Applied Numerical Methods with MATLAB for Engineers and Scientists (4th Edition)

Chapter 18, Problem 6P

Problem

Develop an M-file to compute a cubic spline fit with natural end conditions. Test your code by using it to duplicate Example 18.3.

Example 18.3: Natural Cubic Splines

Problem Statement. Fit cubic splines to the same data used in Examples 18.1 and 18.2 (Table 18.1). Utilize the results to estimate the value at *x* = 5.

Solution. The first step is to employ Eq. (18.27) to generate the set of simultaneous equations that will be utilized to determine the *c* coefficients:

$$\begin{bmatrix} 1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & \\ & h_2 & 2(h_2+h_3) & h_3 \\ & & & 1 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 3(f[x_3,x_2]-f[x_2,x_1]) \\ 3(f[x_4,x_3]-f[x_3,x_2]) \\ 0 \end{Bmatrix}$$

The necessary function and interval width values are

$$\begin{aligned} f_1 &= 2.5 & h_1 &= 4.5-3.0=1.5 \\ f_2 &= 1.0 & h_2 &= 7.0-4.5=2.5 \\ f_3 &= 2.5 & h_3 &= 9.0-7.0=2.0 \\ f_4 &= 0.5 \end{aligned}$$

These can be substituted to yield

$$\begin{bmatrix} 1 & & & \\ 1.5 & 8 & 2.5 & \\ & 2.5 & 9 & 2 \\ & & & 1 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 4.8 \\ -4.8 \\ 0 \end{Bmatrix}$$

These equations can be solved using MATLAB with the results:

$$\begin{aligned} c_1 &= 0 & c_2 &= 0.839543726 \\ c_3 &= -0.766539924 & c_4 &= 0 \end{aligned}$$

Equations (18.21) and (18.18) can be used to compute the *b*'s and *d*'s

$$\begin{aligned} b_1 &= -1.419771863 & d_1 &= 0.186565272 \\ b_2 &= -0.160456274 & d_2 &= -0.214144487 \\ b_3 &= 0.022053232 & d_3 &= 0.127756654 \end{aligned}$$

These results, along with the values for the *a*'s [Eq. (18.11)], can be substituted into Eq. (18.10) to develop the following cubic splines for each interval:

$$s_1(x) = 2.5 - 1.419771863(x-3) + 0.186565272(x-3)^3$$

$$s_2(x) = 1.0 - 0.160456274(x-4.5) + 0.839543726(x-4.5)^2 - 0.214144487(x-4.5)^3$$

$$s_3(x) = 2.5 + 0.022053232(x-7.0) - 0.766539924(x-7.0)^2 + 0.127756654(x-7.0)^3$$

The three equations can then be employed to compute values within each interval. For example, the value at *x* = 5, which falls within the second interval, is calculated as

$$s_2(5) = 1.0 - 0.160456274(5-4.5) + 0.839543726(5-4.5)^2 - 0.214144487(5-4.5)^3 = 1.102889734.$$

The total cubic spline fit is depicted in Fig. 18.4*c*.

Example 18.1: First-Order Splines

Problem Statement. Fit the data in Table 18.1 with first-order splines. Evaluate the Function at *x* = 5.

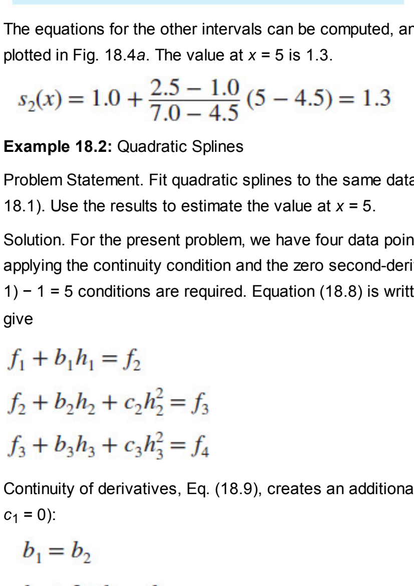
TABLE 18.1 Data to be fit with spline functions.

i	x_i	f_i
1	3.0	2.5
2	4.5	1.0
3	7.0	2.5
4	9.0	0.5

Solution. The data can be substituted into Eq. (18.4) to generate the linear spline functions. For example, for the second interval from *x* = 4.5 to *x* = 7, the function is

$$s_2(x) = 1.0 + \frac{2.5-1.0}{7.0-4.5}(x-4.5)$$

FIGURE 18.4: Spline fits of a set of four points. (*a*) Linear spline, (*b*) quadratic spline, and (*c*) cubic spline, with a cubic interpolating polynomial also plotted.



The equations for the other intervals can be computed, and the resulting first-order splines are plotted in Fig. 18.4*a*. The value at *x* = 5 is 1.3.

$$s_2(x) = 1.0 + \frac{2.5-1.0}{7.0-4.5}(5-4.5) = 1.3$$

Example 18.2: Quadratic Splines

Problem Statement. Fit quadratic splines to the same data employed in Example 18.1 (Table 18.1). Use the results to estimate the value at *x* = 5.

Solution. For the present problem, we have four data points and *n* = 3 intervals. Therefore, after applying the continuity condition and the zero second-derivative condition, this means that 2(4 − 1) − 1 = 5 conditions are required. Equation (18.8) is written for *i* = 1 through 3 (with c_1 = 0) to give

$$f_1 + b_1h_1 = f_2$$

$$f_2 + b_2h_2 + c_2h_2^2 = f_3$$

$$f_3 + b_3h_3 + c_3h_3^2 = f_4$$

Continuity of derivatives, Eq. (18.9), creates an additional 3 − 1 = 2 conditions (again, recall that c_1 = 0):

$$b_1 = b_2$$

$$b_2 + 2c_2h_2 = b_3$$

The necessary function and interval width values are

$$\begin{aligned} f_1 &= 2.5 & h_1 &= 4.5-3.0=1.5 \\ f_2 &= 1.0 & h_2 &= 7.0-4.5=2.5 \\ f_3 &= 2.5 & h_3 &= 9.0-7.0=2.0 \\ f_4 &= 0.5 \end{aligned}$$

These values can be substituted into the conditions which can be expressed in matrix form as

$$\begin{bmatrix} 1.5 & 0 & 0 & 0 & 0 \\ 0 & 2.5 & 6.25 & 0 & 0 \\ 0 & 0 & 0 & 2 & 4 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 5 & -1 & 0 \end{bmatrix} \begin{Bmatrix} b_1 \\ b_2 \\ c_2 \\ b_3 \\ c_3 \end{Bmatrix} = \begin{Bmatrix} -1.5 \\ 1.5 \\ -2 \\ 0 \\ 0 \end{Bmatrix}$$

These equations can be solved using MATLAB with the results:

$$\begin{aligned} b_1 &= -1 \\ b_2 &= -1 & c_2 &= 0.64 \\ b_3 &= 2.2 & c_3 &= -1.6 \end{aligned}$$

These results, along with the values for the *a*'s [Eq. (18.6)], can be substituted into the original quadratic equations to develop the following quadratic splines for each interval:

$$s_1(x) = 2.5 - (x-3)$$

$$s_2(x) = 1.0 - (x-4.5) + 0.64(x-4.5)^2$$

$$s_3(x) = 2.5 + 2.2(x-7.0) - 1.6(x-7.0)^2$$

Because *x* = 5 lies in the second interval, we use s_2 to make the prediction,

$$s_2(5) = 1.0 - (5-4.5) + 0.64(5-4.5)^2 = 0.66$$

The total quadratic spline fit is depicted in Fig. 18.4*b*. Notice that there are two shortcomings that detract from the fit: (1) the straight line connecting the first two points and (2) the spline for the last interval seems to swing too high. The cubic splines in the next section do not exhibit these shortcomings and, as a consequence, are better methods for spline interpolation.

Equation. (18.8):

$$f_i + b_ih_i + c_ih_i^2 = f_{i+1}$$

Eq. (18.9):

$$b_i + 2c_ih_i = b_{i+1}$$

Eq. (18.6):

$$a_i = f_i$$

Eq. (18.4):

$$s_i(x) = f_i + \frac{f_{i+1}-f_i}{x_{i+1}-x_i}(x-x_i)$$

Eq. (18.27):

$$\begin{bmatrix} 1 & & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ & & & & 1 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{Bmatrix} = \begin{Bmatrix} 0 \\ 3(f[x_3,x_2]-f[x_2,x_1]) \\ \vdots \\ 3(f[x_n,x_{n-1}]-f[x_{n-1},x_{n-2}]) \\ 0 \end{Bmatrix}$$

Eq. (18.21):

$$b_i = \frac{f_{i+1}-f_i}{h_i} - \frac{h_i}{3}(2c_i + c_{i+1})$$

Eq. (18.18):

$$d_i = \frac{c_{i+1}-c_i}{3h_i}$$

Eq. (18.10):

$$s_i(x) = a_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)^3$$

Eq. (18.11):

$$a_i = f_i$$

Step-by-step solution

Step 1 of 13

First take a look at a general outline of the cubic spline process. Given data points (x_i, y_i) *i* = 1,2,..., *n*, the cubic spline for this would be the function $s(x)$ which is twice continuously differentiable on the interval $[x_1, x_n]$ and is a cubic polynomial on each subinterval $[x_i, x_{i+1}]$ which interpolates the given data. Assume the cubic spline function is of the form

$$s(x) = \begin{cases} s_1(x) = a_1 + b_1(x-x_1) + c_1(x-x_1)^2 + d_1(x-x_1)^3, \forall x \in [x_1, x_2] \\ s_2(x) = a_2 + b_2(x-x_2) + c_2(x-x_2)^2 + d_2(x-x_2)^3, \forall x \in [x_2, x_3] \\ \vdots \\ s_{n-1}(x) = a_{n-1} + b_{n-1}(x-x_{n-1}) + c_{n-1}(x-x_{n-1})^2 + d_{n-1}(x-x_{n-1})^3, \forall x \in [x_{n-1}, x_n] \end{cases}$$

Comment

Step 2 of 13

The interpolation, continuity and natural end conditions are as follows:

$$\begin{aligned} & s(x_i) = y_i,\ i=1,2,\ldots,n && \text{interpolation conditions} \\ & s_i(x_{i+1}) = s_{i+1}(x_{i+1}),\ i=1,2,\ldots,n-2 && \\ & s_i'(x_{i+1}) = s_{i+1}'(x_{i+1}),\ i=1,2,\ldots,n-2 && \text{continuity conditions} \\ & s_i''(x_{i+1}) = s_{i+1}''(x_{i+1}),\ i=1,2,\ldots,n-2 && \\ & s''(x_1) = s''(x_n) = 0 && \text{natural end conditions} \end{aligned}$$

Comment

Step 3 of 13

After applying these conditions into the cubic polynomials the linear system of equations obtained is

$$\underbrace{\begin{bmatrix} 1 & & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & h_2 & 2(h_2+h_3) & h_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ & & & & & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix}}_{c} = \underbrace{\begin{bmatrix} 0 \\ F_2 \\ \vdots \\ F_{n-1} \\ 0 \end{bmatrix}}_{F}$$

where $h_i = x_{i+1} - x_i,\ i = 1,2,\ldots,n-1$ are the lengths of the subintervals $[x_i, x_{i+1}]$ and

$$F_i = 3\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right),\ i = 2,\ldots,n-1$$

Comment

Step 4 of 13

Once the values of the coefficients c_i's are known the other coefficients a_i, b_i, and d_i can be obtained from the following equations:

$$\left.\begin{aligned} a_i &= y_i, \\ b_i &= \frac{y_{i+1}-y_i}{h_i} - \frac{h_i}{3}(2c_i + c_{i+1}) \\ d_i &= \frac{c_{i+1}-c_i}{3h_i} \end{aligned}\right\},\ i = 1,2,\ldots,n-1 \quad \ldots\ldots (1)$$

Comment

Step 5 of 13

The objective is to implement this cubic spline process in MATLAB. A useful way would be to create a function that takes as input the interpolation data and outputs the cubic spline polynomials' coefficients.

Create a function file and name it **CubicSpline.m** with the following input parameters and output.

```
function [a,b,c,d] = CubicSpline(x,y)
%CUBICSPLINE Given data x,y this function computes the cubic spline of the
%data s(x), assuming the natural end conditions s"(x1)=s"(xn) = 0
% Input: x,y - Arrays of data to be interpolated.
% Output: a,b,c,d - Arrays which contain the coefficients of the cubic
% polynomial si(x) = ai + bi(x-xi)+ ci(x-xi)^2+ di(x-xi)^3 defined on the
% subinterval [xi,xi+1],i=1,2,...n-1
```

Comment

Step 6 of 13

Since the interval lengths $h_i = x_{i+1} - x_i,\ i = 1,2,\ldots,n-1$ will be needed quite often during the code it is better to store these values in an array.

```
n = length(x);
% Compute the interval lengths hi, i=1,2,...n-1
h = x(2:n)-x(1:n-1);
```

Comment

Step 7 of 13

Next the linear system of equations *Ac* = *F* needs to be solved. The matrix *A* and vector *F* need to be created. Generally ***for loops*** would be the way to go. But it is simpler to utilize MATLAB's vector maneuverability. Since *A* is a tridiagonal matrix the MATLAB function **diag(v,k)** is an effective tool to be used. The **diag(v,k)** creates a diagonal matrix with the *k*-th diagonal entries filled by the vector *v*. The integer value for the main diagonal is *k* = 0, for the subdiagonal, that is, the diagonal below the main diagonal, *k* = -1 and for the superdiagonal, the diagonal above the main diagonal, *k* = 1. Using the vector *h* created earlier, the commands to create matrix *A* are

Comment

Step 8 of 13

```
% Start by solving for coefficients ci given by the system of linear
% equations Ac = F. A is a tridiagonal matrix, so diag command is useful.
v1 = [1 2*(h(1:n-2)+h(2:n-1)) 1];     % Diagonal entries of A
v2 = [0 h(2:n-1)];                     % Superdiagonal entries of A
v3 = [h(1:n-2) 0];                     % Subdiagonal entries of A
A = diag(v1)+diag(v2,1)+diag(v3,-1);
```

The '**:**' operator also comes in handy here. This operator allows certain portions of the vector to be accessed instead of having to deal with entire vector at once or elements of the vector one by one.

Comment

Step 9 of 13

Create the right hand side vector *F* and proceed to solve the system of equations for the coefficients *c*.

```
F = [0 3*((y(3:n)-y(2:n-1))./h(2:n-1)-(y(2:n-1)-y(1:n-2))./h(1:n-2)) 0]';
% Solve for c
c = (A\F)';
```

The transpose on the vector *c* is used to maintain it as a row vector or an array rather than a column vector as the remaining coefficients will be arrays.

Comment

Step 10 of 13

Once the values of the coefficients c_i's are obtained use the equations in (1) to compute the values of the other coefficients a_i, b_i and d_i.

```
% Remaining coefficients can be found from c and given data
a = y(1:n-1);
b = (y(2:n)-y(1:n-1))./h -(h/3).*(2*c(1:n-1)+c(2:n));
d = (c(2:n)-c(1:n-1))./(3*h);
c = c(1:n-1);
```

The complete code of the function **CubicSpline(x,y)** in the file **CubicSpline.m** is given below

```
function [a,b,c,d] = CubicSpline(x,y)
%CUBICSPLINE Given data x,y this function computes the cubic spline of the
%data s(x), assuming the natural end conditions s"(x1)=s"(xn) = 0
% Input: x,y - Arrays of data to be interpolated.
% Output: a,b,c,d - Arrays which contain the coefficients of the cubic
% polynomial si(x) = ai + bi(x-xi)+ ci(x-xi)^2+ di(x-xi)^3 defined on the
% subinterval [xi,xi+1],i=1,2,...n-1

n = length(x);
% Compute the interval lengths hi, i=1,2,...n-1
h = x(2:n)-x(1:n-1);

% Start by solving for coefficients ci given by the system of linear
% equations Ac = F. A is a tridiagonal matrix, so diag command is useful.
v1 = [1 2*(h(1:n-2)+h(2:n-1)) 1];     % Diagonal entries of A
v2 = [0 h(2:n-1)];                     % Superdiagonal entries of A
v3 = [h(1:n-2) 0];                     % Subdiagonal entries of A
A = diag(v1)+diag(v2,1)+diag(v3,-1);
F = [0 3*((y(3:n)-y(2:n-1))./h(2:n-1)-(y(2:n-1)-y(1:n-2))./h(1:n-2)) 0]';
% Solve for c
c = (A\F)';

% Remaining coefficients can be found from c and given data
a = y(1:n-1);
b = (y(2:n)-y(1:n-1))./h -(h/3).*(2*c(1:n-1)+c(2:n));
d = (c(2:n)-c(1:n-1))./(3*h);
c = c(1:n-1);

end
```

Comment

Step 11 of 13

Test the code in the function file **CubicSpline.m** on the data given in the textbook example. The **CubicSpline** function requires two arrays as input, which are the interpolation data.

```
>> x = [3 4.5 7 9];
>> y = [2.5 1 2.5 0.5];
>>
```

Comment

Step 12 of 13

Store the output of the function **CubicSpline(x,y)** in the variables *a*, *b*, *c*, and *d*.

```
>> [a,b,c,d] = CubicSpline(x,y);
>>
```

Now verify the results obtained here with the results given in the textbook example.

```
>> a

a =

        2.5        1        2.5

>> b

b =

   -1.41977186311787   -0.160456273764259   0.0220532319391635
```

Step 13 of 13

```
>> c

c =

                    0         0.839543726235742        -0.766539923954373

>> d

d =

        0.186565272496831        -0.214144486692015         0.127756653992395
```

The results obtained here are indeed a match for the results of the example given in the textbook. Next use the cubic spline to estimate the value at x = 5. Since x = 5 falls in the second interval, the cubic spline would be $s_2(x) = a_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3$

```
>> s2 = a(2)+b(2)*(5-x(2))+c(2)*(5-x(2))^2+d(2)*(5-x(2))^3

s2 =

          1.1028897338403

>>
```

Thus the estimated value at x = 5 is $\boxed{s(5) = 1.1028897338403}$

Comment

Was this solution helpful? 1 0

Recommended solutions for you in Chapter 18

Chapter 18, Problem 15P

The U.S. Standard Atmosphere specifies atmospheric properties as a function of altitude above sea level. The following table...

See solution

Chapter 18, Problem 5P

The following data are sampled from the step function depicted in Fig. 18.1:Fit these data with a (a) cubic spline with...

See solution

ABOUT CHEGG
Become a Tutor
Chegg For Good
College Marketing
Corporate Development
Investor Relations
Jobs
Join Our Affiliate Program
Media Center
Site Map

LEGAL & POLICIES
Advertising Choices
Cookie Notice
General Policies
Intellectual Property Rights
International Privacy Policy
Terms of Use
Chegg Tutors Terms of Service
US Privacy Policy
Your CA Privacy Rights
Honor Code

CHEGG PRODUCTS AND SERVICES
Cheap Textbooks
Chegg Coupon
Chegg Play
Chegg Study Help
College Textbooks
eTextbooks
Chegg Math Solver
Mobile Apps
Online Tutoring
Sell Textbooks
Solutions Manual
Study 101
Test Prep
Textbook Rental
Used Textbooks
Digital Access Codes

CHEGG NETWORK
Easybib
Internships.com
Studyblue

CUSTOMER SERVICE
Customer Service
Give Us Feedback
Help with Chegg Tutors
Help with eTextbooks
Help to use EasyBib Plus
Manage Chegg Study Subscription
Return Your Books
Textbook Return Policy

PayPal VERIFIED
OVER 6 MILLION TREES PLANTED

© 2003-2019 Chegg Inc. All rights reserved.