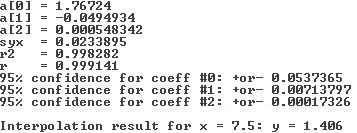
Program #2 Polynomial Regression and Interpolation **11.25%**  
Due: 10/14/2016  
Assignment:

In the following figure, I have solved Problem 20-48 on page 577 using polynomial regression and polynomial interpolation.



In this program, you will write code to perform polynomial regression and polynomial interpolation of an arbitrary, user specified degree. Refer to page **470** for a discussion of polynomial regression. Refer to page **489** for a discussion of polynomial interpolation using Newton's divided difference interpolating polynomials.

**General Matrix Form for Polynomial Regression**

In this part of the assignment, you are assuming that the discrete data values stored reflect a general functional trend, but some error is present in each data point. The least squares fit represents an attempt to find the underlying function that the data represents. **With regression, you model the entire data set at once**.

Expand upon the discussion on page 471 to allow a regression polynomial to be computed for an arbitrary degree. Use the general matrix formulation discussed on page 477.

Report the final best fit parameters, syx, and r2. Also, report the 95% confidence intervals for each parameter.

Use the following public interface to perform your regression and to report the results (**PolyRegression.c**):

* double\* regression(int degree) //returns an array containing the best fit coefficients
* double syx(Poly\* poly) //polynomial was constructed using the coefficients from regression method
* double r2(Poly\* poly)
* double\* confidence(Poly\* poly)

where the array returned from the regression method contains the coefficients of the polynomial of the specified degree corresponding to the best least squares fit to the data passed to the method.

**Divided Difference Interpolating Polynomials**

In this part of the assignment, an array of data points stores discrete values that are known to be very accurate, but the underlying function is still unknown. Therefore, if you need to find values not specifically stored in the array, you must interpolate between the known values by fitting them to a **local** interpolating polynomial, using the known values **closest** to the desired value to perform the interpolation. **You will model a small segment of the data set, the size of which depends on the polynomial that you are fitting to the data**.

* Use **recursion** to compute the divided differences.
* Use the points closest to the desired value to determine the interpolating polynomial.

Use the following public interface to compute an unknown value using a polymonial of a specified degree to model unknown points (**PolyInterpolation.cpp**):

* double interpolateNewton(int degree, double x)

where the number returned from the interpolate method is the interpolated value.

**Use your program to solve Problem 20-22 on page 573**.

* [**t-table.pdf**](http://mboshart.dyndns.org/~mboshart/3020programs/assign2/t-table.pdf)
* [**Prob20-48.txt**](http://mboshart.dyndns.org/~mboshart/3020programs/assign2/Prob20-48.txt)
* C++ files:
  + [**prog2.zip**](http://mboshart.dyndns.org/~mboshart/3020programs/prog2.zip)