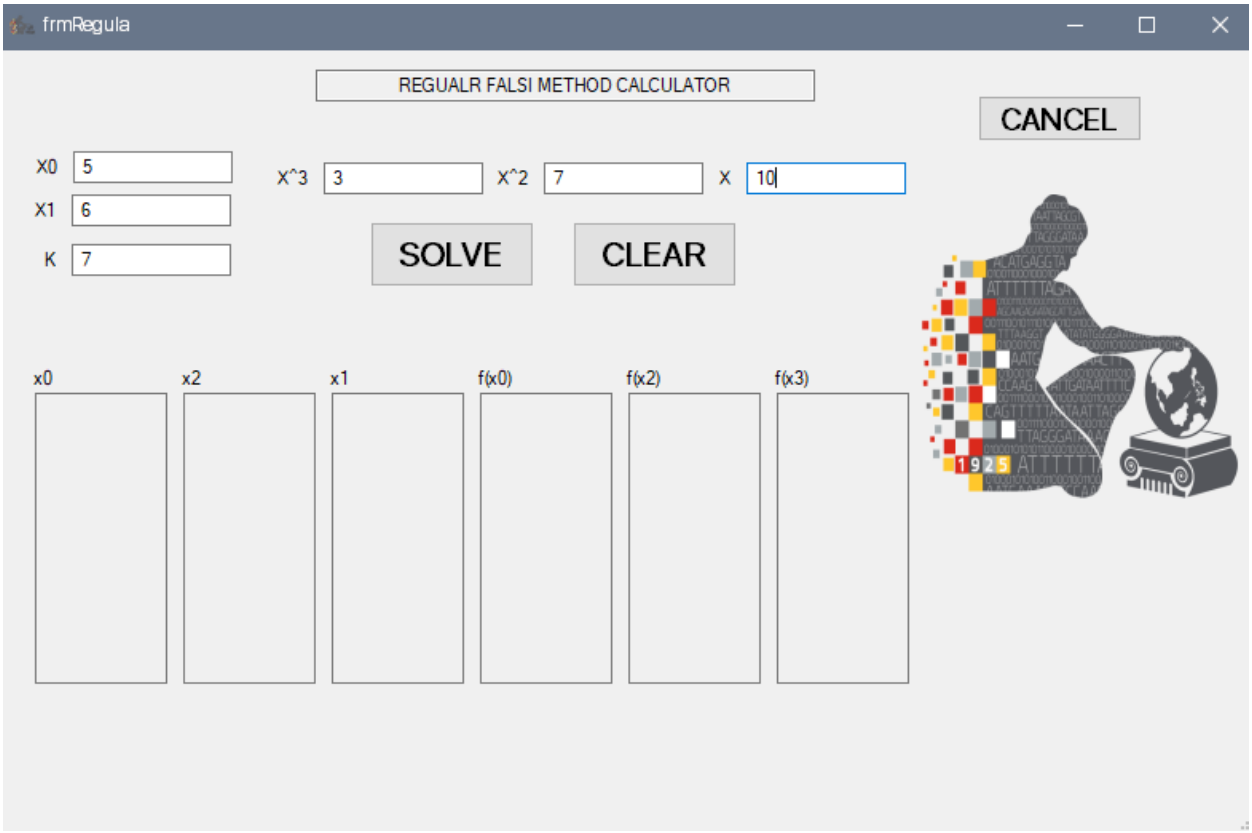


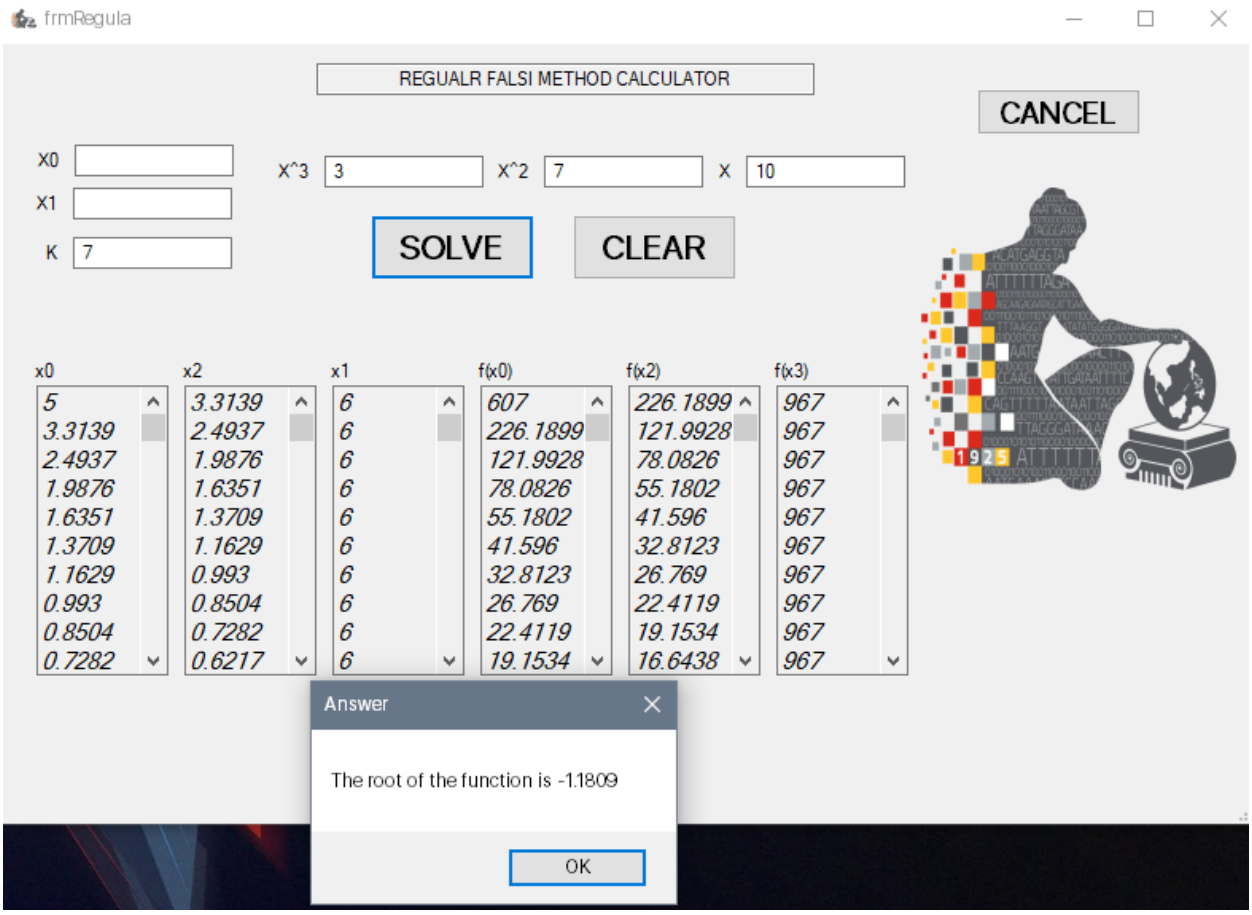
Figure 2. Regula-Falsi Method Calculator

This technique is applicable to both polynomial and transcendental equations, it is also known as Linear Interpolation or more often, as the method of false position. The method shown in Fig. 2 is called the Regula-Falsi method. It is a bracketing technique which requires 2 initial guesses for the root, with the general rule that states that one of them must be positive and the other negative. The two guesses are then substituted into an iterative formula which “brackets” on either side of the root and produces another value. Depending on the sign of the value, it will replace either the positive or negative producing guess from before and the iterative formula repeated until the terminating condition is met.

Steps to use the program:

1. Input initial values of x0, x1, and K.
2. Type in the coefficient values of your function.
3. Click solve.





4. The six-list box at the bottom shows the entire iteration works towards the process.
5. A dialog box appears after the iterations are shown and it displays the root of the function given by the user.
6. You can click clear to remove all values and enter new values and Cancel to go back to the main window.

Source Code:

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;

namespace MachineProblem1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         MachineProblem1.frmRegula form = new MachineProblem1.frmRegula();
         form.ShowDialog();
      }
   }
```

```
}
```

frmRegula.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;

namespace MachineProblem1
{
   public partial class frmRegula : Form
   {
      int i = 0, count = 0, flag = 0;
      double error = 0.0001;
      double[] coeff = new double[10];
      double x0, x1, x2 = 0, t = 0;

      private void btnRegula_clear_Click(object sender, EventArgs e)
      {
         txtBoxRegula_int1.Text = " ";
         txtBoxRegula_int2.Text = " ";
         txtBoxRegula_x1.Text = " ";
         txtBoxRegula_x2.Text = " ";
         txtBoxRegula_x3.Text = " ";
```

```
    txtBoxRegula_k.Text = " ";

    lBoxRegula_x0.Items.Clear();
    lBoxRegula_x1.Items.Clear();
    lBoxRegula_x2.Items.Clear();
    lBoxRegula_fx0.Items.Clear();
    lBoxRegula_fx1.Items.Clear();
    lBoxRegula_fx2.Items.Clear();
}

private void btnRegula_back_Click(object sender, EventArgs e)
{
    this.Close();
}

double fx0 = 0, fx1 = 0, fx2 = 0, temp = 0;
int test = 0;

public frmRegula()
{
    InitializeComponent();
}

public int check()
{

    x0 = double.Parse(txtBoxRegula_int1.Text);
    x1 = double.Parse(txtBoxRegula_int2.Text);
```

```
    fx0 = fx1 = fx2 = 0;

    for (i = 3; i >= 1; i--)
    {

        fx0 += coeff[i] * (Math.Pow(x0, i));
        fx1 += coeff[i] * (Math.Pow(x1, i));

    }

    fx0 += coeff[0];
    fx1 += coeff[0];

    if ((fx0 * fx1) > 0)
    {

        //MessageBox.Show("Input another set of initial values.");
        return (1);

    }
    return (0);

}

private void btnRegula_solve_Click(object sender, EventArgs e)
{
    coeff[0] = double.Parse(txtBoxRegula_k.Text);
    coeff[1] = double.Parse(txtBoxRegula_x1.Text);
    coeff[2] = double.Parse(txtBoxRegula_x2.Text);
```

```
coeff[3] = double.Parse(txtBoxRegula_x3.Text);

do
{

   test = check();

   if (test != 0)
   {

      txtBoxRegula_int1.Text = " ";
      txtBoxRegula_int2.Text = " ";
      break;

   }

} while (check() != 0);

flag = 1;

if (flag == 1)
{

   do
   {

      count++;
      fx0 = fx1 = fx2 = 0;
```

```
lBoxRegula_x0.Items.Add(Math.Round(x0, 4));
lBoxRegula_x2.Items.Add(Math.Round(x1, 4));

for (i = 3; i >= 1; i--)
{

   fx0 += coeff[i] * (Math.Pow(x0, i));
   fx1 += coeff[i] * (Math.Pow(x1, i));

}

fx0 += coeff[0];
lBoxRegula_fx0.Items.Add(Math.Round(fx0, 4));

fx1 += coeff[0];
lBoxRegula_fx2.Items.Add(Math.Round(fx1, 4));



temp = x2;

x2 = ((x0 * fx1) - (x1 * fx0)) / (fx1 - fx0);
lBoxRegula_x1.Items.Add(Math.Round(x2, 4));

for (i = 3; i >= 1; i--)
{

   fx2 += coeff[i] * (Math.Pow(x2, i));
```

```
            }
            fx2 += coeff[0];
            lBoxRegula_fx1.Items.Add(Math.Round(fx2, 4));

            t = fx0 * fx2;

            if (t > 0)
            {

               x0 = x2;

            }

            if (t < 0)
            {

               x1 = x2;

            }
            fx2 = 0;



        } while ((Math.Abs(temp - x2)) >= error);

        MessageBox.Show("The root of the function is " + Math.Round(x2, 4), "Answer");
    }
}
```

```
        }
    }
```

References:

https://mat.iitm.ac.in/home/sryedida/public_html/caimna/transcendental/bracketing%20methods/regula-falsi/regula-falsi.html

https://en.wikipedia.org/wiki/False_position_method

Lecture Notes in Numerical Methods Edition No.1