

CHAPTER 16

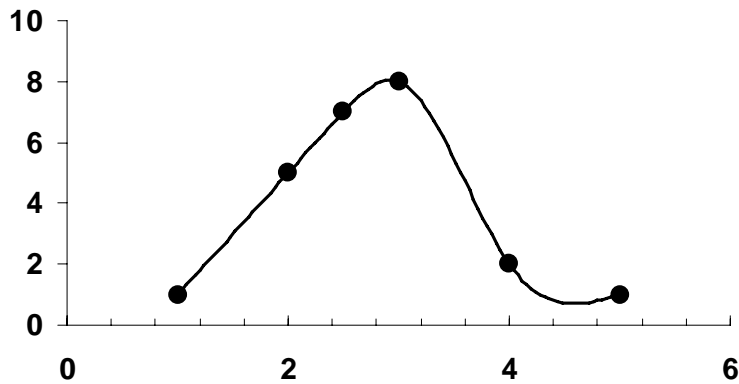
16.1 (a) The simultaneous equations for the natural spline can be set up as

$$\begin{bmatrix} 1 & & & & & \\ 1 & 3 & 0.5 & & & \\ & 0.5 & 2 & 0.5 & & \\ & & 0.5 & 3 & 1 & \\ & & & 1 & 4 & 1 \\ & & & & 1 & \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -6 \\ -24 \\ 15 \\ 0 \end{bmatrix}$$

These equations can be solved for the c 's and then Eqs. (16.21) and (16.18) can be used to solve for the b 's and the d 's. The coefficients for the intervals can be summarized as

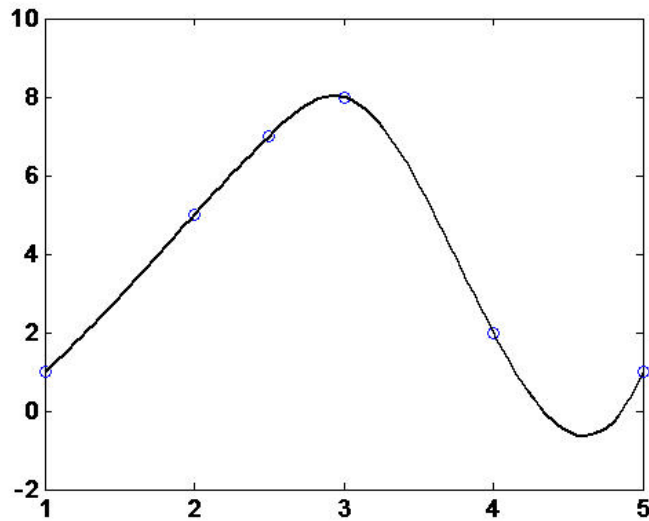
| interval | a | b | c | d |
|----------|-----|----------|----------|----------|
| 1 | 1 | 3.970954 | 0 | 0.029046 |
| 2 | 5 | 4.058091 | 0.087137 | -0.40664 |
| 3 | 7 | 3.840249 | -0.52282 | -6.31535 |
| 4 | 8 | -1.41909 | -9.99585 | 5.414938 |
| 5 | 2 | -5.16598 | 6.248963 | -2.08299 |

These can be used to generate the following plot of the natural spline:



(b) The not-a-knot spline and its plot can be generated with MATLAB as

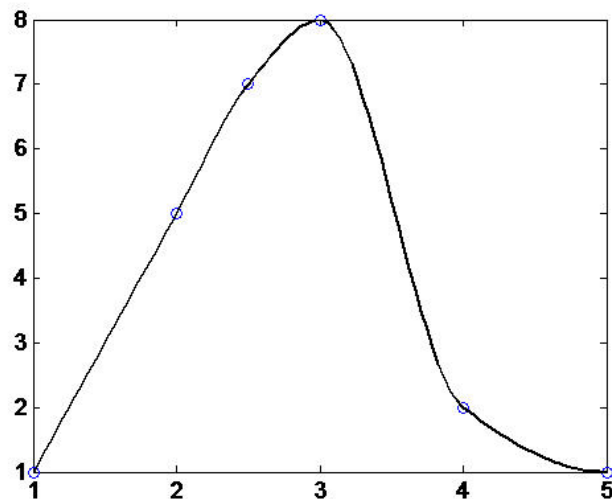
```
>> x = [1 2 2.5 3 4 5];
>> y = [1 5 7 8 2 1];
>> xx = linspace(1,5);
>> yy = spline(x,y,xx);
>> plot(x,y,'o',xx,yy)
```



Notice how the not-a-knot version exhibits much more curvature, particularly between the last points.

(c) The piecewise cubic Hermite polynomial and its plot can be generated with MATLAB as

```
>> x = [1 2 2.5 3 4 5];
>> y = [1 5 7 8 2 1];
>> xx = linspace(1,5);
>> yy = interp1(x,y,xx,'pchip');
>> plot(x,y,'o',xx,yy)
```



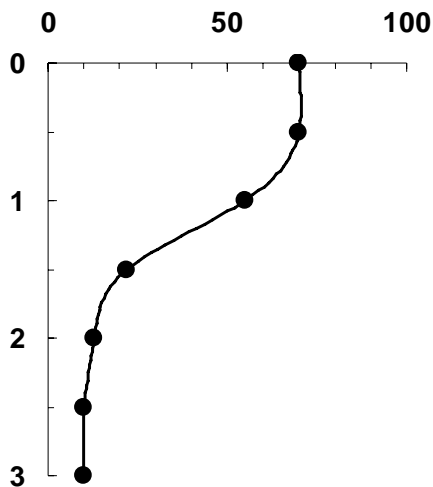
16.2 The simultaneous equations for the clamped spline with zero end slopes can be set up as

$$\begin{bmatrix} 1 & 0.5 & & & & & \\ 0.5 & 2 & 0.5 & & & & \\ & 0.5 & 2 & 0.5 & & & \\ & & 0.5 & 2 & 0.5 & & \\ & & & 0.5 & 2 & 0.5 & \\ & & & & 0.5 & 2 & 0.5 \\ & & & & & 0.5 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 0 \\ -90 \\ -108 \\ 144 \\ 36 \\ 18 \\ \end{bmatrix}$$

These equations can be solved for the c 's and then Eqs. (16.21) and (16.18) can be used to solve for the b 's and the d 's. The coefficients for the intervals can be summarized as

| interval | a | b | c | d |
|----------|-----|----------|----------|----------|
| 1 | 70 | 0 | 15.87692 | -31.7538 |
| 2 | 70 | -7.93846 | -31.7538 | -24.7385 |
| 3 | 55 | -58.2462 | -68.8615 | 106.7077 |
| 4 | 22 | -47.0769 | 91.2 | -66.0923 |
| 5 | 13 | -5.44615 | -7.93846 | 13.66154 |
| 6 | 10 | -3.13846 | 12.55385 | -12.5538 |

The fit can be displayed in graphical form. Note that we are plotting the points as depth versus temperature so that the graph depicts how the temperature changes down through the tank.



Inspection of the plot indicates that the inflection point occurs in the 3rd interval. The cubic equation for this interval is

$$T_3(x) = 55 - 58.2462(d - 1) - 68.8615(d - 1)^2 + 106.7077(d - 1)^3$$

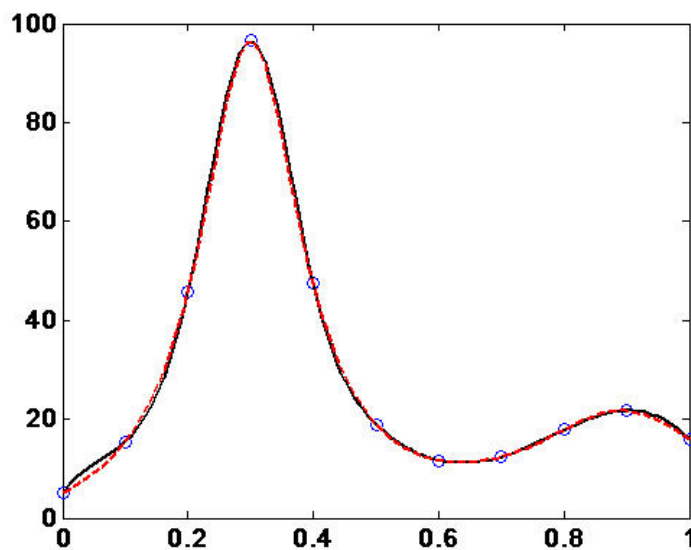
where T = temperature and d = depth. This equation can be differentiated twice to yield the second derivative

$$\frac{d^2 T_3(x)}{dx^2} = -137.729 + 640.2462(d-1)$$

This can be set equal to zero and solved for the depth of the thermocline as $d = 1.21511$ m.

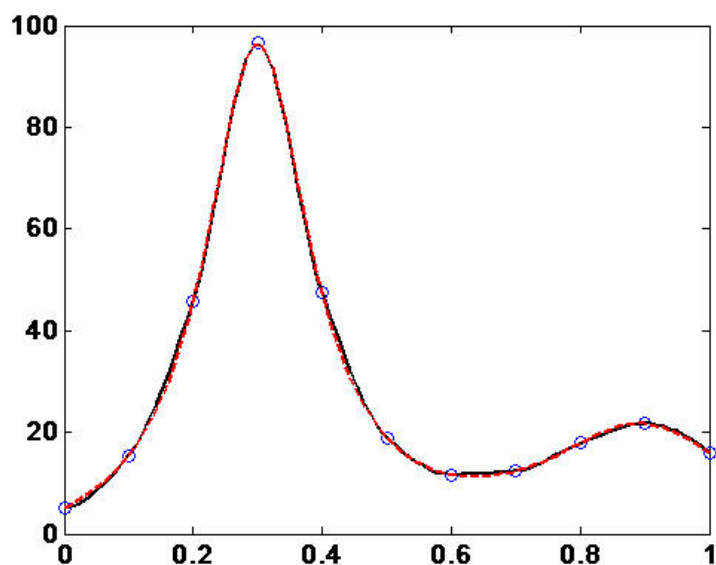
16.3 (a) The not-a-knot fit can be set up in MATLAB as

```
>> x = linspace(0,1,11);
>> y = 1./((x-0.3).^2+0.01)+1./((x-0.9).^2+0.04)-6;
>> xx = linspace(0,1);
>> yy = spline(x,y,xx);
>> yh = 1./((xx-0.3).^2+0.01)+1./((xx-0.9).^2+0.04)-6;
>> plot(x,y,'o',xx,yy,xx,yh,'--')
```



(b) The piecewise cubic Hermite polynomial fit can be set up in MATLAB as

```
>> x = linspace(0,1,11);
>> y = 1./((x-0.3).^2+0.01)+1./((x-0.9).^2+0.04)-6;
>> xx = linspace(0,1);
>> yy = interp1(x,y,xx,'pchip');
>> yh = 1./((xx-0.3).^2+0.01)+1./((xx-0.9).^2+0.04)-6;
>> plot(x,y,'o',xx,yy,xx,yh,'--')
```



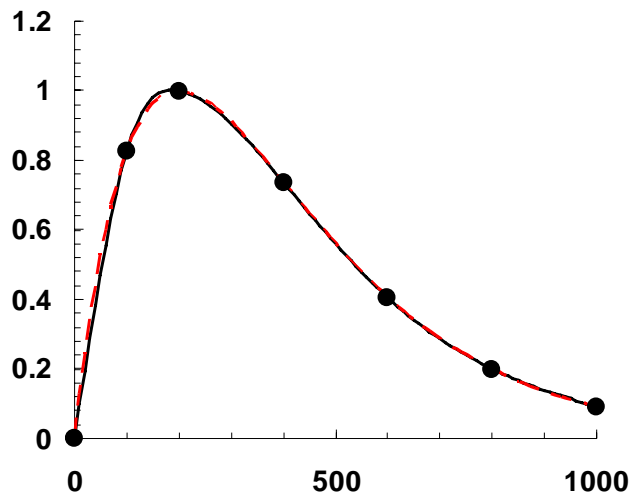
16.4 The simultaneous equations for the clamped spline with zero end slopes can be set up as

$$\begin{bmatrix} 1 & & & & & & \\ 100 & 400 & 100 & & & & \\ & 100 & 600 & 200 & & & \\ & & 200 & 800 & 200 & & \\ & & & 200 & 800 & 200 & \\ & & & & 200 & 800 & 200 \\ & & & & & 1 & \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{Bmatrix} = \begin{Bmatrix} 0 \\ -0.01946 \\ -0.00923 \\ -0.00098 \\ 0.001843 \\ 0.001489 \\ 0 \end{Bmatrix}$$

These equations can be solved for the c 's and then Eqs. (16.21) and (16.18) can be used to solve for the b 's and the d 's. The coefficients for the intervals can be summarized as

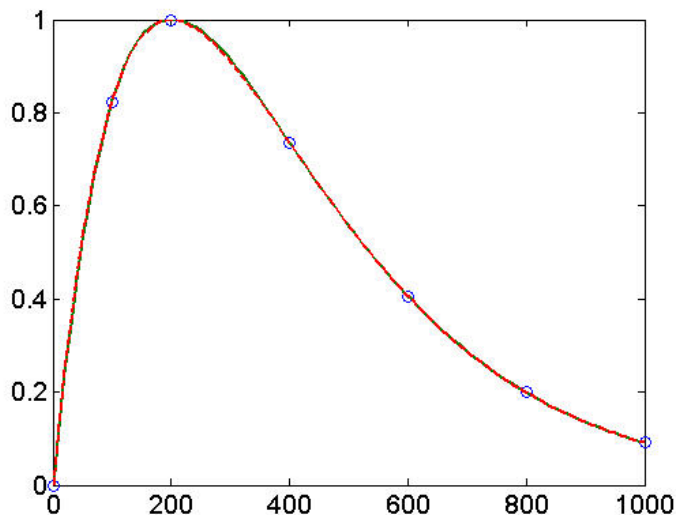
| interval | a | b | c | d |
|----------|----------|----------|----------|----------|
| 1 | 0 | 0.009801 | 0 | -1.6E-07 |
| 2 | 0.824361 | 0.005128 | -4.7E-05 | 1.3E-07 |
| 3 | 1 | -0.00031 | -7.7E-06 | 1.31E-08 |
| 4 | 0.735759 | -0.0018 | 2.13E-07 | 2.82E-09 |
| 5 | 0.406006 | -0.00138 | 1.9E-06 | -8.7E-10 |
| 6 | 0.199148 | -0.00072 | 1.39E-06 | -2.3E-09 |

The fit can be displayed in graphical form as



(b) The not-a-knot fit can be set up in MATLAB as

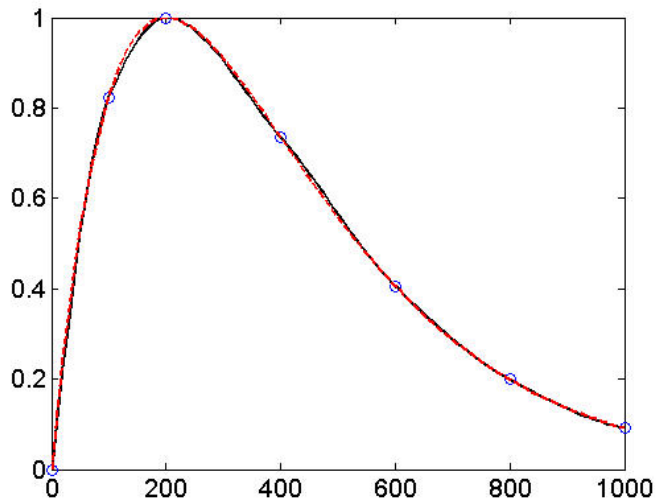
```
>> x = [0 100 200 400 600 800 1000];
>> y = x/200.*exp(-x/200+1);
>> xx = linspace(0,1000);
>> yc = xx/200.*exp(-xx/200+1);
>> yy = spline(x,y,xx);
>> plot(x,y,'o',xx,yy,xx,yc,'--')
```



(c) The piecewise cubic Hermite polynomial fit can be set up in MATLAB as

```
>> x = [0 100 200 400 600 800 1000];
>> y = x/200.*exp(-x/200+1);
>> xx = linspace(0,1000);
>> yc = xx/200.*exp(-xx/200+1);
>> yy = interp1(x,y,xx,'pchip');
```

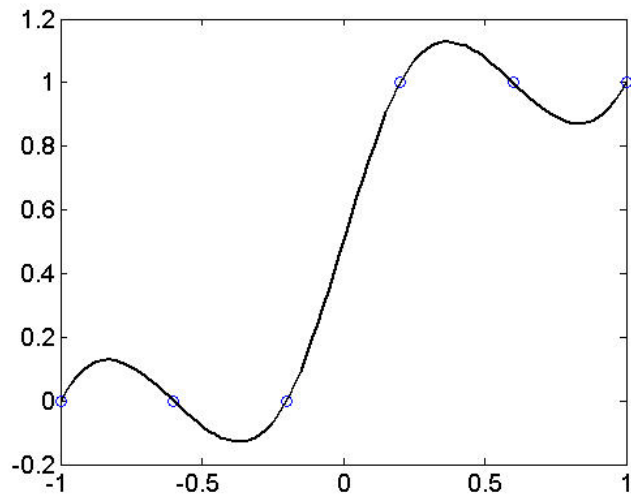
```
>> plot(x,y,'o',xx,yy,xx,yc,'--')
```



Summary: For this case, the not-a-knot fit is the best.

16.5 (a) The not-a-knot fit can be set up in MATLAB as

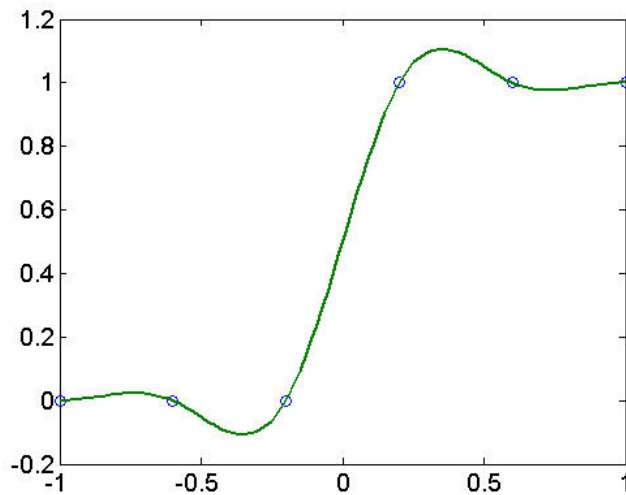
```
>> x = [-1 -0.6 -0.2 0.2 0.6 1];
>> y = [0 0 0 1 1 1];
>> xx = linspace(-1,1);
>> yy = spline(x,y,xx);
>> plot(x,y,'o',xx,yy)
```



(b) The clamped spline with zero end slopes can be set up in MATLAB as

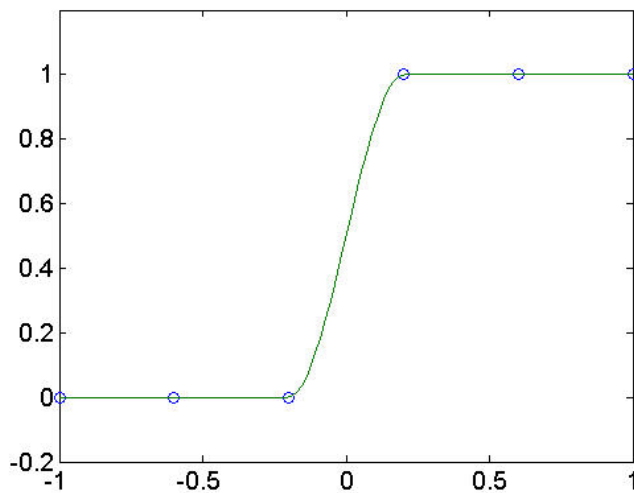
```
>> x = [-1 -0.6 -0.2 0.2 0.6 1];
>> y = [0 0 0 1 1 1];
>> ys = [0 y 0];
```

```
>> xx = linspace(-1,1);
>> yy = spline(x,ys,xx);
>> plot(x,y,'o',xx,yy)
```



(c) The piecewise cubic Hermite polynomial fit can be set up in MATLAB as

```
>> x = [-1 -0.6 -0.2 0.2 0.6 1];
>> y = [0 0 0 1 1 1];
>> xx = linspace(-1,1);
>> yy = interp1(x,y,xx,'pchip');
>> plot(x,y,'o',xx,yy)
```



16.6 An M-file function to implement the natural spline can be written as

```
function yy = natspline(x,y,xx)
% natspline(x,y,xx):
%   uses a natural cubic spline interpolation to find yy, the values
%   of the underlying function y at the points in the vector xx.
```

PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. No part of this Manual may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.


```

% The vector x specifies the points at which the data y is given.

n = length(x);
m = length(xx);
aa(1,1) = 1; aa(n,n) = 1;
bb(1) = 0; bb(n) = 0;
for i = 2:n-1
    aa(i,i-1) = h(x, i - 1);
    aa(i,i) = 2 * (h(x, i - 1) + h(x, i));
    aa(i,i+1) = h(x, i);
    bb(i) = 3 * (fd(i + 1, i, x, y) - fd(i, i - 1, x, y));
end
c = aa\bb';
for i = 1:n - 1
    a(i) = y(i);
    b(i) = fd(i + 1, i, x, y) - h(x, i) / 3 * (2 * c(i) + c(i + 1));
    d(i) = (c(i + 1) - c(i)) / 3 / h(x, i);
end
for i = 1:m
    yy(i) = SplineInterp(x, n, a, b, c, d, xx(i));
end

function hh = h(x, i)
hh = x(i + 1) - x(i);

function fdd = fd(i, j, x, y)
fdd = (y(i) - y(j)) / (x(i) - x(j));

function yyy = SplineInterp(x, n, a, b, c, d, xi)
for ii = 1:n - 1
    if xi >= x(ii) - 0.000001 & xi <= x(ii + 1) + 0.000001
        yyy=a(ii)+b(ii)*(xi-x(ii))+c(ii)*(xi-x(ii))^2+d(ii)*(xi-x(ii))^3;
        break
    end
end
end

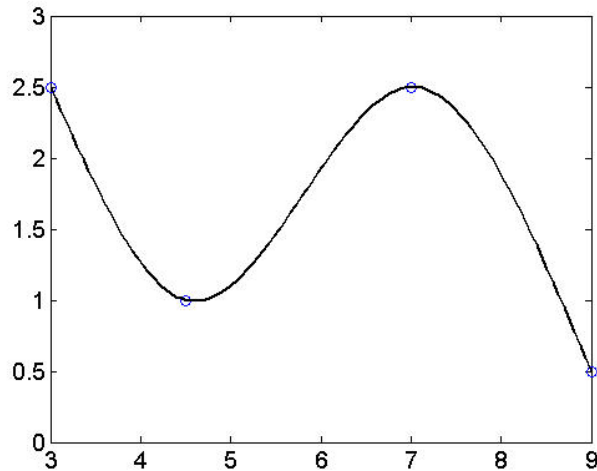
```

The program can be used to duplicate Example 16.3:

```

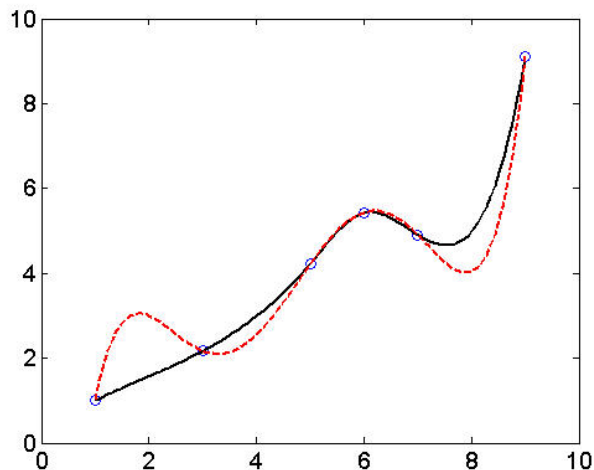
>> x = [3 4.5 7 9];
>> y = [2.5 1 2.5 .5];
>> xx = linspace(3,9);
>> yy = natspline(x,y,xx);
>> plot(x,y,'o',xx,yy)

```



16.7 (a) The not-a-knot fit can be set up in MATLAB as

```
>> x = [1 3 5 6 7 9];
>> y = 0.0185*x.^5-0.444*x.^4+3.9125*x.^3-15.456*x.^2+27.069*x-14.1;
>> xx=linspace(1,9);
>> yy=spline(x,y,xx);
>> yc = 0.0185*xx.^5-0.444*xx.^4+3.9125*xx.^3-15.456*xx.^2+27.069*xx-14.1;
>> plot(x,y,'o',xx,yy,xx,yc,'--')
```



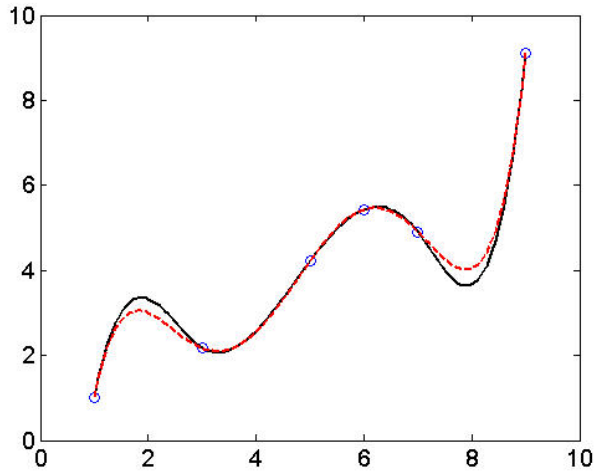
(b) The function can be differentiated to give

$$f'(x) = 0.0925x^4 - 1.776x^3 + 11.7375x^2 - 30.912x + 27.069$$

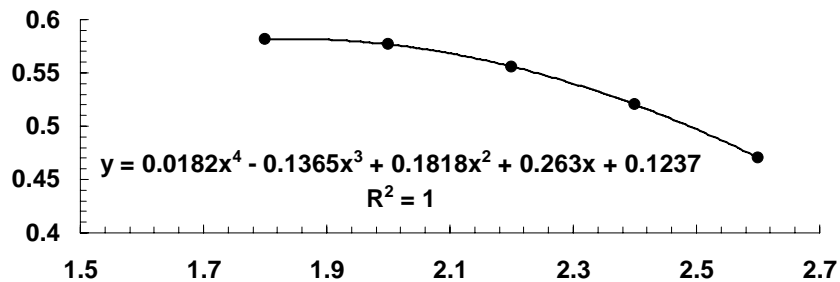
This function can be evaluated at the end nodes to give $f'(1) = 6.211$ and $f'(9) = 11.787$. These values can then be added to the y vector and the `spline` function invoked to develop the clamped fit:

```
>> yd = [6.211 y 11.787];
```

```
>> yy=spline(x,yd,xx);
>> plot(x,y,'o',xx,yy,xx,yc,'--')
```



16.8 (a) A 4th-order interpolating polynomial can be generated as



The polynomial can be used to compute

$$J_1(2.1) = 0.018229(2.1)^4 - 0.13646(2.1)^3 + 0.181771(2.1)^2 + 0.262958(2.1) + 0.1237 = 0.568304$$

The relative error is

$$\varepsilon_t = \left| \frac{0.568292 - 0.568304}{0.568292} \right| \times 100\% = 0.0021\%$$

Thus, the interpolating polynomial yields an excellent result.

(b) A program can be developed to fit natural cubic splines through data based on Fig. 18.18. If this program is run with the data for this problem, the interpolation at 2.1 is 0.56846 which has a relative error of $\varepsilon_t = 0.0295\%$.

A spline can also be fit with MATLAB. It should be noted that MATLAB does not use a natural spline. Rather, it uses a so-called “not-a-knot” spline. Thus, as shown below, although it also yields a very good prediction, the result differs from the one generated with the natural spline,

```
>> format long
>> x=[1.8 2 2.2 2.4 2.6];
>> y=[0.5815 0.5767 0.556 0.5202 0.4708];
>> spline(x,y,2.1)

ans =
    0.56829843750000
```

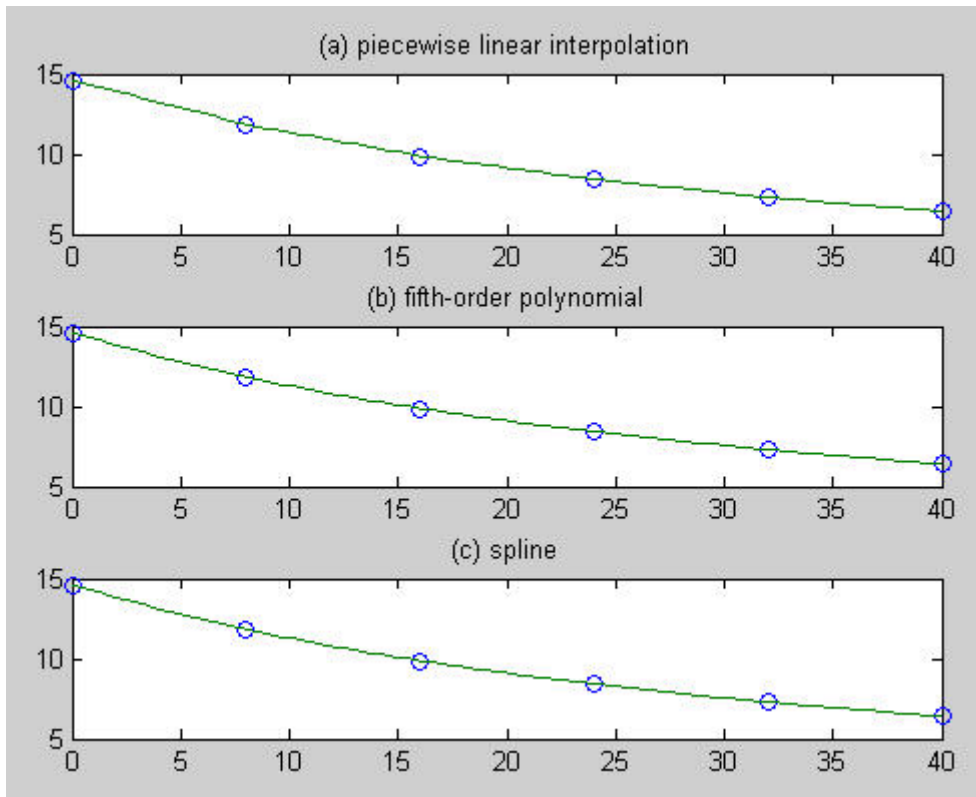
This result has a relative error of $\varepsilon_r = 0.0011\%$.

16.9 The following script generates the solution along with the plots:

```
otrue=7.986;
%(a) piecewise linear interpolation:
T=[0 8 16 24 32 40];
o=[14.621 11.843 9.870 8.418 7.305 6.413];
TT=linspace(0,40);
ol=interp1(T,o,TT);
subplot(3,1,1);plot(T,o,'o',TT,ol)
title('(a) piecewise linear interpolation')
opred=interp1(T,o,27);
et=abs((otrue-opred)/otrue)*100;
fprintf('o(27)-linear = %8.4g et = %8.4g pct\n', opred,et)
%(b) fifth-order polynomial:
a=polyfit(T,o,5);
op=polyval(a,TT);
subplot(3,1,2);plot(T,o,'o',TT,op)
title('(b) fifth-order polynomial')
opred=polyval(a,27);
et=abs((otrue-opred)/otrue)*100;
fprintf('o(27)-polyfit = %8.4g et = %8.4g pct\n', opred,et)
%(c) spline:
os=interp1(T,o,TT,'spline');
subplot(3,1,3);plot(T,o,'o',TT,os)
title('(c) spline')
opred=interp1(T,o,27,'spline');
et=abs((otrue-opred)/otrue)*100;
fprintf('o(27)-spline = %8.4g et = %8.4g pct\n', opred,et)
```

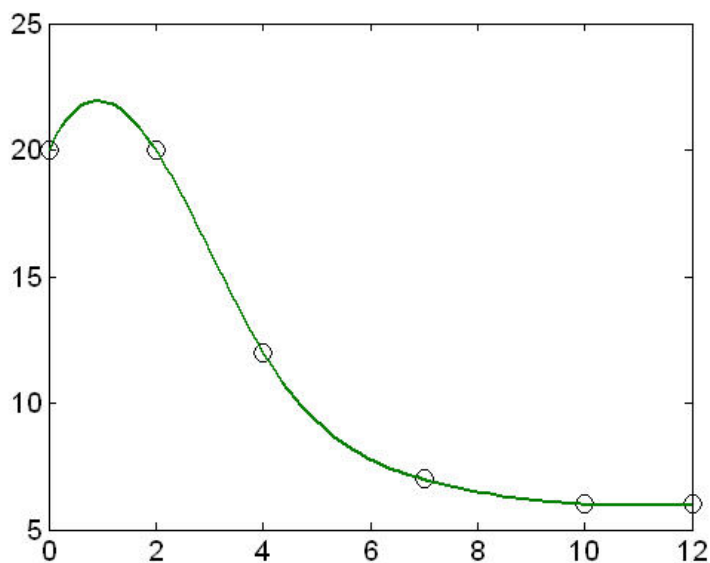
When it is run, the following results are generated:

```
o(27)-linear =      8.001 et =      0.1831 pct
o(27)-polyfit =      7.968 et =      0.2224 pct
o(27)-spline =      7.968 et =      0.2268 pct
```



16.10 (a) Here is a MATLAB session to develop the spline and plot it along with the data:

```
>> x=[0 2 4 7 10 12];
>> y=[20 20 12 7 6 6];
>> xx=linspace(0,12);
>> yy=spline(x,y,xx);
>> plot(x,y,'o',xx,yy)
```



Here is the command to make the prediction:

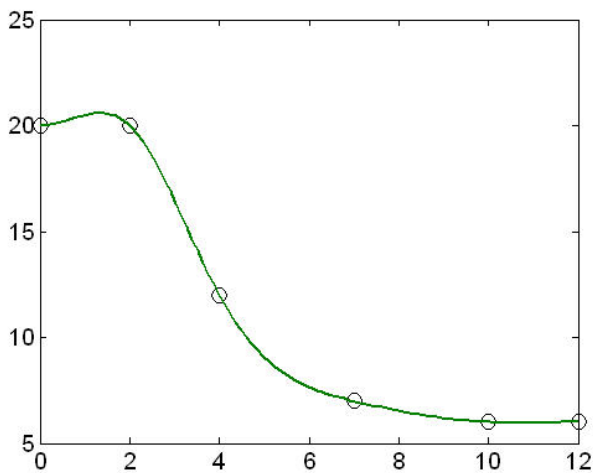
```
>> yp=spline(x,y,1.5)
```

```
yp =  
21.3344
```

(b) To prescribe zero first derivatives at the end knots, the y vector is modified so that the first and last elements are set to the desired values of zero. The plot and the prediction both indicate that there is less overshoot between the first two points because of the prescribed zero slopes.

```
>> yd=[0 y 0];  
>> yy=spline(x,yd,xx);  
>> plot(x,y,'o',xx,yy)  
>> yy=spline(x,yd,1.5)
```

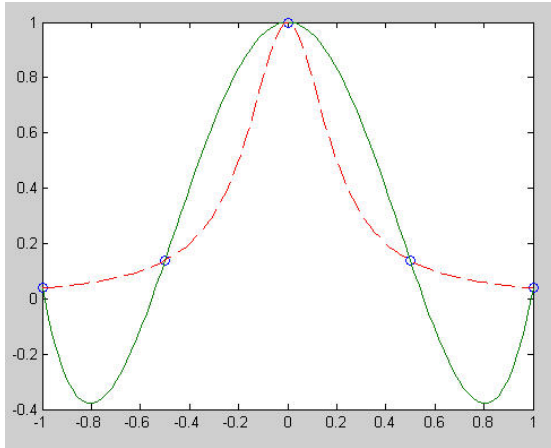
```
yy =  
20.5701
```



Errata: In the first printing, there was no problem 16.11. The following problem was erroneously labeled problem 16.12 in the first printing.

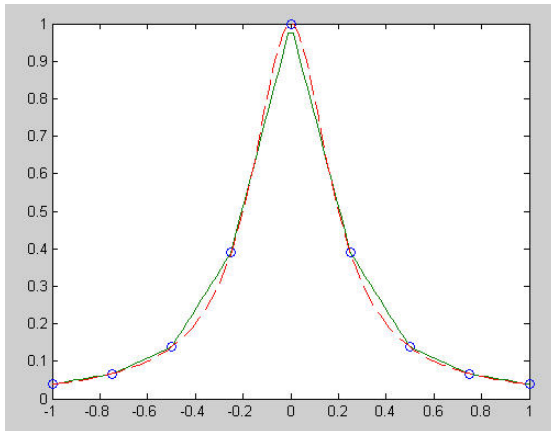
16.11 (a) Eighth-order polynomial:

```
>> x=linspace(-1,1,5);  
>> y=1./(1+25*x.^2);  
>> p=polyfit(x,y,4);  
>> xx=linspace(-1,1);  
>> yy=polyval(p,xx);  
>> yr=1./(1+25*xx.^2);  
>> plot(x,y,'o',xx,yy,xx,yr,'--')
```



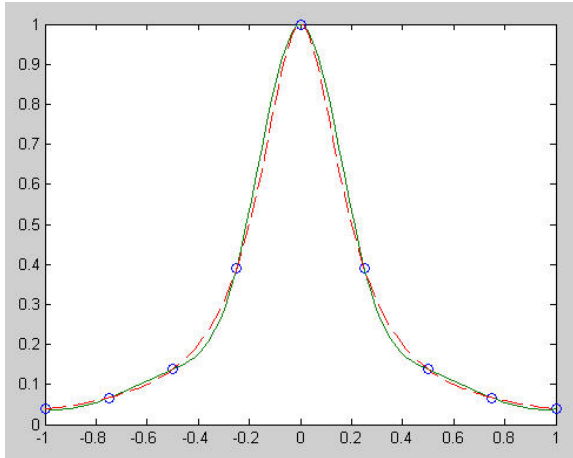
(b) linear spline:

```
>> x=linspace(-1,1,9);
>> y=1./(1+25*x.^2);
>> xx=linspace(-1,1);
>> yy=interp1(x,y,xx);
>> yr=1./(1+25*xx.^2);
>> plot(x,y,'o',xx,yy,xx,yr,'--')
```



(b) cubic spline:

```
>> x=linspace(-1,1,9);
>> y=1./(1+25*x.^2);
>> xx=linspace(-1,1);
>> yy=spline(x,y,xx);
>> yr=1./(1+25*xx.^2);
>> plot(x,y,'o',xx,yy,xx,yr,'--')
```



Errata: In the first printing, there was no problem 16.11. The following problem was erroneously labeled problem 16.13 in the first printing.

16.12 The following script generates the solution along with the plots:

```
t=linspace(0,2*pi,8);
f=sin(t).^2;
tt=linspace(0,2*pi);
ftrue=sin(tt).^2;
%not-a-knot spline
f1=spline(t,f,tt);
subplot(3,1,1);plot(t,f,'o',tt,f1)
Et1=f1-ftrue;
%cubic spline with zero derivatives at the end knots
fp=[0 f 0];
f2=spline(t,fp,tt);
Et2=f1-ftrue;
subplot(3,1,2);plot(t,f,'o',tt,f2)
%piecewise cubic hermite interpolation
f3=pchip(t,f,tt);
subplot(3,1,3);plot(t,f,'o',tt,f3)
Et3=f3-ftrue;
subplot(3,2,1);plot(t,f,'o',tt,f1)
title('not-a-knot')
subplot(3,2,3);plot(t,f,'o',tt,f2)
title('clamped')
subplot(3,2,5);plot(t,f,'o',tt,f3)
title('pchip')
subplot(3,2,2);plot(tt,Et1)
title('error')
subplot(3,2,4);plot(tt,Et2)
title('error')
subplot(3,2,6);plot(tt,Et3)
title('error')
```