Mapua University

School of Electrical, Electronics

and

Computer Engineering

COE60/C1

Machine Problem 3

User Manual

Sardina, Kent Johnric M.

2014150748

COE60/C1

Prof. Carlos V. Hortinela IV

Muller’s Method

Muller's method is a root-finding algorithm, a numerical method for solving equations of the form $f(x) = 0$. It was first presented by David E. Muller in 1956.

Muller's method is based on the secant method, which constructs at every iteration a line through two points on the graph of f. Instead, Muller's method uses three points, constructs the parabola through these three points, and takes the intersection of the x-axis with the parabola to be the next approximation. Muller's method is a generalization of the secant method. Instead of starting with two initial values and then joining them with a straight line in secant method, Mullers method starts with three initial approximations to the root and then join them with a second-degree polynomial (a parabola), then the quadratic formula is used to find a root of the quadratic for the next approximation. That is if x_0, x_1 and x_2 are the initial approximations then x_3 is obtained by solving the quadratic which is obtained by means of x_0, x_1 and x_2. Then two values among x_0, x_1 and x_2 which are close to x_3 are chosen for the next iteration.

$x_3 = x_2 + z$ (*)

where $z = -2c$

$b \pm$Ö (b^2-4ac)

$a = D_1/D_2$, $b = D_2/D$ and $c = f(x_2)$

$D = h_0h_1(h_0-h_1)$, $D_1 = (f_0-c)\, h_1-(f_1-c)\, h_0$, $D_2 = (f_1 -c)\, h_0^2 - (f_0-c)\, h_1^2$

$h_0 = x_0-x_2$, $h_1 = x_1-x_2$

Crout’s Method

Crout’s method also called as Cholesky Method. In linear algebra, the Cholesky decomposition or Cholesky factorization is a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose, which is useful e.g. for efficient numerical solutions and Monte Carlo simulations. It was discovered by André-Louis Cholesky for real matrices. When it is applicable, the Cholesky decomposition is roughly twice as e in linear algebra, the Crout matrix decomposition is an LU decomposition which decomposes a matrix into a lower triangular matrix (L), an upper triangular matrix (U) and, although not always needed, a permutation matrix (P). It was developed by Prescott Durand Crout. [1]

The Crout matrix decomposition algorithm differs slightly from the Doolittle method. Doolittle's method returns a unit lower triangular matrix and an upper triangular matrix, while the Crout method returns a lower triangular matrix and a unit upper triangular matrix.

So, if a matrix decomposition of a matrix A is such that:

$A = LDU$

being L a unit lower triangular matrix, D a diagonal matrix and U a unit upper triangular matrix, then Doolittle's method produces

$A = L(DU)$

and Crout's method produces

A = (LD)U.

being L a lower triangular matrix, D a diagonal matrix and U a normalised upper triangular matrixfficient as the LU decomposition for solving systems of linear equations.

Gauss – Seidel Method

In numerical methods, the Gauss–Seidel method, also known as the Liebman method or the method of successive displacement, is an iterative method used to solve a linear system of equations. It is named after the German mathematicians Carl Friedrich Gauss and Philipp Ludwig von Seidel, and is like the Jacobi method. Though it can be applied to any matrix with non-zero elements on the diagonals, convergence is only guaranteed if the matrix is either diagonally dominant, or symmetric and positive definite. It was only mentioned in a private letter from Gauss to his student Gerling in 1823. A publication was not delivered before 1874 by Seidel. The element-wise formula for the Gauss–Seidel method is extremely like that of the Jacobi method.

The computation of xi(k+1) uses only the elements of x(k+1) that have already been computed, and only the elements of x(k) that have not yet to be advanced to iteration k+1. This means that, unlike the Jacobi method, only one storage vector is required as elements can be overwritten as they are computed, which can be advantageous for very large problems.

However, unlike the Jacobi method, the computations for each element cannot be done in parallel. Furthermore, the values at each iteration are dependent on the order of the original equations.

Gauss-Seidel is the same as SOR (successive over-relaxation)

Parts of the Program:

I. Numerical method selection for MP3 (Main Window)

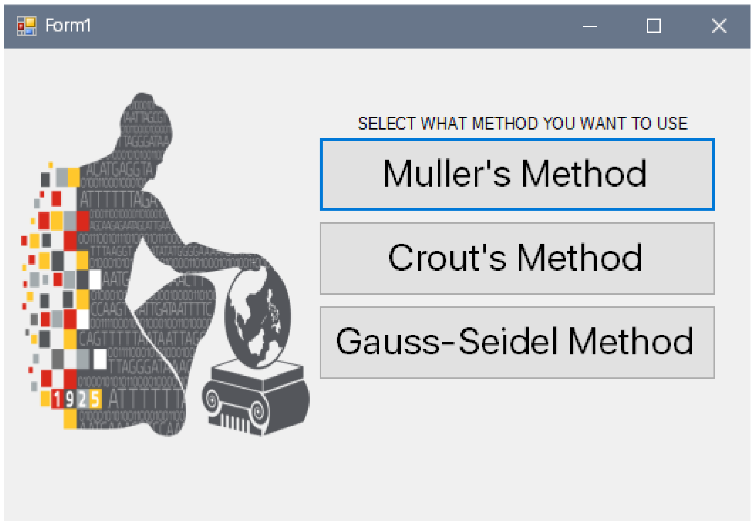


Figure 1. Main Menu

This depicts the available numerical methods to be used by the user. For Machine Problem 3, Muller's, Crout's, and Gauss Seidel is available.

II. Muller’s Method Window

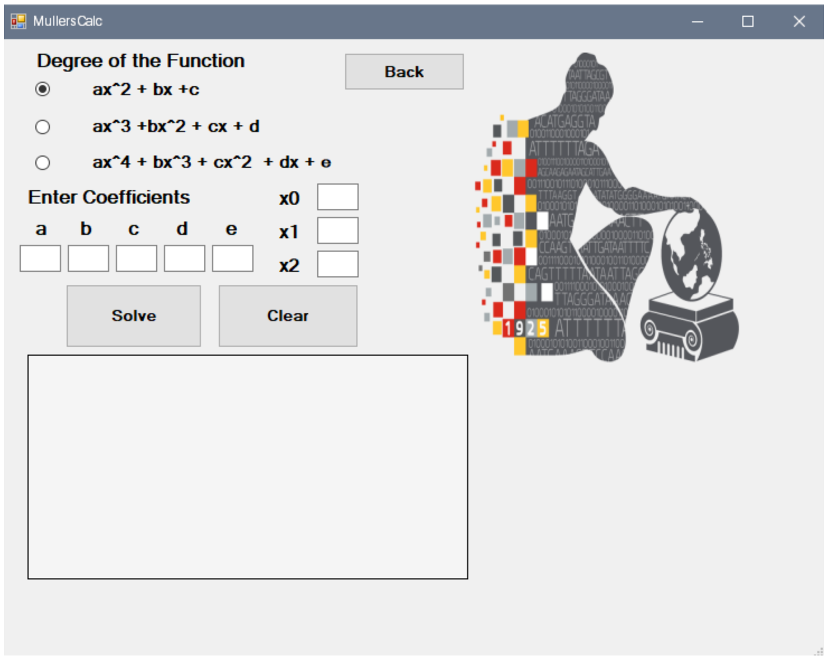


Figure 2. Window for Muller’s Method.

III. Crout’s Method Window

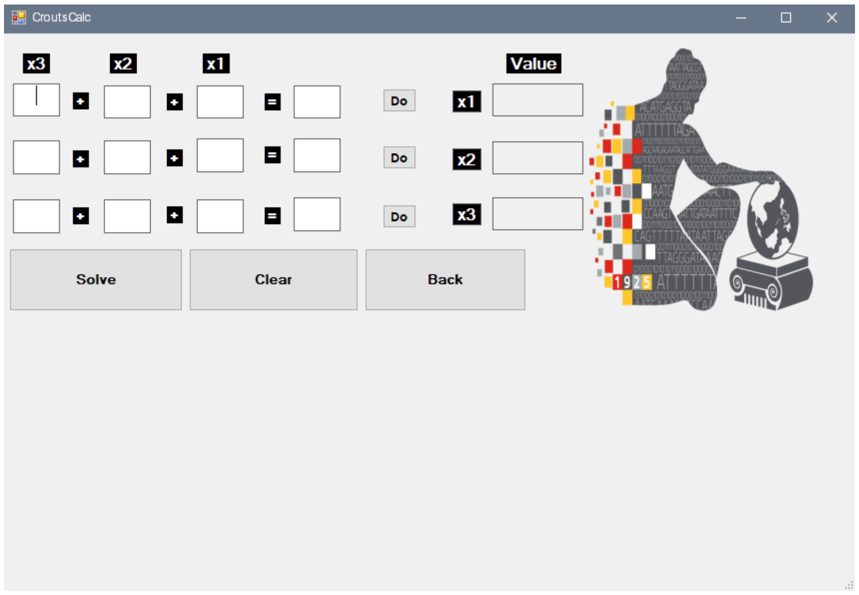


Figure 3. Window for Crout’s Method Program.

IV. Gauss-Seidel Window

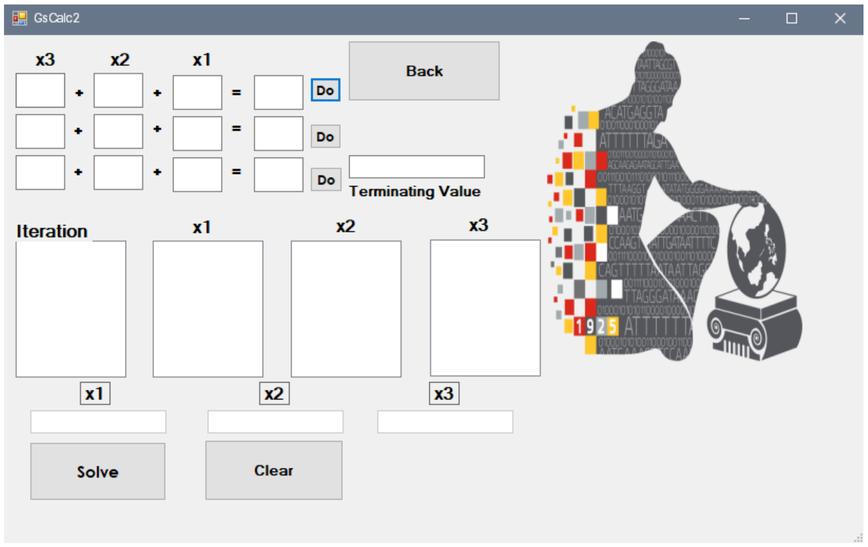
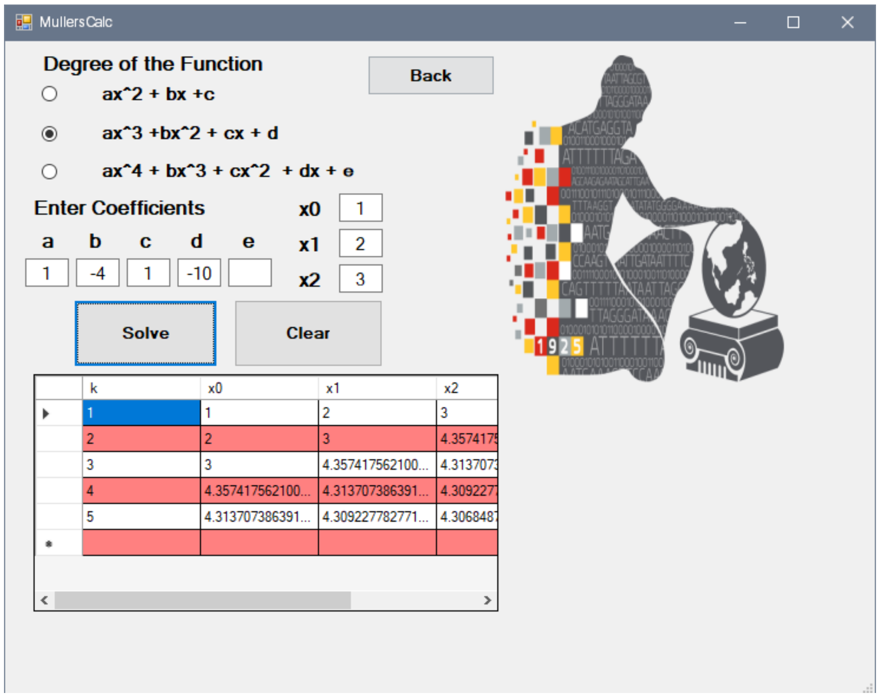


Figure 4. Window for Gauss-Seidel Program

Steps to use the program:

I. Muller's Method

1. First you must choose your desired degree of function, may it be a quadratic, cubic or polynomial function.
2. Enter the coefficient of a, b, c, d, and e. Enter also any initial value.
3. Click Solve.
4. If the coefficient you've entered is correct, then it will successfully show you all iterations.



Degree of the Function

( ) ax^2 + bx +c

[✓ ax^3 +bx^2 + cx + d]

( ) ax^4 + bx^3 + cx^2 + dx + e

Enter Coefficients

a	b	c	d	e
1	-4	1	-10	

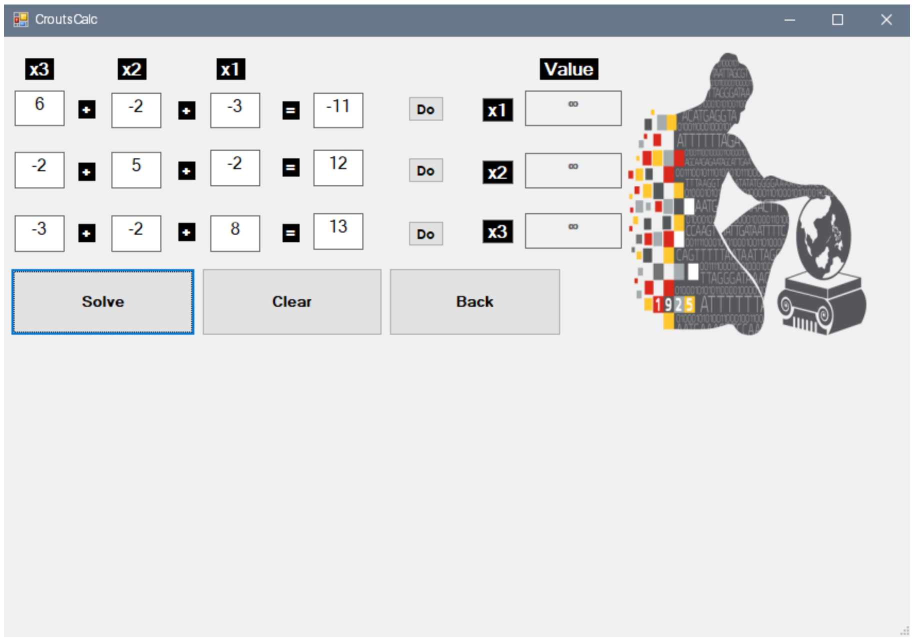
x0: 1
x1: 2
x2: 3

k	x0	x1	x2
1	1	2	3
2	2	3	4.3574175
3	3	4.357417562100...	4.3137073
4	4.357417562100...	4.313707386391...	4.3092277
5	4.313707386391...	4.309227782771...	4.3068487

5. You may click "Clear" if you wanted to enter another set of value for coefficient or the initial values.
6. Click "Back" if you want to try another method.

II. Crout's Method

1. In Crout's Method, 3 equations are always given to solve for the value of X_1,X_2, and X_3. First, enter the coefficients of the 3 equations that will be used for the Matrix Decomposition.
2. Click the "DO" box when you've filled all the boxes.
3. Click Solve.



x3		x2		x1					Value
6	+	-2	+	-3	=	-11	Do	x1	∞
-2	+	5	+	-2	=	12	Do	x2	∞
-3	+	-2	+	8	=	13	Do	x3	∞

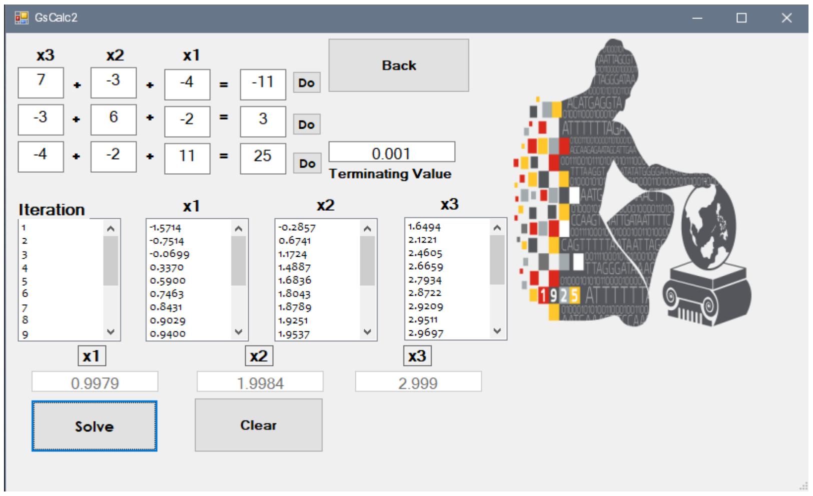
4. Then the value for X_1, X_2, and X_3 will appear

5. You may click "Clear" if you wanted to enter another set of value for coefficient or the initial values. Click "Back" if you want to try another method.

III. Gauss-Seidel Method

1. In Gauss-Seidel Method, 3 equations are always given to solve for the value of X_1,X_2, and X_3. First, enter the coefficients of the 3 equations that will be used. Make sure that X_1 is the highest for the first equation, X_2 for the second equation and X_3 for the last equation.
2. Click the "DO" box once you've filled all the boxes.
3. Enter your desired terminating condition.

4. Click Solve


GsCalc2

x3		x2		x1		
7	+	-3	+	-4	=	-11
-3	+	6	+	-2	=	3
-4	+	-2	+	11	=	25

0.001 Terminating Value

Iteration	x1	x2	x3
1	-1.5714	-0.2857	1.6494
2	-0.7514	0.6741	2.1221
3	-0.0699	1.1724	2.4605
4	0.3370	1.4887	2.6659
5	0.5900	1.6836	2.7934
6	0.7463	1.8043	2.8722
7	0.8431	1.8789	2.9209
8	0.9029	1.9251	2.9511
9	0.9400	1.9537	2.9697

x1: 0.9979
x2: 1.9984
x3: 2.999

5. Then it will show the total number of iterations and the iteration the equation performed.
6. You may click “Clear” if you wanted to enter another set of value for coefficient or the initial values. Click “Back” if you want to try another method.

Source Codes:

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 MachineProblem3

{

    public partial class MainWindow : Form

    {

        public MainWindow()

        {

            InitializeComponent();

        }


        private void MullerBTN_Click(object sender, EventArgs e)

        {

            MachineProblem3.MullersCalc form = new MachineProblem3.MullersCalc();

            form.ShowDialog();

        }


        private void CroutsBTN_Click(object sender, EventArgs e)

        {

            MachineProblem3.CroutsCalc form = new MachineProblem3.CroutsCalc();

            form.ShowDialog();

        }


        private void GSBTN_Click(object sender, EventArgs e)
        {
            MachineProblem3.GsCalc2 form = new
MachineProblem3.GsCalc2();
            form.ShowDialog();
        }
    }
}
```

MullersCalc.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 MachineProblem3
{
    public partial class MullersCalc : Form
    {
        public MullersCalc()
        {
            InitializeComponent();
        }

        void clear()
        {
            txtBoxMullers_a.Text = "";
            txtBoxMullers_b.Text = "";
            txtBoxMullers_c.Text = "";
            txtBoxMullers_d.Text = "";
            txtBoxMullers_e.Text = "";
            txtBoxMullers_x0.Text = "";
            txtBoxMullers_x1.Text = "";
            txtBoxMullers_x2.Text = "";
        }
        private void Form1_Load(object sender,
EventArgs e)
        {
            clear();
            radBtnMullers_c1.Checked = false;
            radBtnMullers_c2.Checked = false;
            radBtnMullers_c3.Checked = false;
            //  gBoxMullers_values.Show();
        }

        private void btnMullers_clear_Click(object
sender, EventArgs e)
        {
            clear();
            radBtnMullers_c1.Checked = false;
            radBtnMullers_c2.Checked = false;
            radBtnMullers_c3.Checked = false;
            // gBoxMullers_values.Hide();
            dgvMullers.Hide();
        }
        private void
radBtnMullers_c1_CheckedChanged(object
sender, EventArgs e)
        {
            clear();
            //gBoxMullers_values.Hide();
            txtBoxMullers_d.Hide();
            txtBoxMullers_e.Hide();
            lbl_d.Hide();
            lbl_e.Hide();
            dgvMullers.Hide();
        }

        private void
radBtnMullers_c2_CheckedChanged(object
sender, EventArgs e)
        {
            clear();
            // gBoxMullers_values.Show();
            txtBoxMullers_d.Show();
            txtBoxMullers_e.Hide();
            lbl_e.Show();
            lbl_d.Hide();
            dgvMullers.Hide();
        }

        private void
radBtnMullers_c3_CheckedChanged(object
sender, EventArgs e)
        {
            clear();
            //  gBoxMullers_values.Show();
            txtBoxMullers_d.Show();
            txtBoxMullers_e.Show();
            lbl_e.Show();
            lbl_d.Show();
            dgvMullers.Hide();
        }

        private void btnMullers_solve_Click(object
sender, EventArgs e)
        {
            double s1, s0, h1, h0, a, b, c, d, f, x1, x0,
x2, x3, f0, f1, f2, error = 100, k = 0, a1, b1, c1,
d1;
            double[] data = new double[5];

            DataTable table = new DataTable();
            table.Columns.Add("k", typeof(double));
            table.Columns.Add("x0", typeof(double));
            table.Columns.Add("x1", typeof(double));
            table.Columns.Add("x2", typeof(double));
            table.Columns.Add("x3", typeof(double));

            x0 =
double.Parse(txtBoxMullers_x0.Text);
            x1 =
double.Parse(txtBoxMullers_x1.Text);
            x2 =
double.Parse(txtBoxMullers_x2.Text);
            a = double.Parse(txtBoxMullers_a.Text);
            b = double.Parse(txtBoxMullers_b.Text);

            c = double.Parse(txtBoxMullers_c.Text);


            if (radBtnMullers_c1.Checked == true)

            {

                while (error > .0001)

                {

                    f0 = (a * (x0 * x0)) + (b * x0) + c;

                    f1 = (a * (x1 * x1)) + (b * x1) + c;

                    f2 = (a * (x2 * x2)) + (b * x2) + c;

                    h0 = x1 - x0;

                    h1 = x2 - x1;

                    s0 = (f1 - f0) / h0;

                    s1 = (f2 - f1) / h1;

                    a1 = (s1 - s0) / h1 + h0;

                    b1 = (a1 * h1) + s1;

                    c1 = f2;

                    d1 = (b1 * b1) - (4 * a1 * c1);

                    if (d1 > 0)

                    {

                        d1 = Math.Sqrt(d1);

                        if (Math.Abs(b1 + d1) >
Math.Abs(b1 - d1))

                        {

                            x3 = x2 + ((-2 * c1) / (b1 + d1));

                        }

                        else

                        {

                            x3 = x2 + ((-2 * c1) / (b1 - d1));

                        }

                        error = Math.Abs((x3 - x2));

                        k++;

                        table.Rows.Add(k, x0, x1, x2, x3);

                        x0 = x1;

                        x1 = x2;

                        x2 = x3;

                    }

                    else

                    {

                        MessageBox.Show("The data gives
an imaginary number. Please change the
coefficients of your equation.", "Error",
MessageBoxButtons.OK);

                        error = 0;

                    }

                };

                dgvMullers.Visible = true;

                dgvMullers.DataSource = table;

            }

            else if (radBtnMullers_c2.Checked ==
true)

            {

                while (error > .0001)

                {

                    d =
double.Parse(txtBoxMullers_d.Text);

                    f0 = (a * (x0 * x0 * x0)) + (b * x0 * x0)
+ (c * x0) + d;

                    f1 = (a * (x1 * x1 * x1)) + (b * x1 * x1)
+ (c * x1) + d;
```

```
                f2 = (a * (x2 * x2 * x2)) + (b * x2 * x2)
+ (c * x2) + d;
                h0 = x1 - x0;
                h1 = x2 - x1;
                s0 = (f1 - f0) / h0;
                s1 = (f2 - f1) / h1;
                a1 = (s1 - s0) / h1 + h0;
                b1 = (a1 * h1) + s1;
                c1 = f2;
                d1 = (b1 * b1) - (4 * a1 * c1);
                if (d1 > 0)
                {
                    d1 = Math.Sqrt(d1);
                    if (Math.Abs(b1 + d1) >
Math.Abs(b1 - d1))
                    {
                        x3 = x2 + ((-2 * c1) / (b1 + d1));
                    }
                    else
                    {
                        x3 = x2 + ((-2 * c1) / (b1 - d1));
                    }
                    error = Math.Abs((x3 - x2));
                    k++;
                    table.Rows.Add(k, x0, x1, x2, x3);
                    x0 = x1;
                    x1 = x2;
                    x2 = x3;
                }
                else
                {
                    MessageBox.Show("The data gives
an imaginary number. Please change the
coefficients of your equation.", "Error",
MessageBoxButtons.OK);
                    error = 0;
                }
            };
            dgvMullers.Visible = true;
            dgvMullers.DataSource = table;
        }
        else if (radBtnMullers_c3.Checked ==
true)
        {


            while (error > .0001)
            {
                d =
double.Parse(txtBoxMullers_d.Text);
                f =
double.Parse(txtBoxMullers_e.Text);
                f0 = (a * (x0 * x0 * x0 * x0)) + (b * x0
* x0 * x0) + (c * x0 * x0) + (d * x0) + f;
                f1 = (a * (x1 * x1 * x1 * x1)) + (b * x1
* x1 * x1) + (c * x1 * x1) + (d * x1) + f;
                f2 = (a * (x2 * x2 * x2 * x2)) + (b * x2
* x2 * x2) + (c * x2 * x2) + (d * x2) + f;
                h0 = x1 - x0;
                h1 = x2 - x1;
                s0 = (f1 - f0) / h0;
                s1 = (f2 - f1) / h1;
            a1 = (s1 - s0) / h1 + h0;

            b1 = (a1 * h1) + s1;

            c1 = f2;

            d1 = (b1 * b1) - (4 * a1 * c1);

            if (d1 > 0)

            {

              d1 = Math.Sqrt(d1);

              if (Math.Abs(b1 + d1) >
Math.Abs(b1 - d1))

              {

                x3 = x2 + ((-2 * c1) / (b1 + d1));

              }

              else

              {

                x3 = x2 + ((-2 * c1) / (b1 - d1));

              }

              error = Math.Abs((x3 - x2));

              k++;

              table.Rows.Add(k, x0, x1, x2, x3);

              x0 = x1;

              x1 = x2;

              x2 = x3;

            }

            else

            {

              MessageBox.Show("The data gives
an imaginary number. Please change the
coefficients of your equation.", "Error",
MessageBoxButtons.OK);

              error = 0;

            }

          };

          dgvMullers.Visible = true;

          dgvMullers.DataSource = table;

        }

      }


      private void btnMullers_back_Click(object
sender, EventArgs e)

      {

        this.Close();

      }

    }

}
```

CroutsCalc.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 MachineProblem3

{

  public partial class CroutsCalc : Form

  {
        double[,] A = new double[3, 4];

        double y1, y2, y3, x1, x2, x3;

        double[,] L = new double[3, 3];


        private void
btnCholeskys_clear_Click(object sender,
EventArgs e)

        {

            textBox1.Text = "";

            textBox2.Text = "";

            textBox3.Text = "";

            textBox4.Text = "";

            textBox5.Text = "";

            textBox6.Text = "";

            textBox7.Text = "";

            textBox8.Text = "";

            textBox9.Text = "";

            textBox10.Text = "";

            textBox11.Text = "";

            textBox12.Text = "";

            txtBoxCholeskys_x1.Text = "";

            txtBoxCholeskys_x2.Text = "";

            txtBoxCholeskys_x3.Text = "";

        }


        private void
btnCholeskys_back_Click(object sender,
EventArgs e)

        {

            this.Close();

        }


        private void
btnCholeskys_1st_Click_1(object sender,
EventArgs e)

        {

            for (int i = 0; i < 3; i++)

            {

                for (int j = 0; j < 3; j++)

                {

                    L[i, j] = 0;

                    U[i, j] = 0;

                }

            }


            U[0, 0] = 1;

            U[1, 1] = 1;

            U[2, 2] = 1;


            A[0, 0] = double.Parse(textBox1.Text);

            A[0, 1] = double.Parse(textBox2.Text);

            A[0, 2] = double.Parse(textBox3.Text);

            A[0, 3] = double.Parse(textBox4.Text);


            if (A[0, 0] < A[0, 1] || A[0, 0] < A[0, 2])

            {

                MessageBox.Show("Third-order
coefficient should be the largest!");

                for (int i = 0; i < 4; i++)

                {
```

```
            A[0, i] = 0;
          }

        }
        else
        {
          flag1 = true;
        }
      }

      private void btnCholeskys_2nd_Click(object
sender, EventArgs e)
      {
        A[1, 0] = double.Parse(textBox5.Text);
        A[1, 1] = double.Parse(textBox6.Text);
        A[1, 2] = double.Parse(textBox7.Text);
        A[1, 3] = double.Parse(textBox8.Text);

        if (A[1, 1] < A[1, 0] || A[1, 1] < A[1, 2])
        {
          MessageBox.Show("Second-order
coefficient should be the largest!");
          for (int i = 0; i < 4; i++)
          {
            A[1, i] = 0;
          }

          textBox5.Text = "";
          textBox6.Text = "";
          textBox7.Text = "";
          textBox8.Text = "";
        }
        else
        {
          flag2 = true;
        }
      }

      private void btnCholeskys_3rd_Click(object
sender, EventArgs e)
      {
        A[2, 0] = double.Parse(textBox9.Text);
        A[2, 1] = double.Parse(textBox10.Text);
        A[2, 2] = double.Parse(textBox11.Text);
        A[2, 3] = double.Parse(textBox12.Text);

        if (A[2, 2] < A[2, 0] || A[2, 2] < A[2, 1])
        {
          MessageBox.Show("First-order
coefficient should be the largest!");
          for (int i = 0; i < 4; i++)
          {
            A[2, i] = 0;
          }

          textBox9.Text = "";
          textBox10.Text = "";
          textBox11.Text = "";
```

```
            textBox12.Text = "";
        }
        else
        {
            flag3 = true;
        }
    }

    double[,] U = new double[3, 3];
    bool flag1 = false, flag2 = false, flag3 = false;

    public CroutsCalc()
    {
        InitializeComponent();
    }

    private void
btnCholeskys_solve_Click(object sender,
EventArgs e)
    {
        if ((flag1 && flag2 && flag3) == true)
        {
            for (int i = 0; i < 3; i++)
            {
                L[i, 0] = A[i, 0];
            }
            for (int i = 1; i < 3; i++)
            {
                U[0, i] = (A[0, i]) / (L[0, 0]);
            }
            for (int i = 1; i < 3; i++)
            {
                L[i, 1] = A[i, 1] - (L[i, 0]) * (U[0, 1]);
            }
            U[1, 2] = (A[1, 2] - L[1, 0] * U[0, 2]) /
L[1, 1];

            y1 = A[0, 3] / L[0, 0];
            y2 = (A[1, 3] + (-1 * L[1, 0] * y1)) / L[1,
1];
            y3 = ((-1 * y1 * L[2, 0]) + (-1 * L[2, 1] *
y2) + A[2, 3]) / L[2, 2];



            x3 = y3;
            x2 = y2 + (x3 * -1 * U[1, 2]);
            x1 = y1 + (-1 * U[0, 1] * x2) + (-1 * U[0,
2] * x3);

            txtBoxCholeskys_x1.Text =
Convert.ToString(Math.Round(x1, 4));
            txtBoxCholeskys_x2.Text =
Convert.ToString(Math.Round(x2, 4));
            txtBoxCholeskys_x3.Text =
Convert.ToString(Math.Round(x3, 4));
        }
        else
        {
            MessageBox.Show("You must click ALL
the 'DO' button first.");
        }
    }

    private void btnCholeskys_1st_Click(object
sender, EventArgs e)
    {
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                L[i, j] = 0;
                U[i, j] = 0;
            }
        }

        U[0, 0] = 1;
        U[1, 1] = 1;
        U[2, 2] = 1;

        A[0, 0] = double.Parse(textBox1.Text);
        A[0, 1] = double.Parse(textBox2.Text);
        A[0, 2] = double.Parse(textBox3.Text);
        A[0, 3] = double.Parse(textBox4.Text);

        if (A[0, 0] < A[0, 1] || A[0, 0] < A[0, 2])
        {
            MessageBox.Show("Third-order
coefficient should be the largest!");
            for (int i = 0; i < 4; i++)
            {
                A[0, i] = 0;
            }

        }
        else
        {
            flag1 = true;
        }
    }
}
}
```

GsCalc2.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 MachineProblem3
{
    public partial class GsCalc2 : Form
    {
        double[] first = new double[10];

        double[] second = new double[10];

        double[] third = new double[10];

        double epsilon;

        double v1 = 0, v2 = 0, v3 = 0, temp1, temp2,
temp3, tempf1, tempf2, tempf3;

        int iter = 0;

        bool flag1 = false, flag2 = false, flag3 = false;

        public GsCalc2()

        {

            InitializeComponent();

        }


        private void btnGS_solve_Click(object
sender, EventArgs e)

        {

            if ((flag1 && flag2 && flag3) == true)

            {

                epsilon =
double.Parse(textBox20.Text);


                do

                {

                    temp1 = v1;

                    temp2 = v2;

                    temp3 = v3;


                    iter++;

                    v1 = (first[3] + (-1 * first[1] * v2) + (-1
* first[2] * v3)) / first[0];

                    v2 = (second[3] + (-1 * second[0] *
v1) + (-1 * second[2] * v3)) / second[1];

                    v3 = (third[3] + (-1 * third[0] * v1) +
(-1 * third[1] * v2)) / third[2];


                    listBox1.Items.Add(iter);

listBox2.Items.Add(v1.ToString("F4"));

listBox3.Items.Add(v2.ToString("F4"));

listBox4.Items.Add(v3.ToString("F4"));


                    tempf1 = Math.Abs(temp1 - v1);

                    tempf2 = Math.Abs(temp2 - v2);

                    tempf3 = Math.Abs(temp3 - v3);

                } while (tempf1 > epsilon && tempf2 >
epsilon && tempf3 > epsilon);


                txtBoxGS_val1.Text =
Convert.ToString(Math.Round(v1, 4));

                txtBoxGS_val2.Text =
Convert.ToString(Math.Round(v2, 4));

                txtBoxGS_val3.Text =
Convert.ToString(Math.Round(v3, 4));

            }

            else

            {

                MessageBox.Show("You must click ALL
the 'DO' button first.");

            }
        }


        private void btnGS_Clear_Click(object
sender, EventArgs e)

        {

            txtBoxGS_val1.Text = "";

            txtBoxGS_val2.Text = "";

            txtBoxGS_val3.Text = "";

            listBox1.Items.Clear();

            listBox2.Items.Clear();

            listBox3.Items.Clear();

            listBox4.Items.Clear();

            textBox1.Text = "";

            textBox2.Text = "";

            textBox3.Text = "";

            textBox4.Text = "";

            textBox5.Text = "";

            textBox6.Text = "";

            textBox7.Text = "";

            textBox8.Text = "";

            textBox9.Text = "";

            textBox10.Text = "";

            textBox11.Text = "";

            textBox12.Text = "";

            textBox20.Text = "";

        }

        private void btnGS_Back_Click(object
sender, EventArgs e)

        {

            this.Close();

        }


        private void btnGS_func1_Click(object
sender, EventArgs e)

        {

            first[0] = double.Parse(textBox1.Text);

            first[1] = double.Parse(textBox2.Text);

            first[2] = double.Parse(textBox3.Text);

            first[3] = double.Parse(textBox4.Text);


            if (first[0] < first[1] || first[0] < first[2])

            {

                MessageBox.Show("X3 must have the
Largest Value");

                for (int i = 0; i < 4; i++)

                {

                    first[i] = 0;

                }


            }

            else

            {

                flag1 = true;

            }

        }
        private void btnGS_func2_Click(object
sender, EventArgs e)
        {
            second[0] = double.Parse(textBox5.Text);
            second[1] = double.Parse(textBox6.Text);
            second[2] = double.Parse(textBox7.Text);
            second[3] = double.Parse(textBox8.Text);

            if (second[1] < second[0] || second[1] <
second[2])
            {
                MessageBox.Show("X2 must have the
Largest Value");
                for (int i = 0; i < 4; i++)
                {
                    second[i] = 0;
                }

            }
            else
            {
                flag2 = true;
            }
        }

        private void btnGS_func3_Click(object
sender, EventArgs e)
        {
            third[0] = double.Parse(textBox9.Text);
            third[1] = double.Parse(textBox10.Text);
            third[2] = double.Parse(textBox11.Text);
            third[3] = double.Parse(textBox12.Text);

            if (third[2] < third[0] || third[2] <
third[1])
            {
                MessageBox.Show("X1 must have the
Largest Value");
                for (int i = 0; i < 4; i++)
                {
                    third[i] = 0;
                }
            }
            else
            {
                flag3 = true;
            }
        }
    }
}
```

References

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

https://en.wikipedia.org/wiki/Muller%27s_method

https://en.wikipedia.org/wiki/Gauss%E2%80%93Seidel_method