Interpolation Tutorial

CS 310 — Fall 1999

In this lab we will experiment with three Matlab functions that can be used to generate interpolating functions. These functions are `polyfit`, `polyval`, and `spline`. First we describe how each of these functions is used.

1 polyfit and polyval

`coef = polyfit(x,y,n)`

This finds the coefficients of a polynomial of degree `n` that fits the data (`x`,`y`) as close as possible. If `n+1` is equal to the number of points, then the graph of the polynomial passes through all of the points. For example,

```
x = [1 2 3];
y = [4 8 1];
coef = polyfit(x,y,2)
```

will return `coef = [-5.5, 20.5, -11]`. This means that the parabola that goes through the points $(1, 4), (2, 8)$, and $(3, 1)$ is $-5.5x^2 + 20.5x - 11$. Since we had 3 points, a degree 2 polynomial fits the points exactly.

`y = polyval(coef,x)`

This is really just another way of evaluating polynomial functions. For example, suppose `coef` has been defined as in the above example. If we enter:

```
x = 0:.1:10;
y = polyval(coef, x);
```

Then the vector `y` contains the values of $-5.5x^2 + 20.5x - 11$ for each of the values in `x`. This means that the two commands

```
z = polyval(coef, x);
z = -5.5*x.^2 + 20.5*x - 11;
```

result in the same vector `z`..

Example

Find the unique polynomial that goes through the points $(-1, 2), (2, 4), (6, 0)$ and $(11, -3)$. Plot the points and the polynomial from $x = -5$ to $x = 15$.

Remember that through n points, there is a unique polynomial of degree $n - 1$. So, through these 4 points, we can get a unique polynomial of degree 3. The following Matlab commands will solve this problem:

```
x = [-1 2 6 11]
y = [2 4 0 -3]
coef = polyfit(x,y,3)
xx = -5:.1:15;
yy = polyval(coef, xx);
plot(x,y,'o',xx,yy);
```

See if you can solve the following problem:

☞ Find the unique polynomial that goes through the points $(-9, -5)$, $(1, 11)$, $(2, -11)$, $(3, 8)$ and $(9, 0)$. Plot the polynomial and the points from $x = -10$ to $x = 10$.

Compare the coefficients for this polynomial with those obtained in the example. Why are they much larger?

☞ What happens if you try to fit a polynomial of degree 5 through the 5 points? Why?

2 spline

`yi = spline(x,y,xi)`

This finds the cubic spline that goes through the points (`x`, `y`), and evaluates it at the points `xi`. For each element of `xi`, the corresponding value of the cubic spline is stored in the vector `yi`.

For example:

```
x = [1 2 3 4 5];
y = [4 6 1 0 2];
xi = 1:.01:5;
yi = spline(x,y,xi);
plot(x,y,xi,yi);
```

In this example, the data points are $(1, 4), (2, 6), (3, 1), (4, 0)$, and $(5, 2)$. Matlab calculates the equations of the cubic spline, and then evaluates the spline at each element of `xi` to get `yi`.

The `plot` statement then plots `y` against `x`, connecting each point with a line segment (i.e., using linear interpolation), and `yi` against `xi`, connecting each of these with a line segment as well, but since the elements of `xi` are so narrowly spaced, the second plot looks like a smooth curve.

☞ For the same points you used above, $(-9, -5), (1, 11), (2, -11), (3, 8)$ and $(9, 0)$, plot the cubic spline on the same graph as the polynomial you got before. Put the points on the graph as well.

Which one looks like a better interpolating function: the spline or the polynomial?

☞ Copy the files `interp0.m`, `interp1.m`, and `interp2.m` from the the course Web page to your matlab directory.

3 interp0.m

Run the Matlab script `interp0.m`. It is a good idea to have an editor open with the file `interp0.m` to refer to as the program runs so you can see how everything is being done.

The program uses interpolation functions of various degrees between successive points. Remember that through k points, we can get a polynomial of degree $k-1$. So, to get a polynomial of degree 1 (a line), we need 2 points. Through 3 points, we can get a polynomial of degree 2.

For each polynomial, say, of degree d, groups of $d+1$ points are used to generate polynomials of degree d, and each is plotted one at a time.

Low-degree polynomials do a good job of interpolating between the points that determine the polynomial. At higher degrees, this is not always the case. In this example, though, the unique polynomial of degree 6 through the 7 points is a good interpolant.

The Matlab functions `polyfit` and `polyval` are being used to generate these interpolating polynomials.

4 interp1.m

Run the Matlab script `interp1.m`. It is a good idea to have an editor open with the file `interp1.m` to refer to as the program runs so you can see how everything is being done.

Look at some of the choices for the low-degree polynomials, and be sure to look at the polynomial of degree 11 that interpolates these 12 data points. Notice that it is a poor interpolant, especially near the ends of the interval.

The cubic spline does a very good job of interpolating. To calculate the spline, the Matlab function `spline` is used.

5 interp2.m

This time, you get to modify the data. When you run `interp2.m`, the program asks for a value of perturbation to apply to the data. A small random number between 0 and the perturbation you enter is added to each of the `y` data values. A different random number is used for each data value.

With no perturbation (i.e., 0) the interpolation with a polynomial of degree 12 is quite good. With a small perturbation of the data (e.g., 0.3), the polynomial does a very poor job of interpolating. So, small changes in the data can produce very large changes in the interpolating polynomial. However, the cubic spline does a good job in both cases, giving very similar results.

☞ For some smooth functions, the polynomial interpolant can be quite bad. In a comment near the beginning of the file is a function, called Runge's function, for which the interpolation by high-degree polynomials is quite bad. Change the program to use this function to calculate the `y` data, then re-run the program. The cubic spline should do a good job with Runge's function.

☞ Splines are often used in computer graphics to represent continuous curves with just a few points. In this problem, you are given 20 points along the top side of a profile of a jet airplane.

- Use `spline` to generate a graph of the top side of the plane. The points are:

(1,2)	(45,12)	(74,25)	(180,20)
(2,3)	(47,13)	(105,24)	(190,24)
(4,4)	(51,15)	(125,23)	(204,33)
(11,6)	(58,20)	(145,21)	(215,39)
(25,9)	(63,23)	(170,19)	(228,40)

- You can save yourself some typing by copying the file `plane.dat` from the course Web page to your matlab directory. Then using these Matlab commands:

```
load plane.dat
x = plane(:,1)
y = plane(:,2)
```

  The first command loads the data into the matrix `plane`, and the second two commands extract the two columns of `plane` into vectors `x` and `y`.

- To make your plot look good, you should make sure that the axes are both the same. You can do this with the `axis` command:

```
axis([0 250 0 250])
```

  This forces the range of both x and y to be 0 to 250.

- Since there are 20 points, a degree 19 polynomial will exactly fit the points. What do you think will happen if you try this?