1. To visualise cubic spline approximation of a sample data set, type the following commands into MATLAB (note that input after `%` is ignored):

```
>> x=[-3,-2,-1,-.5,0,0.5,1,2,3], % x-values
>> f=[-1,-1,-1,-0.5,0,0.5,1,0.75,0.75], % function values
>> figure(1) % data plot
>> plot(x,f,'ro'), axis square, axis([-3,3,-1.5,1.5]),
>> title('function values'), hold on, shg
>> plot(x,f,'-ro')
>> title('... and the linear spline approximation'), shg
>> t=[-3:0.01:3]; % interpolant evaluation points
>> p=pchip(x,f,t);
>> s=spline(x,f,t);
>> figure(2) % interpolant plot
>> plot(x,f,'ro',t,p,'-b',t,s,'-k'), axis square
>> legend('data','Hermite spline','standard spline', ...
      'location','northwest'), shg
```

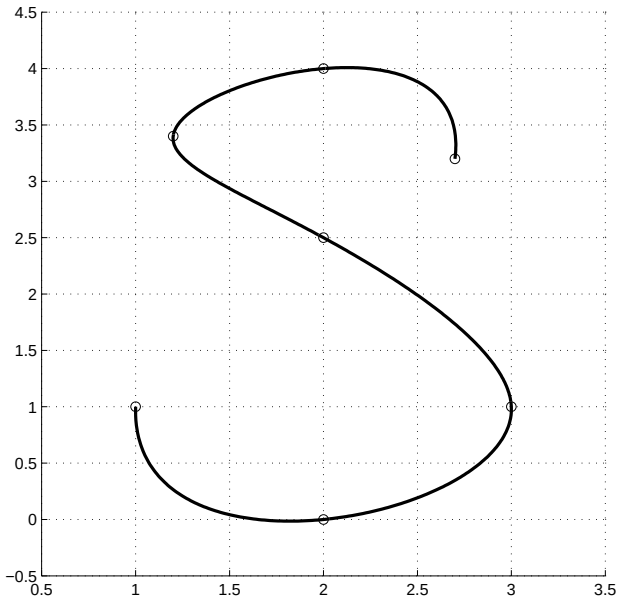
   Make sure that you understand what each individual command typed above actually does, for example, by typing `help pchip`.

2. Use `spline.m` to generate the not-a-knot cubic spline interpolant of the function $f(x) = e^{-2x}\sin(10\pi x)$ over the interval $[0, 1]$. Take twenty equally sized subintervals so that $x_0 = 0$ and $x_{20} = 1$ and plot the knots together with the curve obtained by evaluating the spline function at ten points in every subinterval.

3. Postscript and TrueType letters are created with splines using only a few points for each letter. The MATLAB code below creates a *parametric* spline with the following data:

t	0	1	2	3	4	5	6
x	1	2	3	2	1.2	2	2.7
y	1	0	1	2.5	3.4	4	3.2

   Note that in the resulting curve, y is not a "function" of x; hence a parametric spline is constructed.



MATLAB code:

```
>> x=[1,2,3,2,1.2,2,2.7]; % x-values
>> y=[1,0,1,2.5,3.4,4,3.2]; % y-values
>> n=length(x);
>> data=[x',y'],
>> axis square, hold on
>> t=0:1:n-1; % parametric coordinate
>> tt=[0:0.01:n-1]; % interpolant evaluation points
>> xx=spline(t,x,tt); yy=spline(t,y,tt);
>> plot(xx,yy'), plot(x,y,'o'),
>> grid on, shg
```

Create an M-file `letterS.m` consisting of the above command sequence. Then run the M-file (by typing `letterS`) so as to generate the letter S.

To animate the drawing of the letter, try typing `shg, comet(xx,yy')` having run `letterS`.

4. Building on the previous exercise, create a new M-file which generates the script letter defined by the following data:

t	0	1	2	3	4	5	6	7	8	9	10	11
x	3	1.75	0.9	0	0.5	1.5	3.25	4.25	4.25	3	3.75	6.00
y	4	1.60	0.5	0	1.0	0.5	0.50	2.25	4.00	4	3.25	4.25

On the same axes, plot the original letter together with the letter doubled in size. (Simply rerun the M-file without closing the original figure, with `x` replaced by `2*x` and `y` replaced by `2*y`.) Notice how easy it is to scale the size of a font that is stored in such a way.