Cubic Splines and Matlab

In this section, we introduce the concept of the cubic spline, and how they are implemented in Matlab. Of particular importance are the new Matlab data structures that we will see.

Cubic Splines Defined

Definition: Given n data points, $(x_1, y_1), \ldots, (x_n, y_n)$, a cubic spline is a piecewise-defined function of the form:

$$\begin{array}{rll} S_1(x) & = y_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 & \text{for } x \in [x_1, x_2] \\ S_2(x) & = y_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3 & \text{for } x \in [x_2, x_3] \\ \vdots & \vdots & \\ S_{n-1}(x) & = y_{n-1} + b_{n-1}(x - x_{n-1}) + c_{n-1}(x - x_{n-1})^2 + d_{n-1}(x - x_{n-1})^3 & \text{for } x \in [x_{n-1}, x_n] \end{array}$$

Note the difference between this and in class- We can go ahead and substitute the y's in directly.

The $3n-3$ unknowns (b's, c's, d's) are chosen to satisfy the interpolation constraints and also some smoothness constraints:

- (Interpolation Conditions) We already have $S_i(x_i) = y_i$ for $i = 1$ to $n-1$. In addition, we want the overall function to be continuous. At the interior points, $i = 1, 2, \ldots, n-2$,

  $$S_i(x_{i+1}) = S_{i+1}(x_{i+1}) = y_{i+1}$$

  and at the far right value of x, $S_{n-1}(x_n) = y_n$

  Altogether, there are $n - 1$ constraints here.

- (Smoothness Condition 1) At the interior points,

  $$S_i{}'(x_{i+1}) = S_{i+1}{}'(x_{i+1})$$

  There are $n - 2$ constraints here.

- (Smoothness Condition 2) At the interior points,

  $$S_i{}''(x_{i+1}) = S_{i+1}{}''(x_{i+1})$$

  There are $n - 2$ constraints here.

So far, we have $(n-1)+(n-2)+(n-2) = 3n-5$ constraints, but we have $3(n-1) = 3n-3$ coefficients. We need two more constraints to define the cubics uniquely.

Using the Two Extra Constraints

We can have different cubic splines depending on how we want to use our two extra constraints. Here are some common ones:

- (The Natural Spline) ${S_1}''(x_1) = 0 = {S_{n-1}}''(x_n)$
- (The Clamped Spline) The user defines ${S_1}'(x_1)$ and ${S_{n-1}}'(x_n)$
- "Not-A-Knot": (Matlab Default)

  Third derivatives match at x_2 and x_{n-1}; That is, $d_1 = d_2$ and $d_{n-2} = d_{n-1}$, or

$$S'''_1(x_2) = S'''_2(x_2) \qquad S'''_{n-2}(x_{n-2}) = S'''_{n-1}(x_{n-2})$$

Derivatives and Integrals

We just note here the derivative and antiderivative of each piece of the cubic spline,

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

$${S_i}'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$$

and

$${S_i}''(x) = 2c_i + 6d_i(x - x_i)$$

The antiderivative (note the shift):

$$\int S_i(x)\,dx = y_i(x - x_i) + \frac{1}{2}b_i(x - x_i)^2 + \frac{1}{3}c_i(x - x_i)^3 + \frac{1}{4}d_i(x - x_i)^4$$

In particular, if we integrate over the interval $[x_i, x_{i+1}]$ and the data is evenly spaced with width h, then:

$$\int_{x_i}^{x_{i+1}} S_i(x)\,dx = y_i h + \frac{1}{2}b_i h^2 + \frac{1}{3}c_i h^3 + \frac{1}{4}d_i h^4$$

Matlab Data Structures

In Matlab, we can collect different types of data together into a single "structure". For example, suppose that you have a vector `x` and a matrix `A` and several constants, a, b, c that you are going to be using over and over again. To wrap them up into a structure, you could write:

```
Examp.x=[0;1;2];
Examp.A=[1 2;3 4;5 6];
Examp.constants=[1,2,3];
```

To access, for example, the $(3, 2)$ element you've stored in the matrix A, you would type: `Examp.A(3,2)`

Piecewise Polynomials

Matlab has built-in commands for dealing with piecewise-defined polynomials, like cubic splines. A piecewise-defined polynomial is defined in Matlab by a vector containing the breaks and a matrix defining the polynomial coefficients.

Caution: A vector of coefficients, like `[3,2,1]`, over an interval like `[2,3]` is interpreted by the piecewise polynomial routines as:

$$3(x-2)^2+2(x-2)+1 \qquad \text{NOT} \qquad 3x^2+2x+1$$

Example: Constructing a piecewise polynomial by hand:

Suppose we want to define the following:

$$P(x)=\begin{cases} x^2-3x+1 & \text{if } x\in[1,2) \\ 3x-5 & \text{if } x\in[2,3] \end{cases}$$

(NOTE: In the Matlab documentation, it is unclear which function is used to evaluate the interior breaks. This is the way it is done- The intervals are all assumed to be of the form $[a,b)$ except the last interval.)

First we need to rewrite these[1]:

$$x^2-3x+1=(x-1)^2-(x-1)-1$$

and

$$3x-5=3(x-2)+1$$

In Matlab, make the piecewise polynomial and plot the results:

```
breaks=[1,2,3];
coefs=[1,-1,-1;0,3,1];
pp=mkpp(breaks,coefs);
x=linspace(1,3);
y=ppval(pp,x);
plot(x,y);
```

In Matlab, there is a command `polyval` that can be used to evaluate a polynomial. For example, the command:

```
y=polyval([1,2,3,4],x,[],[3,5]);
```

will evaluate the polynomial:

$$\frac{(x-3)^3}{5}+2\frac{(x-3)^2}{5}+3\frac{(x-3)}{5}+4$$

[1]Is there an easy way to do this? Use a Taylor expansion based at the left endpoint.

1 Implementing Splines In Matlab

The relevent commands are:

`spline` `ppval` `mkpp`

(Also see the help page for any of these commands).

The relevent command is:

```
pp = spline(x,y)
```

Builds the cubic spline using the data in x, y, and outputs the result as a piecewise polynomial, `pp`. The structure `pp` will contain several pieces of data. Before continuing, let's try one:

```
x=0:10;
y=sin(x);

A=spline(x,y);
```

If you look at A, you will see:

```
A =
      form: 'pp'
    breaks: [0 1 2 3 4 5 6 7 8 9 10]
     coefs: [10x4 double]
    pieces: 10
     order: 4
       dim: 1
```

The coefficients for the spline (highest power first) are in the array `A.coefs`

Let's see how we might pull this structure apart and look at functions separately. Here, we will take out the third cubic function, and plot it together with the original spline on the interval $[-1, 10]$.

```
c = A.coefs(3,:);     %Coefficients for cubic
xx= linspace(-1,10);  %Domain for plot

%We could evaluate the polynomial manually:
yy=c(1)*(xx-2).^3+c(2)*(xx-2).^2+c(3)*(xx-2)+c(4);

%We could use Matlab's polyval command to evaluate.
y2=polyval(c,xx,[],[2,1]);

%Here we evaluate the spline:
```

```
y3=ppval(A,xx);

%Plot the spline with the third cubic:
plot(xx,yy,xx,y3,'k-');

%Show that y2 and yy are the same thing:
max(abs(y2-y))
```

Matlab Exercises

1. Use Maple to construct the spline for the data

x	0	1	2	3	4
y	1	-1	2	1	0

2. How would we manipulate the coefficients to get the first, and second derivatives of a spline? Below are two sample $M-$files. The first will return the second derivative, and the second will return the third derivative. Your homework: Write a similar $M-$file named `dspline` that will return the first derivative.

```
function pp=dspline(pp)
% function p2 = dspline(pp)
%  Returns the piecewise polynomial that represents the second
%  derivative of the cubic spline in the data structure pp.
%  Example:  x=0:10; y=sin(x); pp=spline(x,y);
%            p2=dspline(pp);
%            xx=linspace(0,10);
%            yy=ppval(p2,xx);
%            plot(xx,yy);

A=zeros(size(pp.coefs));
A(:,3)=6*pp.coefs(:,1);
A(:,4)=2*pp.coefs(:,2);
pp.coefs=A;
```

Here's an $M-$file for the third derivative.

```
function pp=d3spline(pp)
% function p2 = d3spline(pp)
%  Returns the piecewise polynomial that represents the third
%  derivative of the cubic spline in the data structure pp.
```

```
%  Example:  x=0:10; y=sin(x); pp=spline(x,y);
%            p2=d3spline(pp);
%            xx=linspace(0,10);
%            yy=ppval(p2,xx);
%            plot(xx,yy);

A=zeros(size(pp.coefs));  %Initializes a matrix of zeros
A(:,4)=6*pp.coefs(:,1);
pp.coefs=A;
```

3. Do the same thing as before, but implement the antiderivative of the spline. Call the function `antispline`, and assume the constant of integration is zero. Be sure you also change `pp.order`.

   Use your code to estimate

$$\int_{-1}^{1} \frac{1}{1+x^2}\,dx$$

   Extra: If you plot the antiderivative, you will get a very choppy curve (because each piece starts at zero). Is it possible to change this so that you get a smooth curve starting at zero? How would this code then be used to numerically estimate a definite integral?