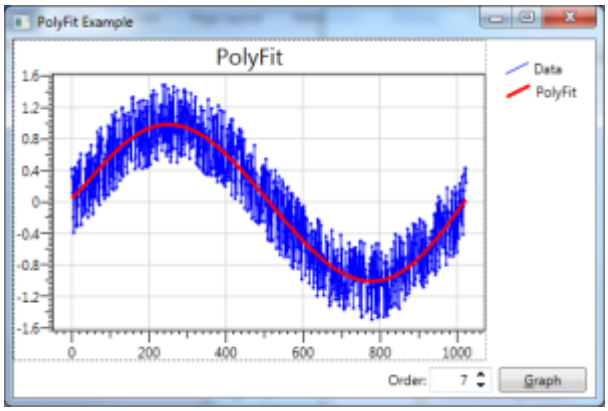
Vili Petek

Senior Software Engineer

Polynomial Fitting in C#

December 13, 2014 by vilipetek

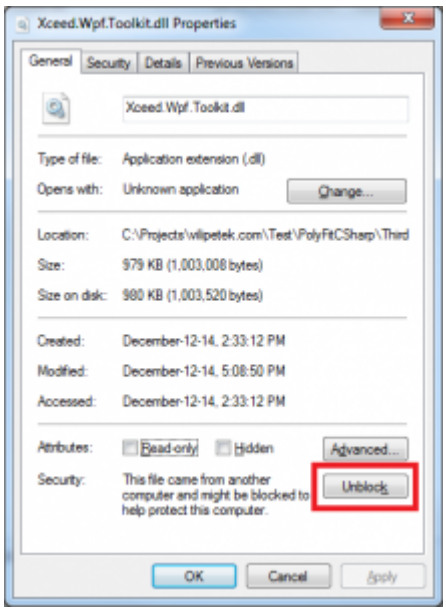


This is a third post in the polynomial fitting series demonstrating how to perform polynomial fitting using LU decomposition in C#. For more information on polynomial fitting please refer to the pervious post polynomial fitting in C++.

Full source code can be downloaded below. Please note that the code is not highly optimized for readability purposes.

Download Example
Download Source Code

Once downloaded you will need to navigate to the folder '*PolyFitCSharp\ThirdPartGuiLibraries*' and unblock *DynamicDataDisplay.dll* and *Xceed.Wpf.Toolkit.dll* by right clicking each file and selecting *Properties* from the menu then clicking *Unblock* button as shown in the image below.



The polynomial fitting code is located in its own project called *PolynomialRegression* under the *PolyFitExample* solution. Below is code listing for the example usage.

```
namespace DemoApplication
{
    class Program
    {
        const int DataSize = 1024;
        static void Main(string[] args)
        {

            double[] x = new double[DataSize];
            double[] y = new double[DataSize];

            // generate the data
            Random rand = new Random();
            for (int i = 0; i < DataSize; i++)
            {
                x[i] = i;
                y[i] = Math.Sin((double)i / 1024.0 * Math.PI * 2) + (rand.I
            }

            // fit the data
            var polyfit = new PolyFit(x, y, 3);
            var fitted = polyfit.Fit(x);
        }
    }
}
```

Main class used for polynomial fitting is *PolyFit*. It relies on further internal classes but their listing is beyond the scope of this post.

```
public class PolyFit
{
    /// <summary>
    /// Coefficients of a polynomial starting at the constant coefficient
    /// and ending with the coefficient of power to the chosen order.
    /// </summary>
    public double[] Coeff { get; private set; }

    /// <summary>
    /// Finds the coefficients of a polynomial p(x) of degree n that fits
```

```
11      /// p(x(i)) to y(i), in a least squares sense. The result p is a row v
12      /// length n+1 containing the polynomial coefficients in incremental p
13      /// </summary>
14      /// <param name="x">x axis values</param>
15      /// <param name="y">y axis values</param>
16      /// <param name="order">polynomial order including the constant</param
17      public PolyFit(double[] x, double[] y, int order)
18      {
19          // incrememnt the order to match matlab way
20          double[,] matrixX = new double[x.Count(), ++order];
21          double[,] matrixY = new double[x.Count(), 1];
22
23          if (x.Length != y.Length)
24          {
25              throw new ArgumentException("x and y array lengths do not matc
26          }
27
28          // copy y matrix
29          for (int i = 0; i < y.Count(); i++)
30          {
31              matrixY[i, 0] = y[i];
32          }
33
34          // create the X matrix
35          for (int row = 0; row < x.Count(); row++)
36          {
37              double nVal = 1.0f;
38              for (int col = 0; col < order; col++)
39              {
40                  matrixX[row, col] = nVal;
41                  nVal *= x[row];
42              }
43          }
44
45          var matrixXt = matrixX.Transpose();
46          var matrixXtX = matrixXt.Product(matrixX);
47          var matrixXtY = matrixXt.Product(matrixY);
48
49          var lu = new LUDecomposition(matrixXtX);
50          Coeff = lu.Solve(matrixXtY).GetColumn(0).ToArray();
51      }
52
53      /// <summary>
54      /// Calculates the value of a polynomial of degree n evaluated at x. T
55      /// pCoeff is a vector of length n+1 whose elements are the coefficien
56      /// powers of the polynomial to be evaluated.
57      /// </summary>
58      /// <param name="x">Array of x values</param>
59      /// <returns>Array of fitted y values</returns>
60      public double[] Fit(double[] x)
61      {
62          double[] y = new double[x.Length];
63          int pos = 0;
64
65          foreach (double xval in x)
66          {
67              double xcoeff = 1;
68              foreach (double coeffval in Coeff)
69              {
70                  // multiply current x by a coefficient
71                  y[pos] += coeffval * xcoeff;
72                  // power up the X
73                  xcoeff *= xval;
74              }
75              pos++;
76          }
77
78          return y;
79      }
```

```
80 }
```

C Sharp, Math

WPF Window Size, Position, and Style Persistence
Binary Perceptron in C#

Search ...

Recent Posts

Categories