



Mapua University
School of Electrical, Electronics
and
Computer Engineering

COE60/C1

Machine Problem 4
User Manual

Sardina, Kent Johnric M.

2014150748

COE60/C1

Prof. Carlos V. Hortinela IV

Linear Regression Method

Linear regression is the most basic type of regression and commonly used predictive analysis. The overall idea of regression is to examine two things: does a set of predictor variables do a good job in predicting an outcome variable? Is the model using the predictors accounting for the variability in the changes in the dependent variable? Which variables are significant predictors of the dependent variable? And in what way do they--indicated by the magnitude and sign of the beta estimates--impact the dependent variable? These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables.

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is an explanatory variable, and the other is a dependent variable. For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model.

In statistics, linear regression is a linear approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X . The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable. Before attempting to fit a linear model to observed data, a modeler should first determine whether there is a relationship between the variables of interest. This does not necessarily imply that one variable causes the other (for example, higher SAT scores do not cause higher college grades), but that there is some significant association between the two variables. A scatterplot can be a helpful tool in determining the strength of the relationship between two variables. If there appears to be no association between the proposed explanatory and dependent variables (i.e., the scatterplot does not indicate any increasing or decreasing trends), then fitting a linear regression model to the data probably will not provide a useful model. A valuable numerical measure of association between two variables is the correlation coefficient, which is a value between -1 and 1 indicating the strength of the association of the observed data for the two variables.

A linear regression line has an equation of the form $Y = a + bX$, where X is the explanatory variable and Y is the dependent variable. The slope of the line is b , and a is the intercept (the value of y when $x = 0$).

Simpsons 3/8

The main objective of this chapter is to develop appropriate formulas for approximating the integral of the form. In numerical analysis, Simpson's rule is a method for numerical integration, the numerical approximation of definite integrals. The numerical integration technique known as "Simpson's 3/8 rule" is credited to the mathematician Thomas Simpson (1710-1761) of Leicestershire, England. His also worked in the areas of numerical interpolation and probability theory.

Theorem ([Simpson's 3/8 Rule](#)) Consider $y = f(x)$ over $[x_0, x_3]$, where $x_1 = x_0 + h$, $x_2 = x_0 + 2h$, and $x_3 = x_0 + 3h$. Simpson's 3/8 rule is

$$SE(f, h) = \frac{3h}{8} (f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3))$$

This is an numerical approximation to the integral of $f(x)$ over $[x_0, x_3]$ and we have the expression

$$\int_{x_0}^{x_3} f(x) dx \approx SE(f, h)$$

The remainder term for Simpson's 3/8 rule is $R_{SE}(f, h) = -\frac{3}{80} f^{(4)}(c) h^5$, where c lies somewhere between x_0 and x_3 , and have the equality

$$\int_{x_0}^{x_3} f(x) dx = \frac{3h}{8} (f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)) - \frac{3}{80} f^{(4)}(c) h^5$$

Composite Simpson's 3/8 Rule

Our next method of finding the area under a curve is by approximating that curve with a series of cubic segments that lie above the intervals. When several Cubic's are used, we call it the composite Simpson's 3/8 rule.

Theorem (Composite Simpson's 3/8 Rule) Consider $y = f(x)$ over $[a, b]$. Suppose that the interval $[a, b]$ is subdivided into $3m$ subintervals $\{[x_{k-1}, x_k]\}_{k=1}^{3m}$ of equal width $h = \frac{b-a}{3m}$ by using the equally spaced sample points $x_k = x_0 + kh$ for $k = 0, 1, 2, \dots, 3m$. The composite Simpson's 3/8 rule for $3m$ subintervals is

$$SC(f, h) = \frac{3h}{8} \sum_{k=1}^m (f(x_{3k-3}) + 3f(x_{3k-2}) + 3f(x_{3k-1}) + f(x_{3k}))$$

Remainder term for the composite Simpson's 3/8 Rule

Corollary (Simpson's 3/8 Rule: Remainder term) Suppose that $[a, b]$ is subdivided into $3m$ subintervals $\{[x_{k-1}, x_k]\}_{k=1}^{3m}$ of width $h = \frac{b-a}{3m}$. The composite Simpson's 3/8 rule

$$SC(f, h) = \frac{3h}{8} \sum_{k=1}^m (f(x_{3k-2}) + 3f(x_{3k-1}) + 3f(x_{3k}) + f(x_{3k+1}))$$

Newton's Interpolation

In the mathematical field of numerical analysis, a Newton polynomial, named after its inventor Isaac Newton, is the interpolation polynomial for a given set of data points in the Newton form. The Newton polynomial is sometimes called Newton's divided differences interpolation polynomial because the coefficients of the polynomial are calculated using divided differences. As with other difference formulas, the degree of a Newton's interpolating polynomial can be increased by adding more terms and points without discarding existing ones. Newton's form has the simplicity that the new points are always added at one end: Newton's forward formula can add new points to the right, and Newton's backwards formula can add new points to the left.

The accuracy of polynomial interpolation depends on how close the interpolated point is to the middle of the x values of the set of points used. Obviously, as new points are added at one end, that middle becomes farther and farther from the first data point. Therefore, if it isn't known how many points will be needed for the desired accuracy, the middle of the x -values might be far from where the interpolation is done.

Centered Finite Method

In mathematics, finite-difference methods (FDM) are numerical methods for solving differential equations by approximating them with difference equations, in which finite differences approximate the derivatives. FDMs are thus discretization methods. The finite volume method is the most natural discretization scheme, because it makes use of the conservation laws in integral form. It subdivides the domain into cells and evaluates the field equations in integral form on these cells. In the area of TCAD this method is often called the finite box method to express the geometrical origin of the discretized domain.

Parts of the program:

I. Numerical Method selection for MP 4 (main window)

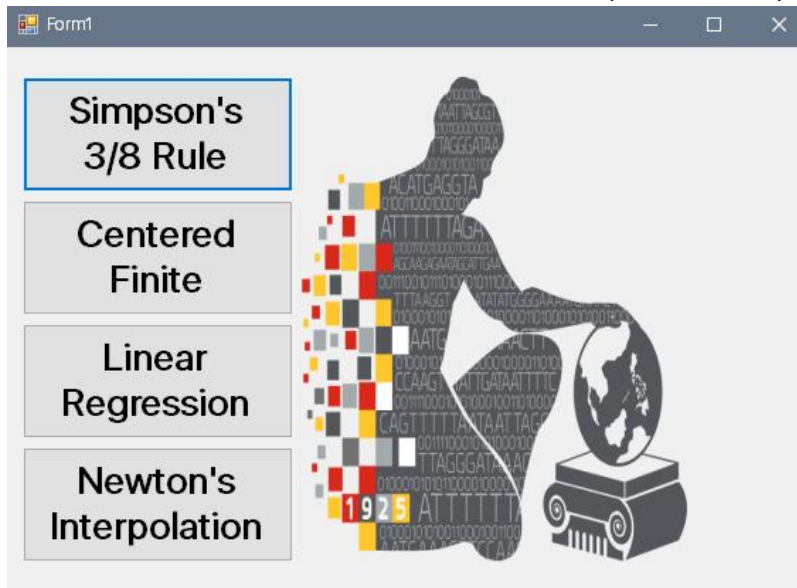


Figure 1. Main Window Selection

This depicts the available numerical methods, for machine problem 4.

II. Simpson's 3/8 Rule Program

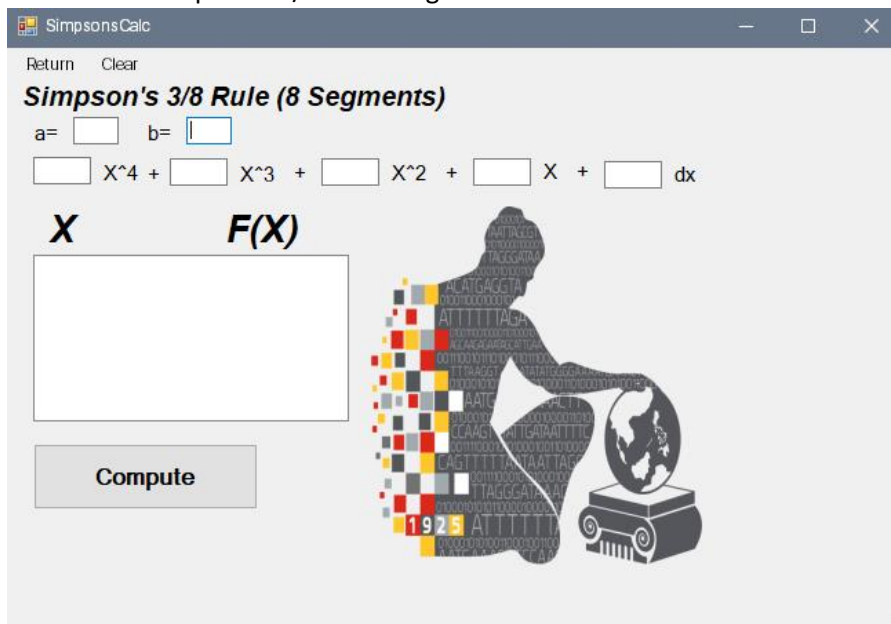


Figure 2. Window for Simpson's 3/8 Rule

III. Newton's Interpolation Program

NewtonInterpolationCalc

Return Clear

Newton's Interpolation

X F(x)

Enter X:

COMPUTE




Figure 3. Window for Newton's Interpolation

IV. Linear Regression Program

LinearRegressionCalc

Return Clear

Linear Regression

X F(x)

COMPUTE




Figure 4. Window for Linear Regression.

V. Centered Finite Divided Difference

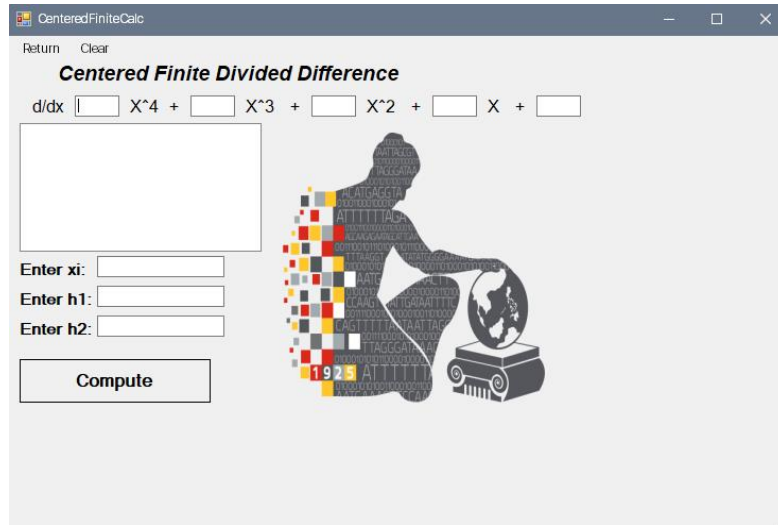
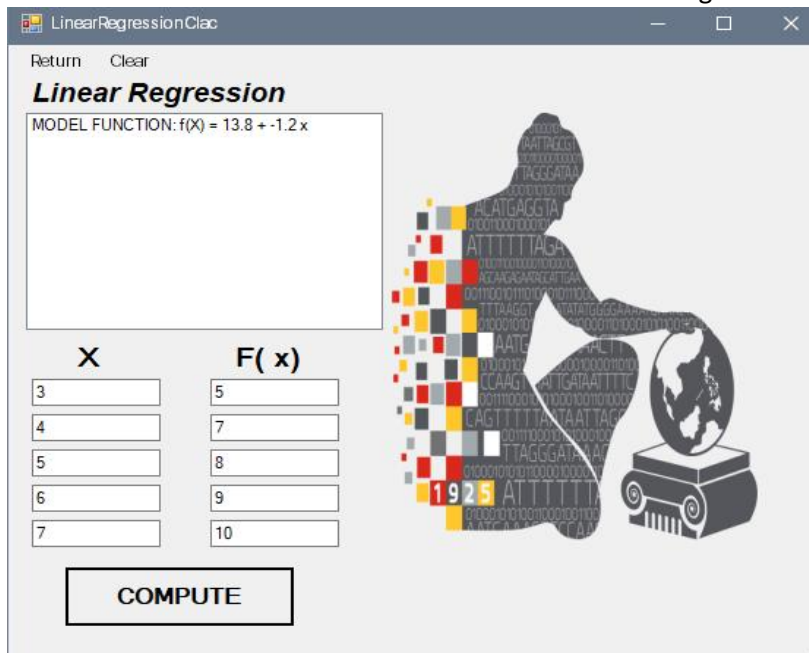


Figure 5. Window for Centered Finite Divided Difference

Steps to use the program:

Linear Regression

1. Input initial values of x and its corresponding value for $f(x)$.
2. Click Compute.
3. You will be able to see the model function for the given values of x and $f(x)$.



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

Newton's Interpolation

1. Input initial values of x and its corresponding value for f(x).
2. Input desired value of x to be estimated.
3. Click Compute.
4. Estimated value of f(x) will show.

3	6	∞	-∞	∞
3	8	0.75	-0.875	-0.4375
7	11	-4.5	-1.75	
9	2	-1		
5	6			

f(2) = -∞

3 6
3 8
7 11
9 2
5 6

X F(x)

Enter X: 2

COMPUTE

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

Simpson's 3/8 Rule

1. Input values for a and b which are the limits of the function.
2. Input Coefficients of your function.
3. Click Compute
4. Then it will show the values for x, its corresponding f(x), and the estimated value of integration.

SimpsonsCalc

Return Clear


Simpson's 3/8 Rule (8 Segments)

a= b=

X^4 + X^3 + X^2 + X + dx

X	F(X)
3	421
3.25	540
3.5	684
3.75	855
4	1056
4.25	1291
4.5	1564
4.75	1879
5	2239

Compute



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

Centered Finite Divided Difference

1. Input coefficient of your given function
2. Input the value of initial x.
3. Input the desired step sizes.
4. Click Compute.
5. The derivative estimation will be displayed.

CenteredFiniteCalc

Return Clear

Centered Finite Divided Difference

d/dx X^4 + X^3 + X^2 + X +

CENTERED FINITE DIVIDED DIFFERENCE
 f(x) = 499
 f'(h1) = 1588
 f'(h2) = 2596


 D(a) = 1252

Enter xi:

Enter h1:

Enter h2:

Compute



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

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 MachineProblem4
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```
    }

    private void button1_Click(object sender,
EventArgs e)
    {
        MachineProblem4.SimpsonsCalc form =
new MachineProblem4.SimpsonsCalc();

        form.ShowDialog();
    }

    private void button3_Click(object sender,
EventArgs e)
    {
        MachineProblem4.CenteredFiniteCalc
form = new
MachineProblem4.CenteredFiniteCalc();

        form.ShowDialog();
    }

    private void button2_Click(object sender,
EventArgs e)
```

```

    {
        MachineProblem4.LinearRegressionClac
form = new
MachineProblem4.LinearRegressionClac();

        form.ShowDialog();
    }

    private void button4_Click(object sender,
EventArgs e)
    {

MachineProblem4.NewtonInterpolationCalc
form = new
MachineProblem4.NewtonInterpolationCalc();

        form.ShowDialog();
    }
}

```

SimpsonsCalc.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 MachineProblem4
{
    public partial class SimpsonsCalc : Form
    {
        public SimpsonsCalc()
        {
            InitializeComponent();

            private void button1_Click(object sender,
EventArgs e)
            {
                try
                {
                    double b, a, h;

                    b = double.Parse(txtb.Text);

                    a = double.Parse(txta.Text);

                    if (b <= a)
                    {
                        MessageBox.Show("Please enter
larger value for the upper limit b");

                        txtb.Text = "";
                    }
                    else
                    {

                        double[] x;

                        x = new double[9];

```

```

double y = 0;

y = fx(a);

listBox1.Items.Add(a + "\t\t" +
y.ToString() + "\t\t" + y.ToString());

h = (b - a) / 8;

double fx11 = 0;

double trule = 0;

for (int i = 1; i < 8; i++)
{
    a += h; x[i] = fx(a);

    if (i % 3 == 0) { trule = 2 * x[i]; }

    else { trule = 3 * x[i]; }

    fx11 += trule;

    listBox1.Items.Add(a + "\t\t" +
x[i].ToString() + "\t\t" + trule.ToString());

}

double k = 0;

k = fx(b);

label13.Visible = true;

label15.Visible = true;

label16.Visible = true;

listBox1.Items.Add(b + "\t\t" + k +
"\t\t" + k);

double Efx = 0; Efx = fx11 + k + y;

double ansl = 0; ansl = (3 * h / 8) *
Efx;

listBox1.Items.Add("=====
=====
=====");

```

```

listBox1.Items.Add("\t\t" + " +
Math.Round(Efx, 4));

}

}

catch (SystemException exa) {
MessageBox.Show(exa.Message);

}

}

private double fx(double x)
{
    float x2, x1, x3, x4, C = 0;

    x2 = float.Parse(txtx2.Text);

    x1 = float.Parse(txtx1.Text);

    x3 = float.Parse(txtx3.Text);

    x4 = float.Parse(txtx4.Text);

    C = float.Parse(txtC.Text);

    x = Math.Round((x4 * Math.Pow(x, 4)) +
(x3 * Math.Pow(x, 3)) + (x2 * Math.Pow(x, 2)) +
(x1 * Math.Pow(x, 1)) + C);

    return x;

}

private void
clearToolStripMenuItem_Click(object sender,
EventArgs e)

{
    txta.Text = "";

    txtb.Text = "";

    txtC.Text = "";

    txtx1.Text = "";

    txtx2.Text = "";

```

```

        txtx3.Text = "";

        txtx4.Text = "";

        listBox1.Items.Clear();

        label13.Visible = false;

        label15.Visible = false;

        label16.Visible = false;
    }

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

```

CenteredFiniteCalc.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 MachineProblem4
{
    public partial class CenteredFiniteCalc : Form
    {
        public CenteredFiniteCalc()
        {
            InitializeComponent();
        }

        private double fx(double x)
        {
            float x2, x1, x3, x4, C = 0;

            x2 = float.Parse(txtx2.Text);

            x1 = float.Parse(txtx1.Text);

            x3 = float.Parse(txtx3.Text);

            x4 = float.Parse(txtx4.Text);

            C = float.Parse(txtC.Text);

            x = Math.Round((x4 * Math.Pow(x, 4)) +
(x3 * Math.Pow(x, 3)) + (x2 * Math.Pow(x, 2)) +
(x1 * Math.Pow(x, 1)) + C);

            return x;
        }

        private void
clearToolStripMenuItem_Click(object sender,
EventArgs e)
        {

```

```

        txth1.Text = "";
        txth2.Text = "";
        txtxi.Text = "";
        txtC.Text = "";
        txtx1.Text = "";
        txtx2.Text = "";
        txtx3.Text = "";
        txtx4.Text = "";
        listBox1.Items.Clear();
    }

    private void button1_Click(object sender,
    EventArgs e)
    {
        double xi, h1, h2, low, big, fxi1, fxi2,
        fpxi1, xip1, fpxi2, xip2, fxi2, xim1, xim2,
        dxim1, fxim2, fxim1, D;

        xi = double.Parse(txtxi.Text);
        h1 = double.Parse(txth1.Text);
        h2 = double.Parse(txth2.Text);
        if (h1 < h2)
        {
            low = h1; big = h2; //1
            fxi1 = fx(xi); xip1 = xi + (1 * low);
            xim1 = xi - (1 * low);
            fxi2 = fx(xip1);
            fxim1 = fx(xim1);
            fpxi1 = (fxip1 - fxim1) / (2 * low);
            //2
            fxi2 = fx(xi); xip2 = xi + (1 * big);

```

```

            xim2 = xi - (1 * big);
            fxi2 = fx(xip2);
            fxim2 = fx(xim2); fpxi2 = (fxip2 - fxim2)
            / (2 * big);
            D = fpxi1 + ((1 / (Math.Pow(2, 2) - 1)) *
            (fpxi1 - fpxi2));
            listBox1.Items.Add("CENTERED FINITE
            DIVIDED DIFFERENCE");
            listBox1.Items.Add("f(xi) = " +
            Math.Round(fxi2, 4));
            listBox1.Items.Add("f'(h1) = " +
            Math.Round(fpxi1, 4));
            listBox1.Items.Add("f'(h2) = " +
            Math.Round(fpxi2, 4));
            listBox1.Items.Add("");
            listBox1.Items.Add("=====
            =====
            =====");
            listBox1.Items.Add("D(a) = " +
            Math.Round(D, 4));
        }
        else
        {
            low = h2;
            big = h1;
            fxi1 = fx(xi);
            xip1 = xi + (1 * low);
            xim1 = xi - (1 * low);
            fxi2 = fx(xip1);
            fxim1 = fx(xim1);
            fpxi1 = (fxip1 - fxim1) / (2 * low);

```

```

//2

fxi2 = fx(xi); xip2 = xi + (1 * big);

xim2 = xi - (1 * big);

fxip2 = fx(xip2);

fxim2 = fx(xim2);

fpxi2 = (fxip2 - fxim2) / (2 * big);

D = fpxi1 + ((1 / (Math.Pow(2, 2) - 1)) *
(fpxi1 - fpxi2));

listBox1.Items.Add("CENTERED FINITE
DIVIDED DIFFERENCE");

listBox1.Items.Add("f(xi) = " +
Math.Round(fxi2, 4));

listBox1.Items.Add("f'(h1) = " +
Math.Round(fpxi2, 4));

listBox1.Items.Add("f'(h2) = " +
Math.Round(fpxi1, 4));

listBox1.Items.Add("");

listBox1.Items.Add("=====
=====
=====");

listBox1.Items.Add("D(a) = " +
Math.Round(D, 4));
}
}

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

```

```

}

```

LinearRegressionCalc.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 MachineProblem4
{
    public partial class LinearRegressionClac :
    Form
    {
        public LinearRegressionClac()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender,
        EventArgs e)
        {
            double x1, x2, x3, x4, x5, fx1, fx2, fx3, fx4,
            fx5 = 0;

            try
            {

```

```

x1 = double.Parse(txtx1.Text);

x2 = double.Parse(txtx2.Text);

x3 = double.Parse(txtx3.Text);

x4 = double.Parse(txtx4.Text);

x5 = double.Parse(txtx5.Text);

fx1 = double.Parse(txtfx1.Text);

fx2 = double.Parse(txtfx2.Text);

fx3 = double.Parse(txtfx3.Text);

fx4 = double.Parse(txtfx4.Text);

fx5 = double.Parse(txtfx5.Text);

//summation of xi^2

double sxi2 = 0; sxi2 = ((x1 * x1) + (x2 *
x2) + (x3 * x3) + (x4 * x4) + (x5 * x5));

//summation of xi

double sxi; sxi = x1 + x2 + x3 + x4 + x5;

//summation of fxi

double fxi; fxi = fx1 + fx2 + fx3 + fx4 +
fx5;

//summation of xiyi

double sxy; sxy = (x1 + fx1) + (x2 + fx2)
+ (x3 * fx3) + (x4 * fx4) + (x5 * fx5);

double a1; a1 = Math.Round((((5 * sxy)
- (sxi * fxi)) / ((5 * sxi2) - (sxi * sxi))), 4);

double x, y = 0; x = sxi / 5; y = fxi / 5;

double a0; a0 = Math.Round((y - (x *
a1)), 4);

string answer; answer = "MODEL
FUNCTION: f(X) = " + a0 + " + " + a1 + " x";
listBox1.Items.Add(answer);

}

catch (SystemException exa) {
MessageBox.Show(exa.Message); }

```

```

}

private void
clearToolStripMenuItem_Click(object sender,
EventArgs e)
{
txtx1.Text = "";
txtfx1.Text = "";
txtx2.Text = "";
txtfx2.Text = "";
txtx3.Text = "";
txtfx3.Text = "";
txtx4.Text = "";
txtfx4.Text = "";
txtx5.Text = "";
txtfx5.Text = "";

listBox1.Items.Clear();
}

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

```

NewtonInterpolationCalc.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 MachineProblem4
{
    public partial class NewtonInterpolationCalc :
    Form
    {
        public NewtonInterpolationCalc()
        {
            InitializeComponent();

            private void button1_Click(object sender,
            EventArgs e)
            {
                double x1, x2, x3, x4, x5, fx1, fx2, fx3, fx4,
                fx5 = 0;

                try
                {
                    x1 = double.Parse(txtx1.Text);
                    x2 = double.Parse(txtx2.Text);
                    x3 = double.Parse(txtx3.Text);
                    x4 = double.Parse(txtx4.Text);
                    x5 = double.Parse(txtx5.Text);

```

```

                    fx1 = double.Parse(txtfx1.Text);
                    fx2 = double.Parse(txtfx2.Text);
                    fx3 = double.Parse(txtfx3.Text);
                    fx4 = double.Parse(txtfx4.Text);
                    fx5 = double.Parse(txtfx5.Text);

                    double f1x1, f1x2, f1x3, f1x4, f2x1,
                    f2x2, f2x3, f3x1, f3x2, f4x1, b0, b1, b2, b3, b4 =
                    0;

                    f1x1 = Math.Round((fx2 - fx1) / (x2 -
                    x1), 4);
                    f1x2 = Math.Round((fx3 - fx2) / (x3 -
                    x2), 4);
                    f1x3 = Math.Round((fx4 - fx3) / (x4 -
                    x3), 4);
                    f1x4 = Math.Round((fx5 - fx4) / (x5 -
                    x4), 4);
                    f2x1 = Math.Round((f1x2 - f1x1) / (x3 -
                    x1), 4);
                    f2x2 = Math.Round((f1x3 - f1x2) / (x4 -
                    x2), 4);
                    f2x3 = Math.Round((f1x4 - f1x3) / (x5 -
                    x3), 4);
                    f3x1 = Math.Round((f2x2 - f2x1) / (x4 -
                    x1), 4);
                    f3x2 = Math.Round((f2x3 - f2x2) / (x5 -
                    x1), 4);
                    f4x1 = Math.Round((f3x2 - f3x1) / (x5 -
                    x1), 4);

                    b0 = fx1; b1 = f1x1; b2 = f2x1; b3 =
                    f3x1;

                    b4 = f4x1;

                    double x = double.Parse(txtX.Text);

                    double answer;

```

```

        /* label5.Visible = true;

        label6.Visible = true;

        label7.Visible = true;

        label8.Visible = true;

        label9.Visible = true;

        label10.Visible = true;*/

        listBox1.Items.Add(x1 + "\t" + fx1 + "\t"
+ f1x1 + "\t" + f2x1 + "\t" + f3x1 + "\t" + f4x1);

        listBox1.Items.Add(x2 + "\t" + fx2 + "\t"
+ f1x2 + "\t" + f2x2 + "\t" + f3x2);

        listBox1.Items.Add(x3 + "\t" + fx3 + "\t"
+ f1x3 + "\t" + f2x3);

        listBox1.Items.Add(x4 + "\t" + fx4 + "\t"
+ f1x4);

        listBox1.Items.Add(x5 + "\t" + fx5);

        answer = Math.Round((b0 + (b1 * (x -
x1)) + (b2 * (x - x1) * (x - x2)) + (b3 * (x - x1) * (x
- x2) * (x - x3)) + (b4 * (x - x1) * (x - x2) * (x - x3)
* (x - x4))), 4);

        listBox1.Items.Add("");

listBox1.Items.Add("=====
=====
=====");

        listBox1.Items.Add("f(" + x.ToString() +
") =" + answer.ToString());

    }

    catch (SystemException exa) {
        MessageBox.Show(exa.Message);
    }

    private void
clearToolStripMenuItem_Click(object sender,
EventArgs e)

```

```

    {
        txtx1.Text = "";

        txtfx1.Text = "";

        txtx2.Text = "";

        txtfx2.Text = "";

        txtx3.Text = "";

        txtfx3.Text = "";

        txtx4.Text = "";

        txtfx4.Text = "";

        txtx5.Text = "";

        txtfx5.Text = "";

        listBox1.Items.Clear();

        txtX.Text = "";

        /* label5.Visible = false;

        label6.Visible = false;

        label7.Visible = false;

        label8.Visible = false;

        label9.Visible = false;

        label10.Visible = false;*/

    }

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

    }

}

```

References

<http://www.statisticssolutions.com/what-is-linear-regression/>

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

http://mathforcollege.com/nm/mws/gen/07int/mws_gen_int_txt_simpson3by8.pdf

https://en.wikipedia.org/wiki/Simpson%27s_rule

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

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

