- The known polynomial is used for interpolation.

Program 6-5: User-defined function. Interpolation using Newton's polynomial.

```
function Yint = NewtonsINT(x,y,Xint)
% NewtonsINT fits a Newtons polynomial to a set of given points and
% uses the polynomial to determines the interpolated value of a point.
% Input variables:
% x  A vector with the x coordinates of the given points.
% y  A vector with the y coordinates of the given points.
% Xint  The x coordinate of the point to be interpolated.
% Output variable:
% Yint  The interpolated value of Xint.

n = length(x);
a(1)= y(1);
for i = 1:n - 1
  divDIF(i,1)=(y(i+1)-y(i))/(x(i+ 1)-x(i));
end
for j = 2:n - 1
  for i = 1:n - j
    divDIF(i,j)=(divDIF(i+1,j-1)-divDIF(i,j-1))/(x(j+i)-x(i));
  end
end
for j = 2:n
  a(j) = divDIF(1,j - 1);
end
Yint = a(1);
xn = 1;
for k = 2:n
  xn = xn*(Xint - x(k - 1));
  Yint = Yint + a(k)*xn;
end
```

The length of the vector x gives the number of coefficients (and terms) of the polynomial.

The first coefficient a_1.

Calculate the finite divided differences. They are assigned to the first column of `divDIF`.

Calculate the second and higher divided differences (up to an order of $(n-1)$). The values are assigned in columns to `divDIF`.

Assign the coefficients a_2 through a_n. to vector `a`.

Calculate the interpolated value of `Xint`. The first term in the polynomial is a_1. The following terms are added by using a loop.

The `NewtonsINT(x,y,Xint)` Function is then used in the Command Window for calculating the interpolated value of $x=3$.

```
>> x = [1 2 4 5 7];
>> y = [52 5 -5 -40 10];
>> Yinterpolated = NewtonsINT(x,y,3)
Yinterpolated =
     6
```

6.6 PIECEWISE (SPLINE) INTERPOLATION

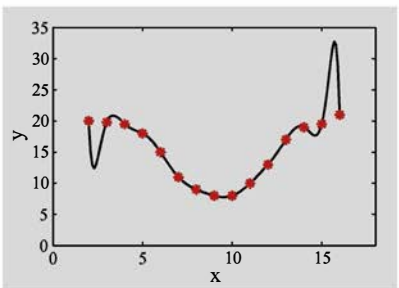


Figure 6-17: Fitting 16 data points with a 15th order polynomial.

When a set of n data points is given and a single polynomial is used for interpolation between the points, the polynomial gives the exact values at the points (passes through the points) and yields estimated (interpolated) values between the points. When the number of points is small such that the order of the polynomial is low, typically the interpolated values are reasonably accurate. However, as already mentioned in Section 6.4, large errors might occur when a high-order polynomial is used for interpolation involving a large number of points. This is shown in Fig. 6-17 where a polynomial of 15th order is used for interpolation with a set of 16 data points. It is clear from the figure that near the ends the polynomial deviates significantly from the trend of the data, and thus cannot be reliably used for interpolation.

When a large number of points is involved, a better interpolation can be obtained by using many low-order polynomials instead of a single high-order polynomial. Each low-order polynomial is valid in one interval between two or several points. Typically, all of the polynomials are of the same order, but the coefficients are different in each interval. When first-order polynomials are used, the data points are connected with straight lines. For second-order (quadratic), and third-order (cubic) polynomials, the points are connected by curves. Interpolation in this way is called ***piecewise***, or ***spline***, interpolation. The data points where the polynomials from two adjacent intervals meet are called ***knots***. The name "spline" comes from a draftsman's spline, which is a thin flexible rod used to physically interpolate over discrete points marked by pegs.

The three types of spline interpolation are linear, quadratic, and cubic.

6.6.1 Linear Splines

With linear splines, interpolation is carried out by using a first-order polynomial (linear function) between the points (the points are connected with straight lines), as shown in Fig. 6-18. Using the Lagrange form, the equation of the straight line that connects the first two points is given by:

$$f_1(x) = \frac{(x-x_2)}{(x_1-x_2)}y_1 + \frac{(x-x_1)}{(x_2-x_1)}y_2 \tag{6.64}$$

For n given points, there are $n-1$ intervals. The interpolation in interval i, which is between points x_i and x_{i+1} $(x_i \le x \le x_{i+1})$, is done by using the equation of the straight line that connects point (x_i, y_i) with point (x_{i+1}, y_{i+1}):

$$f_i(x) = \frac{(x-x_{i+1})}{(x_i-x_{i+1})}y_i + \frac{(x-x_i)}{(x_{i+1}-x_i)}y_{i+1} \quad \text{for} \quad i = 1, 2, \ldots, n-1 \tag{6.65}$$

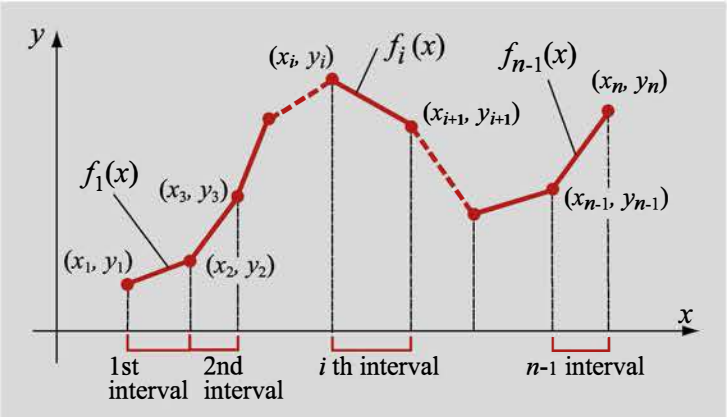


Figure 6-18: Linear splines.

It is obvious that linear splines give continuous interpolation since the two adjacent polynomials have the same value at a common knot. There is, however, a discontinuity in the slope of the linear splines at the knots.

Interpolation with linear splines is easy to calculate and program, and gives good results when the data points are closely spaced. Example 6-6 shows a numerical application of linear splines by hand and by using a user-defined MATLB function.

Example 6-6: Linear splines.

The set of the following four data points is given:

x	8	11	15	18
y	5	9	10	8

(*a*) Determine the linear splines that fit the data.
(*b*) Determine the interpolated value for $x = 12.7$.
(*c*) Write a MATLAB user-defined function for interpolation with linear splines. The inputs to the function are the coordinates of the given data points and the x coordinate of the point at which y is to be interpolated. The output from the function is the interpolated y value at the given point. Use the function for determining the interpolated value of y for $x = 12.7$.

SOLUTION

(*a*) There are four points and thus three splines. Using Eq. (6.65) the equations of the splines are:

$$f_1(x) = \frac{(x-x_2)}{(x_1-x_2)}y_1 + \frac{(x-x_1)}{(x_2-x_1)}y_2 = \frac{(x-11)}{(8-11)}5 + \frac{(x-8)}{(11-8)}9 = \frac{5}{-3}(x-11) + \frac{9}{2}(x-8) \quad \text{for} \quad 8 \le x \le 11$$

$$f_2(x) = \frac{(x-x_3)}{(x_2-x_3)}y_2 + \frac{(x-x_2)}{(x_3-x_2)}y_3 = \frac{(x-15)}{(11-15)}9 + \frac{(x-11)}{(15-11)}10 = \frac{9}{-4}(x-15) + \frac{10}{4}(x-11) \quad \text{for} \quad 11 \le x \le 15$$

$$f_3(x) = \frac{(x-x_4)}{(x_3-x_4)}y_3 + \frac{(x-x_3)}{(x_4-x_3)}y_4 = \frac{(x-18)}{(15-18)}10 + \frac{(x-15)}{(18-15)}8 = \frac{10}{-3}(x-18) + \frac{8}{3}(x-15) \quad \text{for} \quad 15 \le x \le 18$$

(*b*) The interpolated value of y for $x = 12.7$ is obtained by substituting the value of x in the equation for $f_2(x)$ above:

$$f_2(12.7) = \frac{9}{-4}(12.7 - 15) + \frac{10}{4}(12.7 - 11) = 9.425$$

(*c*) The MATLAB user-defined function for linear spline interpolation is named `Yint=LinearSpline(x,y,Xint)`. `x` and `y` are vectors with the coordinates of the given data points, and `Xint` is the coordinate of the point at which y is to be interpolated.

Program 6-6: User-defined function. Linear splines.

```
function Yint = LinearSpline(x, y, Xint)
% LinearSpline calculates interpolation using linear splines.
% Input variables:
% x    A vector with the coordinates x of the data points.
% y    A vector with the coordinates y of the data points.
% Xint    The x coordinate of the interpolated point.
% Output variable:
% Yint    The y value of the interpolated point.

n = length(x);
for i = 2:n
   if Xint < x(i)
       break
   end
end
Yint=(Xint-x(i))*y(i-1)/(x(i-1)-x(i))+(Xint-x(i-1))*y(i)/(x(i)-x(i-1));
```

The length of the vector x gives the number of terms in the data.

Find the interval that includes `Xint`.

Calculate `Yint` with Eq. (6.65).

The `LinearSpline(x,y,Xint)` function is then used in the Command Window for calculating the interpolated value of $x = 12.7$.

```
>> x = [8 11 15 18];
>> y = [5 9 10 8];
>> Yint = LinearSpline(x,y,12.7)
Yint =
    9.4250
```

6.6.2 Quadratic Splines

In quadratic splines, second-order polynomials are used for interpolation in the intervals between the points (Fig. 6-19). For n given points there are $n-1$ intervals, and using the standard form, the equation of the polynomial in the ith interval, between points x_i and x_{i+1}, is given by:

$$f_i(x) = a_i x^2 + b_i x + c_i \quad \text{for} \quad i = 1, 2, \ldots, n-1 \tag{6.66}$$

Overall, there are $n-1$ equations, and since each equation has three coefficients, a total of $3(n-1) = 3n-3$ coefficients have to be determined. The coefficients are determined by applying the following conditions:

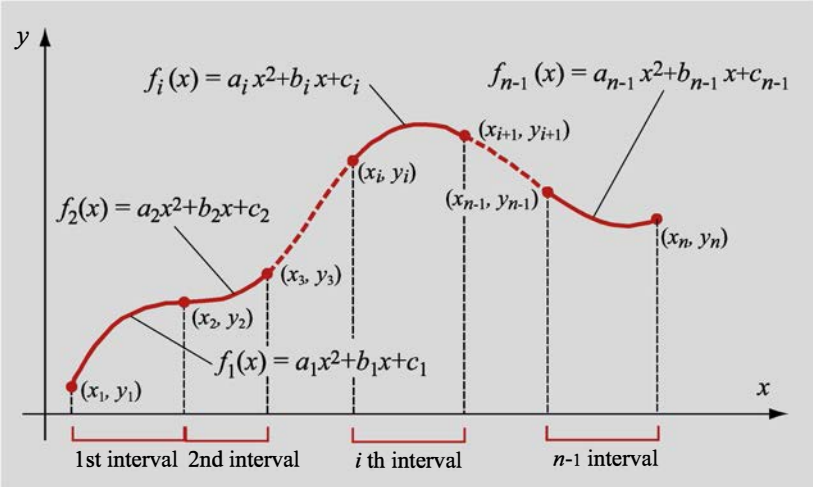


Figure 6-19: Quadratic splines.

1. Each polynomial $f_i(x)$ must pass through the endpoints of the interval, (x_i, y_i) and (x_{i+1}, y_{i+1}), which means that $f_i(x_i) = y_i$ and $f_i(x_{i+1}) = y_{i+1}$:

$$a_i x_i^2 + b_i x_i + c_i = y_i \quad \text{for} \quad i = 1, 2, \ldots, n-1 \tag{6.67}$$

$$a_i x_{i+1}^2 + b_i x_{i+1} + c_i = y_{i+1} \quad \text{for} \quad i = 1, 2, \ldots, n-1 \tag{6.68}$$

   Since there are $n-1$ intervals, these conditions give $2(n-1) = 2n-2$ equations.

2. At the interior knots, the slopes (first derivative) of the polynomials from adjacent intervals are equal. This means that as the curve that passes through an interior knot switches from one polynomial to another, the slope is continuous. In general, the first derivative of the ith polynomial is:

$$f'(x) = \frac{df}{dx} = 2a_i x + b_i \tag{6.69}$$

   For n points, the first interior point is $i = 2$, and the last is $i = n-1$. Equating the successive first derivatives at all of the interior points gives:

$$2a_{i-1} x_i + b_{i-1} = 2a_i x_i + b_i \quad \text{for} \quad i = 2, 3, \ldots, n-1 \tag{6.70}$$

   Since there are $n-2$ interior points, this condition gives $n-2$ equations.

Together, the two conditions give $3n-4$ equations. However, the $n-1$ polynomials have $3n-3$ coefficients so that one more equation (condition) is needed in order to solve for all of the coefficients. An additional condition that is commonly applied is that the second derivative at either the first point or the last point is zero. Consider the first choice:

3. The second derivative at the first point, (x_1, y_1), is zero. The polynomial in the first interval (between the first and the second points) is:

$$f_1(x) = a_1x^2 + b_1x + c_1 \tag{6.71}$$

The second derivative of the polynomial is $f_1''(x) = 2a_1$, and equating it to zero means that $a_1 = 0$. This condition actually means that a straight line connects the first two points (the slope is constant).

A note on quadratic and cubic splines

Quadratic splines have a continuous first derivative at the interior points (knots), and for n given points they require the solution of a linear system of $3n-4$ equations for the coefficients of the polynomials. As is shown in the next section, cubic splines have continuous first and second derivatives at the interior points, and can be written in a form that requires the solution of a linear system of only $n-2$ equations for the coefficients.

Example 6-7 shows an application of quadratic splines for interpolation of a given set of five points.

Example 6-7: Quadratic splines.

The set of the following five data points is given:

x	8	11	15	18	22
y	5	9	10	8	7

(*a*) Determine the quadratic splines that fit the data.
(*b*) Determine the interpolated value of y for $x = 12.7$.
(*c*) Make a plot of the data points and the interpolating polynomials.

SOLUTION

(*a*) There are five points ($n = 5$) and thus four splines ($i = 1, \ldots, 4$). The quadratic equation for the ith spline is:

$$f_i(x) = a_ix^2 + b_ix + c_i$$

There are four polynomials, and since each polynomial has three coefficients, a total of 12 coefficients have to be determined. The coefficients are $a_1, b_1, c_1, a_2, b_2, c_2, a_3, b_3, c_3, a_4, b_4$, and c_4. The coefficient a_1 is set equal to zero (see condition 3). The other 11 coefficients are determined from a linear system of 11 equations.

Eight equations are obtained from the condition that the polynomial in each interval passes through the endpoints, Eqs. (6.67) and (6.68):

$$i = 1 \quad f_1(x) = a_1x_1^2 + b_1x_1 + c_1 = b_1\,8 + c_1 = 5$$

$$f_1(x) = a_1x_2^2 + b_1x_2 + c_1 = b_1\,11 + c_1 = 9$$

$$i = 2 \quad f_2(x) = a_2x_2^2 + b_2x_2 + c_2 = a_2\,11^2 + b_2\,11 + c_2 = 9$$

$$f_2(x) = a_2x_3^2 + b_2x_3 + c_2 = a_2\,15^2 + b_2\,15 + c_2 = 10$$

$$i = 3 \quad f_3(x) = a_3x_3^2 + b_3x_3 + c_3 = a_3\,15^2 + b_3\,15 + c_3 = 10$$
$$f_3(x) = a_3x_4^2 + b_3x_4 + c_3 = a_3\,18^2 + b_3\,18 + c_3 = 8$$
$$i = 4 \quad f_4(x) = a_4x_4^2 + b_4x_4 + c_4 = a_4\,18^2 + b_4\,18 + c_4 = 8$$
$$f_4(x) = a_4x_5^2 + b_4x_5 + c_4 = a_4\,22^2 + b_4\,22 + c_4 = 7$$

Three equations are obtained from the condition that at the interior knots the slopes (first derivative) of the polynomials from adjacent intervals are equal, Eq. (6.70).

$$i = 2 \quad 2a_1x_2 + b_1 = 2a_2x_2 + b_2 \longrightarrow b_1 = 2a_2 11 + b_2 \quad \text{or:} \quad b_1 - 2a_2 11 - b_2 = 0$$
$$i = 3 \quad 2a_2x_3 + b_2 = 2a_3x_3 + b_3 \longrightarrow 2a_2 15 + b_2 = 2a_3 15 + b_3 \quad \text{or:} \quad 2a_2 15 + b_2 - 2a_3 15 - b_3 = 0$$
$$i = 4 \quad 2a_3x_4 + b_3 = 2a_4x_4 + b_4 \longrightarrow 2a_3 18 + b_3 = 2a_4 18 + b_4 \quad \text{or:} \quad 2a_3 18 + b_3 - 2a_4 18 - b_4 = 0$$

The system of 11 linear equations can be written in a matrix form:

$$\begin{bmatrix} 8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 11^2 & 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15^2 & 15 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15^2 & 15 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 18^2 & 18 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18^2 & 18 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 22^2 & 22 & 1 \\ 1 & 0 & -22 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 1 & 0 & -30 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 36 & 1 & 0 & -36 & -1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ a_3 \\ b_3 \\ c_3 \\ a_4 \\ b_4 \\ c_4 \end{bmatrix} = \begin{bmatrix} 5 \\ 9 \\ 9 \\ 10 \\ 10 \\ 8 \\ 8 \\ 7 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.72}$$

The system in Eq. (6.72) is solved with MATLAB:

```
>> A = [8 1 0 0 0 0 0 0 0 0 0; 11 1 0 0 0 0 0 0 0 0 0; 0 0 11^2 11 1 0 0 0 0 0 0
0 0 15^2 15 1 0 0 0 0 0 0; 0 0 0 0 0 15^2 15 1 0 0 0; 0 0 0 0 0 18^2 18 1 0 0 0
0 0 0 0 0 0 0 0 18^2 18 1; 0 0 0 0 0 0 0 0 22^2 22 1; 1 0 -22 -1 0 0 0 0 0 0 0
0 0 30 1 0 -30 -1 0 0 0 0; 0 0 0 0 0 36 1 0 -36 -1 0];
>> B = [5; 9; 9; 10; 10; 8; 8; 7; 0; 0; 0];
>> coefficients = (A\B)'
coefficients =
 1.3333 -5.6667 -0.2708  7.2917 -38.4375  0.0556 -2.5000 35.0000  0.0625 -2.7500  37.2500
```

b_1 c_1 a_2 b_2 c_2 a_3 b_3 c_3 a_4 b_4 c_4

With the coefficients known, the polynomials are:

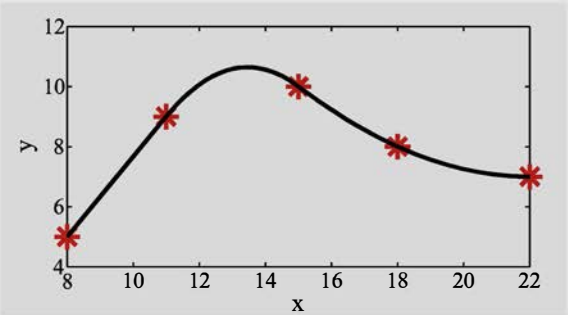
$f_1(x) = 1.333x - 5.6667$ for $8 \le x \le 11$, $\quad f_2(x) = (-0.2708)x^2 + 7.2917x - 38.4375$ for $11 \le x \le 15$

$f_3(x) = 0.0556x^2 - 2.5x + 35$ for $15 \le x \le 18$, $\quad f_4(x) = 0.0625x^2 - 2.75x + 37.25$ for $18 \le x \le 22$

(*b*) The interpolated value of y for $x = 12.7$ is calculated by substituting the value of x in $f_2(x)$:

$$f_2(12.7) = (-0.2708) \cdot 12.7^2 + 7.2917 \cdot 12.7 - 38.4375 = 10.4898$$

(*c*) The plot on the right shows the data points and the polynomials. The plot clearly shows that the first spline is a straight line (constant slope).



6.6.3 Cubic Splines

In cubic splines, third-order polynomials are used for interpolation in the intervals between the points. For n given points there are $n-1$ intervals, and since each third-order polynomial has four coefficients the determination of all of the coefficients may require a large number of calculations. As was explained earlier in this chapter, polynomials can be written in different forms (standard, Lagrange, Newton) and theoretically any of these forms can be used for cubic splines. Practically, however, the amount of calculations that have to be executed for determining all the coefficients varies greatly with the form of the polynomial that is used. The presentation that follows shows two derivations of cubic splines. The first uses the standard form of the polynomials, and the second uses a variation of the Lagrange form. The derivation with the standard form is easier to follow, understand, and use (it is similar to the derivation of the quadratic splines), but it requires the solution of a system of $4n-4$ linear equations. The derivation that is based on the Lagrange form is more sophisticated, but requires the solution of a system of only $n-2$ linear equations.

Cubic splines with standard form polynomials

For n given points, as shown in Fig. 6-20, there are $n-1$ intervals, and using the standard form, the equation of the polynomial in the ith interval, between points x_i and x_{i+1} is given by:

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \tag{6.73}$$

Overall, there are $n-1$ equations, and since each equation has four coefficients, a total of $4(n-1) = 4n-4$ coefficients have to be determined. The coefficients are found by applying the following conditions:

1. Each polynomial $f_i(x)$ must pass through the endpoints of the interval, (x_i, y_i) and (x_{i+1}, y_{i+1}), which means that $f_i(x_i) = y_i$ and $f_i(x_{i+1}) = y_{i+1}$:

$$a_i x_i^3 + b_i x_i^2 + c_i x_i + d_i = y_i \quad \text{for} \quad i = 1, 2, \ldots, n-1 \tag{6.74}$$

$$a_i x_{i+1}^3 + b_i x_{i+1}^2 + c_i x_{i+1} + d_i = y_{i+1} \quad \text{for} \quad i = 1, 2, \ldots, n-1 \tag{6.75}$$

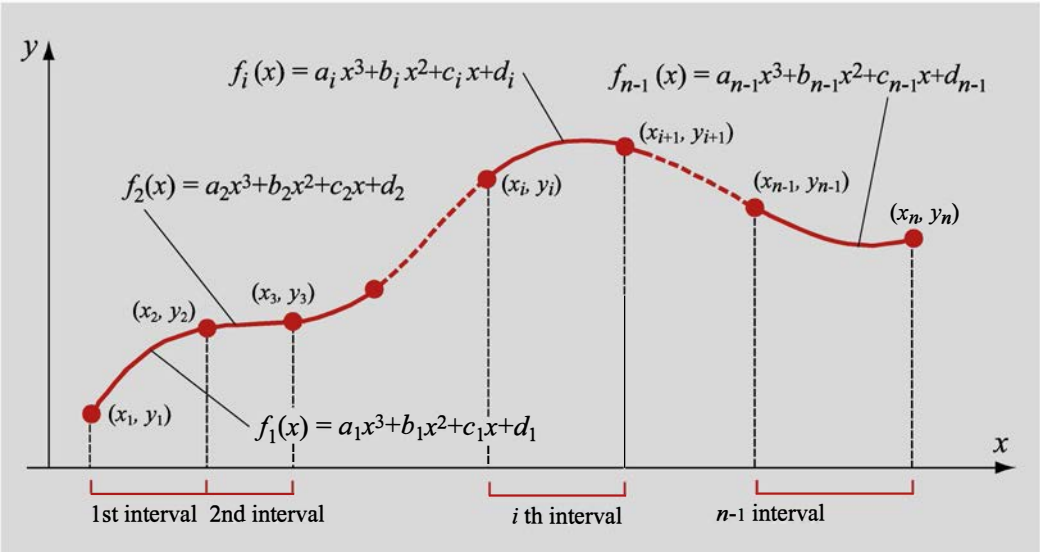


Figure 6-20: Cubic splines.

Since there are $n-1$ intervals, this condition gives $2(n-1) = 2n-2$ equations.

2. At the interior knots, the slopes (first derivatives) of the polynomials from the adjacent intervals are equal. This means that as the curve that passes through an interior knot switches from one polynomial to another, the slope must be continuous. The first derivative of the ith polynomial is:

$$f_i'(x) = \frac{df_i}{dx} = 3a_i x^2 + 2b_i x + c_i \tag{6.76}$$

For n points the first interior point is $i = 2$, and the last is $i = n-1$. Equating the first derivatives at each interior point gives:

$$3a_{i-1}x_i^2 + 2b_{i-1}x_i + c_{i-1} = 3a_i x_i^2 + 2b_i x_i + c_i \text{ for } i = 2, 3, \ldots, n-1 \tag{6.77}$$

Since there are $n-2$ interior points, this condition gives $n-2$ additional equations.

3. At the interior knots, the second derivatives of the polynomials from adjacent intervals must be equal. This means that as the curve that passes through an interior knot switches from one polynomial to another, the rate of change of the slope (curvature) must be continuous. The second derivative of the polynomial in the ith interval is:

$$f_i''(x) = \frac{d^2 f_i}{dx^2} = 6a_i x + 2b_i \tag{6.78}$$

For n points, the first interior point is $i = 2$, and the last is $i = n-1$. Equating the second derivatives at each interior point gives:

$$6a_{i-1}x_i + 2b_{i-1} = 6a_i x_i + 2b_i \text{ for } i = 2, 3, \ldots, n-1 \tag{6.79}$$

Since there are $n-2$ interior points, this condition gives $n-2$ additional equations.

Together, the three conditions give $4n-6$ equations. However, the $n-1$ polynomials have $4n-4$ coefficients, and two more equations (conditions) are needed in order to solve for the coefficients. The additional conditions are usually taken to be that the second derivative is zero at the first point and at the last point. This gives two additional equations:

$$6a_1x_1 + 2b_1 = 0 \text{ and } 6a_{n-1}x_n + 2b_{n-1} = 0 \tag{6.80}$$

Cubic splines with the second derivatives at the endpoints set equal to zero are called ***natural cubic splines***. Applying all the conditions gives a system of $4n-4$ linear equations for the $4n-4$ coefficients. The system can be solved using one of the methods from Chapter 4.

Cubic splines based on Lagrange form polynomials

The derivation of cubic splines using the Lagrange form starts with the second derivative of the polynomial. Figure 6-21 shows spline interpolation with cubic polynomials in (*a*), the first derivatives of the polynomials in (*b*), and their second derivatives in (*c*). The figure shows an *i*th interval with the adjacent $i-1$ and $i+1$ intervals. The second derivative of a third-order polynomial is a linear function. This means that within each spline the second derivative is a linear function of x (see Fig. 6-21*c*). For the *i*th interval, this linear function can be written in the Lagrange form:

$$f_i''(x) = \frac{x-x_{i+1}}{x_i-x_{i+1}}f_i''(x_i) + \frac{x-x_i}{x_{i+1}-x_i}f_i''(x_{i+1}) \tag{6.81}$$

where the values of the second derivative of the third-ordered polynomial at the endpoints (knots) of the *i*th interval are $f_i''(x_i)$ and $f_i''(x_{i+1})$. The third order polynomial in interval i can be determined by integrating Eq. (6.81) twice. The resulting expression contains two constants of integration. These two constants can be determined from the condition that the values of the polynomial at the knots are known:

$$f_i(x_i) = y_i \qquad \text{and} \qquad f_i(x_{i+1}) = y_{i+1}$$

Once the constants of integration are determined, the equation of the third-order polynomial in interval i is given by:

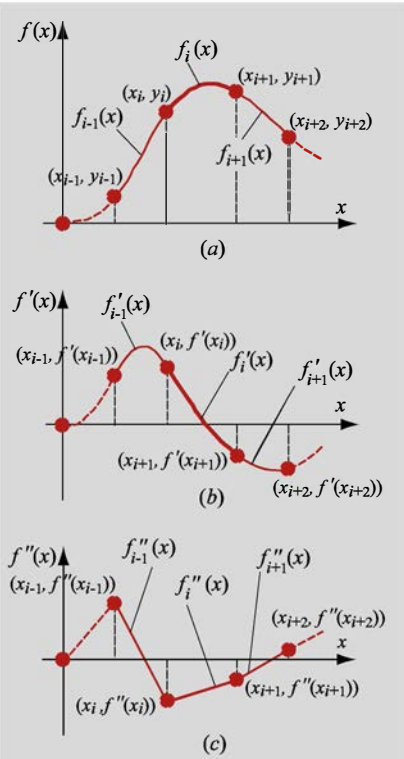


Figure 6-21: Third-order polynomial (*a*) and its first (*b*) and second (*c*) derivatives.

$$f_i(x) = \frac{f_i''(x_i)}{6(x_{i+1}-x_i)}(x_{i+1}-x)^3 + \frac{f_i''(x_{i+1})}{6(x_{i+1}-x_i)}(x-x_i)^3 + \left[\frac{y_i}{x_{i+1}-x_i} - \frac{f_i''(x_i)(x_{i+1}-x_i)}{6}\right](x_{i+1}-x) + \left[\frac{y_{i+1}}{x_{i+1}-x_i} - \frac{f_i''(x_{i+1})(x_{i+1}-x_i)}{6}\right](x-x_i) \quad \text{for } x_i \le x \le x_{i+1} \quad \text{and } i = 1, 2, \ldots, n-1 \tag{6.82}$$

For each interval Eq. (6.82) contains two unknowns, $f_i''(x_i)$ and $f_i''(x_{i+1})$. These are the values of the second derivative at the endpoints of the interval. Equations that relate the values of the second derivatives at the $n-2$ interior points can be derived by requiring continuity of the first derivatives of polynomials from adjacent intervals at the interior knots:

$$f_i'(x_{i+1}) = f_{i+1}'(x_{i+1}) \quad \text{for} \quad i = 1, 2, \ldots, n-2 \tag{6.83}$$

This condition is applied by using Eq. (6.82) to write the expressions for $f_i(x)$ and $f_{i+1}(x)$, differentiating the expressions, and substituting the derivatives in Eq. (6.83). These operations give (after some algebra) the following equations:

$$(x_{i+1}-x_i)f''(x_i) + 2(x_{i+2}-x_i)f''(x_{i+1}) + (x_{i+2}-x_{i+1})f''(x_{i+2}) = 6\left[\frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}} - \frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right] \quad \text{for} \quad i = 1, 2, \ldots, n-2 \tag{6.84}$$

This is a system of $n-2$ linear equations that contains n unknowns.

How is the polynomial in each interval determined?

- For n given data points there are $n-1$ intervals. The cubic polynomial in each interval is given by Eq. (6.82), (total of $n-1$ polynomials).
- The $n-1$ polynomials contain n coefficients $f''(x_1)$, through $f''(x_n)$. These are the values of the second derivatives of the polynomials at the data points. The second derivative at the interior knots is taken to be continuous. This means that at the interior knots, the second derivatives of the polynomials from adjacent intervals are equal. Consequently, for n data points there are n values (the value of the second derivative at each point) that have to be determined.
- Equations (6.84) give a system of $n-2$ linear equations for the n unknown coefficients $f''(x_1)$ through $f''(x_n)$. To solve for the coefficients, two additional relations are required. Most commonly, the second derivative at the endpoints of the data (the first and last

points) is set to zero (natural cubic splines):

$$f''(x_1) = 0 \quad \text{and} \quad f''(x_n) = 0 \tag{6.85}$$

With these conditions, the linear system of Eqs. (6.84) can be solved, and the coefficients can be substituted in the equations for the polynomials (Eqs. (6.82)). Cubic splines with the second derivatives at the endpoints set equal to zero are called ***natural cubic splines***.

Simplified form of the equations

The form of Eqs. (6.82) and (6.84) can be simplified by defining h_i as the length of the ith interval (the intervals do not have to be of the same length):

$$h_i = x_{i+1} - x_i \tag{6.86}$$

and defining a_i as the second derivative of the polynomial at point x_i:

$$a_i = f''(x_i) \tag{6.87}$$

With these definitions, the equation of the polynomial in the ith interval is:

$$\begin{aligned} f_i(x) &= \frac{a_i}{6h_i}(x_{i+1}-x)^3 + \frac{a_{i+1}}{6h_i}(x-x_i)^3 \\ &+ \left[\frac{y_i}{h_i} - \frac{a_i h_i}{6}\right](x_{i+1}-x) + \left[\frac{y_{i+1}}{h_i} - \frac{a_{i+1}h_i}{6}\right](x-x_i) \\ &\text{for } x_i \le x \le x_{i+1} \qquad \text{and} \qquad i = 1, 2, \ldots, n-1 \end{aligned} \tag{6.88}$$

and the system of linear equations that has to be solved for the a_is is given by:

$$\begin{aligned} h_i a_i + 2(h_i + h_{i+1})a_{i+1} + h_{i+1}a_{i+2} &= 6\left[\frac{y_{i+2}-y_{i+1}}{h_{i+1}} - \frac{y_{i+1}-y_i}{h_i}\right] \\ &\text{for} \quad i = 1, 2, \ldots, n-2 \end{aligned} \tag{6.89}$$

To carry out cubic spline interpolation, Eq. (6.89) is used for writing a system of $n-2$ equations with $n-2$ unknowns, a_2 through a_{n-1}. (Remember that with natural cubic splines a_1 and a_n are equal to zero.) Equations (6.89) result in a tridiagonal system of equations that can be solved efficiently using the Thomas algorithm (see Chapter 4). Once the system of equations is solved, the cubic polynomials for each interval can be written using Eq. (6.88). Example 6-8 shows a solution of the problem that is solved in Example 6-7 with cubic splines.

Note on using cubic splines in MATLAB

Cubic splines are available as a built-in function within MATLAB. However, the option labeled `'cubic'` (also called `'pchip'`) is ***not*** the method of cubic splines. Rather, the option labeled `'spline'` is the appropriate option to use for cubic splines. Even when using the option `'spline'`, the user is cautioned that it is not the natural splines

described in this chapter. The cubic splines available in MATLAB under the option `'spline'` use the not-a-knot conditions at the end-points, that is, at the first and last data points. The not-a-knot condition refers to the fact that the third derivatives are continuous at the second point and at the second to last point.

Example 6-8: Cubic splines.

The set of the following five data points is given:

x	8	11	15	18	22
y	5	9	10	8	7

(*a*) Determine the natural cubic splines that fit the data.
(*b*) Determine the interpolated value of y for $x = 12.7$.
(*c*) Plot of the data points and the interpolating polynomials.

SOLUTION

(*a*) There are five points ($n = 5$), and thus four splines ($i = 1, \ldots, 4$). The cubic equation in the ith spline is:

$$f_i(x) = \frac{a_i}{6h_i}(x_{i+1}-x)^3 + \frac{a_{i+1}}{6h_i}(x-x_i)^3 + \left[\frac{y_i}{h_i} - \frac{a_i h_i}{6}\right](x_{i+1}-x) + \left[\frac{y_{i+1}}{h_i} - \frac{a_{i+1}h_i}{6}\right](x-x_i) \quad \text{for} \quad i=1, \ldots 4$$

where $h_i = x_{i+1} - x_i$. The four equations contain five unknown coefficients a_1, a_2, a_3, a_4, and a_5. For natural cubic splines the coefficients a_1 and a_5 are set to be equal to zero. The other three coefficients are determined from a linear system of three equations given by Eq. (6.89).

The values of the h_is are: $h_1 = x_2 - x_1 = 11 - 8 = 3$, $h_2 = x_3 - x_2 = 15 - 11 = 4$

$h_3 = x_4 - x_3 = 18 - 15 = 3$, $h_4 = x_5 - x_4 = 22 - 18 = 4$

$$i = 1 \quad h_1 a_1 + 2(h_1 + h_2)a_2 + h_2 a_3 = 6\left[\frac{y_3 - y_2}{h_2} - \frac{y_2 - y_1}{h_1}\right]$$

$$3 \cdot 0 + 2(3+4)a_2 + 4a_3 = 6\left[\frac{10-9}{4} - \frac{9-5}{3}\right] \longrightarrow 14a_2 + 4a_3 = -6.5$$

$$i = 2 \quad h_2 a_2 + 2(h_2 + h_3)a_3 + h_3 a_4 = 6\left[\frac{y_4 - y_3}{h_3} - \frac{y_3 - y_2}{h_2}\right]$$

$$4a_2 + 2(4+3)a_3 + 3a_4 = 6\left[\frac{8-10}{3} - \frac{10-9}{4}\right] \longrightarrow 4a_2 + 14a_3 + 3a_4 = -5.5$$

$$i = 3 \quad h_3 a_3 + 2(h_3 + h_4)a_4 + h_4 a_5 = 6\left[\frac{y_5 - y_4}{h_4} - \frac{y_4 - y_3}{h_3}\right]$$

$$3a_3 + 2(3+4)a_4 + 4 \cdot 0 = 6\left[\frac{y_5 - y_4}{h_4} - \frac{y_4 - y_3}{h_3}\right] \longrightarrow 3a_3 + 14a_4 = 2.5$$

The system of three linear equations can be written in a matrix form:

$$\begin{bmatrix} 14 & 4 & 0 \\ 4 & 14 & 3 \\ 0 & 3 & 14 \end{bmatrix} \begin{bmatrix} a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} -6.5 \\ -5.5 \\ 2.5 \end{bmatrix} \tag{6.90}$$

The system in Eq. (6.90) is solved with MATLAB:

```
>> A = [14 4 0; 4 14 3; 0 3 14];
>> B = [-6.5; -5.5; 2.5];
```

```
>> coefficients = (A\B)'
coefficients =
 -0.3665 -0.3421 0.2519
```

a_2 a_3 a_4

With the coefficients known, the polynomials are (from Eq. (6.88)):

$$i = 1 \quad f_1(x) = \frac{a_1}{6h_1}(x_2 - x)^3 + \frac{a_2}{6h_1}(x - x_1)^3 + \left[\frac{y_1}{h_1} - \frac{a_1 h_1}{6}\right](x_2 - x) + \left[\frac{y_2}{h_1} - \frac{a_2 h_1}{6}\right](x - x_1)$$

$$f_1(x) = \frac{0}{6 \cdot 3}(11 - x)^3 + \frac{-0.3665}{6 \cdot 3}(x - 8)^3 + \left[\frac{5}{3} - \frac{0 \cdot 3}{6}\right](11 - x) + \left[\frac{9}{3} - \frac{-0.3665 \cdot 3}{6}\right](x - 8)$$

$$f_1(x) = (-0.02036)(x - 8)^3 + 1.667(11 - x) + 3.183(x - 8) \quad \text{for } 8 \le x \le 11$$

$$i = 2 \quad f_2(x) = \frac{a_2}{6h_2}(x_3 - x)^3 + \frac{a_3}{6h_2}(x - x_2)^3 + \left[\frac{y_2}{h_2} - \frac{a_2 h_2}{6}\right](x_3 - x) + \left[\frac{y_3}{h_2} - \frac{a_3 h_2}{6}\right](x - x_2)$$

$$f_2(x) = \frac{-0.3665}{6 \cdot 4}(15 - x)^3 + \frac{-0.3421}{6 \cdot 4}(x - 11)^3 + \left[\frac{9}{4} - \frac{-0.3665 \cdot 4}{6}\right](15 - x) + \left[\frac{10}{4} - \frac{-0.3421 \cdot 4}{6}\right](x - 11)$$

$$f_2(x) = (-0.01527)(15 - x)^3 + (-0.01427)(x - 11)^3 + 2.494(15 - x) + 2.728(x - 11) \quad \text{for } 11 \le x \le 15$$

$$i = 3 \quad f_3(x) = \frac{a_3}{6h_3}(x_4 - x)^3 + \frac{a_4}{6h_3}(x - x_3)^3 + \left[\frac{y_3}{h_3} - \frac{a_3 h_3}{6}\right](x_4 - x) + \left[\frac{y_4}{h_3} - \frac{a_4 h_3}{6}\right](x - x_3)$$

$$f_3(x) = \frac{-0.3421}{6 \cdot 3}(18 - x)^3 + \frac{0.2519}{6 \cdot 3}(x - 15)^3 + \left[\frac{10}{3} - \frac{-0.3421 \cdot 3}{6}\right](18 - x) + \left[\frac{8}{3} - \frac{0.2519 \cdot 3}{6}\right](x - 15)$$

$$f_3(x) = (-0.019)(18 - x)^3 + 0.014(x - 15)^3 + 3.504(18 - x) + 2.5407(x - 15) \quad \text{for } 15 \le x \le 18$$

$$i = 4 \quad f_4(x) = \frac{a_4}{6h_4}(x_5 - x)^3 + \frac{a_5}{6h_4}(x - x_4)^3 + \left[\frac{y_4}{h_4} - \frac{a_4 h_4}{6}\right](x_5 - x) + \left[\frac{y_5}{h_4} - \frac{a_5 h_4}{6}\right](x - x_4)$$

$$f_4(x) = \frac{0.2519}{6 \cdot 4}(22 - x)^3 + \frac{0}{6 \cdot 4}(x - 18)^3 + \left[\frac{8}{4} - \frac{0.2519 \cdot 4}{6}\right](22 - x) + \left[\frac{7}{4} - \frac{0 \cdot 4}{6}\right](x - 18)$$

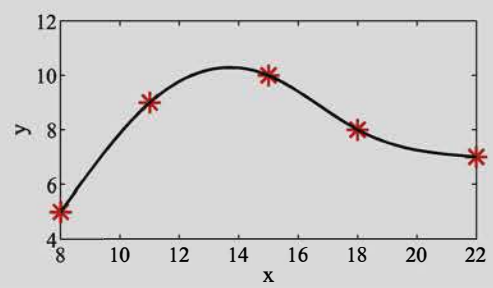
$$f_4(x) = 0.0105(22 - x)^3 + 1.832(22 - x) + 1.75(x - 18) \quad \text{for } 18 \le x \le 22$$

(*b*) The interpolated value of y for $x = 12.7$ is calculated by substituting the value of x in $f_2(x)$:

$$f_2(x) = (-0.01527)(15 - 12.7)^3 + (-0.01427)(12.7 - 11)^3 + 2.494(15 - 12.7) + 2.728(12.7 - 11)$$

$$f_2(x) = 10.11$$

(*c*) The plot on the right shows the data points and the polynomial.



6.7 USE OF MATLAB BUILT-IN FUNCTIONS FOR CURVE FITTING AND INTERPOLATION

MATLAB has built-in functions for curve fitting and interpolation. In addition, MATLAB has an interactive tool for curve fitting, called the basic fitting interface. This section describes how to use the functions `polyfit` (for curve fitting) and `interp1` (for interpolation). Polynomials can be easily used and mathematically manipulated with MATLAB.

The `polyfit` *command*

The `polyfit` command can be used for curve fitting a given set of n points with polynomials of various degrees and for determining the polynomial of order $n-1$ that passes through all the points. The form of the command is:

```
p = polyfit(x,y,m)
```

`p` is a vector of the coefficients of the polynomial that best fits the data.

`x` and `y` are vectors with the horizontal and vertical coordinates of the data points, respectively.

`m` is the degree of the polynomial.

The `interp1` *command*

The `interp1` (the last character is the number one) command executes one-dimensional interpolation at one point. The format of the command is:

```
yi = interp1(x,y,xi,'me hod')
```

`yi` is the interpolated value (dependent variable).

`x` and `y` are vectors with the independent variable and dependent variable of the data points, respectively. `xi` is the value of x at which y is to be interpolated.

Method of interpolation, typed as a string (optional).

- The vector `x` must be monotonic (the elements must be in ascending or descending order).
- `xi` can be a scalar (interpolation at one point) or a vector (interpolation at several points). `yi` is a scalar or a vector with the corresponding interpolated values at the point(s) `xi`.
- MATLAB can interpolate using one of several methods that can be specified. These methods include:

`'nearest'`	returns the value of the data point that is nearest to the interpolated point.
`'linear'`	uses linear spline interpolation.

`'spline'`	uses cubic spline interpolation with "not-a-knot" conditions where the third derivatives at the second and second to last points are continuous. This is not the natural spline presented in this chapter.
`'pchip'`	also called `'cubic'`, uses piecewise cubic Hermite interpolation.

- When the `'nearest'` and the `'linear'` methods are used, the value(s) of `xi` must be within the domain of `x`. If the `'spline'` or the `'pchip'` methods are used, `xi` can have values outside the domain of `x` and the function `interp1` performs extrapolation.
- The `'spline'` method can give large errors if the input data points are nonuniform such that some points are much closer together than others.

Specification of the method is optional. If no method is specified, the default is `'linear'`.

Two examples of using MATLAB's built-in functions for curve fitting and interpolation are shown. First, the `polyfit` function is used for determining the fourth order polynomial that curve-fits the data points in Example 6-3:

```
>> x = 0:0.4:6;
>> y = [0 3 4.5 5.8 5.9 5.8 6.2 7.4 9.6 15.6 20.7 26.7 31.1
35.6 39.3 41.5];
>> p = polyfit(x,y,4)
p =
  -0.2644   3.1185 -10.1927  12.8780  -0.2746
```

The polynomial that corresponds to these coefficients is:
$f(x)=(-0.2644)x^4+3.1185x^3-10.1927x^2+12.878x-0.2746.$

In the second example, the `interp1` command is used for the interpolation in Example 6-8:

```
>> x = [8 11 15 18 22];
>> y = [5 9 10 8 7];
>> xint=8:0.1:22;
>> yint=interp1(x,y,xint,'pchip');
>> plot(x,y,'*',xint,yint)
```

Assign the data points to x and y.

Vector with points for interpolation.

Calculate the interpolated values.

Create a plot with the data points and interpolated values.

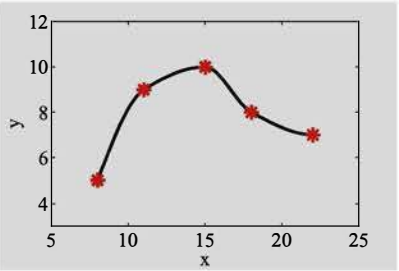


Figure 6-22: Interpolation using MATLAB's `interp1` function.

The resulting plot is shown in Fig. 6-22.

MATLAB also has an interactive tool for curve fitting and interpolation, called the basic fitting interface. To activate the interface, the user first has to make a plot of the data points and then in the Figure Window select ***Basic Fitting*** in the ***Tools*** menu. (A detailed description of the basic fitting interface is available in MATLAB, An Introduction with Applications, Fourth Edition, by Amos Gilat, Wiley, 2011.)

6.8 CURVE FITTING WITH A LINEAR COMBINATION OF NONLINEAR FUNCTIONS

The method of least squares that was applied for curve fitting with a linear function in Section 6.2, and with quadratic and higher-order polynomials in Section 6.4, can be generalized in terms of curve fitting with a linear combination of nonlinear functions. A linear combination of m nonlinear functions can be written as:

$$F(x) = C_1 f_1(x) + C_2 f_2(x) + C_3 f_3(x) + \ldots + C_m f_m(x) = \sum_{j=1}^{m} C_j f_j(x_i) \qquad (6.91)$$

where $f_1, f_2, \ldots, f_m$ are prescribed functions, and $C_1, C_2, \ldots, C_m$ are unknown coefficients. Using least-squares regression, Eq. (6.91) is used to fit a given set of n points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ by minimizing the total error that is given by the sum of the squares of the residuals:

$$E = \sum_{i=1}^{n} \left[y_i - \sum_{j=1}^{m} C_j f_j(x_i) \right]^2 \qquad (6.92)$$

The function E in Eq. (6.92) has a minimum for those values of the coefficients $C_1, C_2, \ldots, C_m$ where the partial derivative of E with respect to each of the coefficients is equal to zero:

$$\frac{\partial E}{\partial C_k} = 0 \quad \text{for } k = 1, 2, \ldots, m \qquad (6.93)$$

Substituting Eq. (6.92) into Eq. (6.93) gives:

$$\frac{\partial E}{\partial C_k} = \sum_{i=1}^{n} 2 \left[y_i - \sum_{j=1}^{m} C_j f_j(x_i) \right] \left[-\frac{\partial}{\partial C_k} \left(\sum_{j=1}^{m} C_j f_j(x_i) \right) \right] = 0$$
$$\text{for } k = 1, 2, \ldots, m \qquad (6.94)$$

Since the coefficients $C_1, C_2, \ldots, C_m$ are independent of each other,

$$\frac{\partial}{\partial C_k} \left(\sum_{j=1}^{m} C_j f_j(x_i) \right) = f_k(x_i) \qquad (6.95)$$

and Eq. (6.94) becomes:

$$\frac{\partial E}{\partial C_k} = -\sum_{i=1}^{n} 2 \left[y_i - \sum_{j=1}^{m} C_j f_j(x_i) \right] f_k(x_i) = 0 \qquad (6.96)$$

The last equation can be rewritten in the form:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} C_j f_j(x_i) f_k(x_i) = \sum_{i=1}^{n} y_i f_k(x_i) \quad \text{for } k = 1, 2, \ldots, m \qquad (6.97)$$

In Eq. (6.97), x_i, y_i, and $f_k(x_i)$ are all known quantities, and the $C_1, C_2, \ldots, C_m$ are the unknowns. The set of Eqs. (6.97) is a system of m linear equations for the unknown coefficients $C_1, C_2, \ldots, C_m$.

The functions $f_k(x)$ can be any functions. For example, if $F(x) = C_1 f_1(x) + C_2 f_2(x)$ such that $f_1(x) = 1$ and $f_2(x) = x$, then Eqs. (6.97) reduce to Eqs. (6.9) and (6.10). If the functions $f_k(x)$ are chosen such that $F(x)$ is quadratic (i.e., $f_1(x) = 1$, $f_2(x) = x$, and $f_3(x) = x^2$), then Eqs. (6.97) reduce to Eqs. (6.23)–(6.25). In general, the functions $f_k(x)$ are chosen because there is a guiding theory that predicts the trend of the data. Example 6-9 shows how the method is used for curve fitting data points with nonlinear approximating functions.

Example 6-9: Curve fitting with linear combination of nonlinear functions.

The following data is obtained from wind-tunnel tests, for the variation of the ratio of the tangential velocity of a vortex to the free stream flow velocity $y = V_\theta / V_\infty$ versus the ratio of the distance from the vortex core to the chord of an aircraft wing, $x = R/C$:

x	0.6	0.8	0.85	0.95	1.0	1.1	1.2	1.3	1.45	1.6	1.8
y	0.08	0.06	0.07	0.07	0.07	0.06	0.06	0.06	0.05	0.05	0.04

Theory predicts that the relationship between x and y should be of the form $y = \frac{A}{x} + \frac{Be^{-2x^2}}{x}$. Find the values of A and B using the least-squares method to fit the above data.

SOLUTION

In the notation of Eq. (6.91) the approximating function is $F(x) = C_1 f_1(x) + C_2 f_2(x)$ with $F(x) = y$, $C_1 = A$, $C_2 = B$, $f_1(x) = \frac{1}{x}$, and $f_2(x) = \frac{e^{-2x^2}}{x}$. The equation has two terms, which means that $m = 2$, and since there are 11 data points, $n = 11$. Substituting this information in Eq. (6.97) gives the following system of two linear equations for A and B.

$$\sum_{i=1}^{11} A \frac{1}{x_i}\frac{1}{x_i} + \sum_{i=1}^{11} B \frac{e^{-2x_i^2}}{x_i}\frac{1}{x_i} = \sum_{i=1}^{11} y_i \frac{1}{x_i} \quad \text{for } k = 1$$

$$\sum_{i=1}^{11} A \frac{1}{x_i}\frac{e^{-2x_i^2}}{x_i} + \sum_{i=1}^{11} B \frac{e^{-2x_i^2}}{x_i}\frac{e^{-2x_i^2}}{x_i} = \sum_{i=1}^{11} y_i \frac{e^{-2x_i^2}}{x_i} \quad \text{for } k = 2$$

These two equations can be rewritten as:

$$A\sum_{i=1}^{11}\frac{1}{x_i^2}+B\sum_{i=1}^{11}\frac{e^{-2x_i^2}}{x_i^2}=\sum_{i=1}^{11}y_i\frac{1}{x_i}$$

$$A\sum_{i=1}^{11}\frac{e^{-2x_i^2}}{x_i^2}+B\sum_{i=1}^{11}\frac{e^{-4x_i^2}}{x_i^2}=\sum_{i=1}^{11}y_i\frac{e^{-2x_i^2}}{x_i}$$

The system can be written in a matrix form:

$$\begin{bmatrix}\sum_{i=1}^{11}\frac{1}{x_i^2} & \sum_{i=1}^{11}\frac{e^{-2x_i^2}}{x_i^2}\\ \sum_{i=1}^{11}\frac{e^{-2x_i^2}}{x_i^2} & \sum_{i=1}^{11}\frac{e^{-4x_i^2}}{x_i^2}\end{bmatrix}\begin{bmatrix}A\\B\end{bmatrix}=\begin{bmatrix}\sum_{i=1}^{11}y_i\frac{1}{x_i}\\ \sum_{i=1}^{11}y_i\frac{e^{-2x_i^2}}{x_i}\end{bmatrix}$$

The system is solved by using MATLAB. The following MATLAB program in a script file solves the system and then makes a plot of the data points and the curve-fitted function.

```
x = [0.6 0.8 0.85 0.95 1.0 1.1 1.2 1.3 1.45 1.6 1.8];
y = [0.08 0.06 0.07 0.07 0.07 0.06 0.06 0.06 0.05 0.05 0.04];
a(1,1) = sum(1./x.^2);
a(1,2) = sum(exp(-2*x.^2)./x.^2);
a(2,1) = a(1,2);
a(2,2) = sum(exp(-4*x.^2)./x.^2);
b(1,1) = sum(y./x);
b(2,1) = sum((y.*exp(-2*x.^2))./x);
AB = a\b
xfit = 0.6:0.02:1.8;
yfit = AB(1)./xfit + AB(2)*exp(-2*xfit.^2)./xfit;
plot(x,y,'o',xfit,yfit)
```

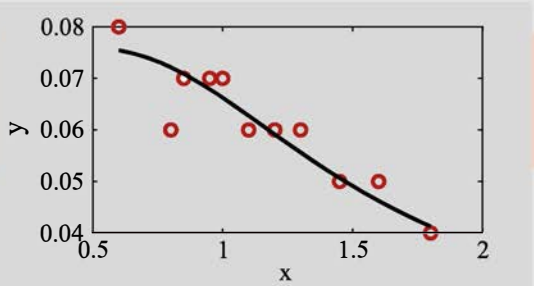
When the program is executed, the solution for the coefficients is displayed in the Command Window (the two elements of the vector `AB`), and a plot with the data points and the curve-fitted function is created.

Command Window:

```
AB =
    0.0743
   -0.0597
```

The coefficient *A*.

The coefficient *B*.



6.9 PROBLEMS

Problems to be solved by hand

Solve the following problems by hand. When needed, use a calculator, or write a MATLAB script file to carry out calculations. If using MATLAB, do not use built-in functions for curve fitting and interpolation.

6.1 The following data is given:

x	1	3	4	6	9	12	14
y	2	4	5	6	7	9	11

(*a*) Use linear least-squares regression to determine the coefficients m and b in the function $y = mx + b$ that best fit the data.
(*b*) Use Eq. (6.5) to determine the overall error.

6.2 The following data is given:

x	–7	–4	–1	0	2	5	7
y	20	14	5	3	–2	–10	–15

(*a*) Use linear least-squares regression to determine the coefficients m and b in the function $y = mx + b$ that best fit the data.
(*b*) Use Eq. (6.5) to determine the overall error.

6.3 The following data give the approximate population of China for selected years from 1900 until 2010:

Year	1900	1950	1970	1980	1990	2000	2010
Population (millions)	400	557	825	981	1135	1266	1370

Assume that the population growth can be modeled with an exponential function $p = be^{mx}$, where x is the year and p is the population in millions. Write the equation in a linear form (Section 6.3), and use linear least-squares regression to determine the constants b and m for which the function best fits the data. Use the equation to estimate the population in the year 1985.

6.4 The following data is given:

x	0.2	0.5	1	2	3
y	3	2	1.4	1	0.6

Determine the coefficients m and b in the function $y = \dfrac{1}{mx + b}$ that best fit the data. Write the equation in a linear form (Section 6.3), and use linear least-squares regression to determine the value of the coefficients.

6.5 The following data is given:

x	–2	–1	0	1	2
y	1.5	3.2	4.5	3.4	2

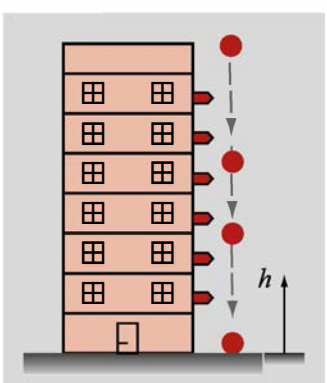
Determine the coefficients a and b in the function $y = \frac{a}{x^2 + b}$ that best fit the data. (Write the function in a linear form (Section 6.3), and use linear least-squares regression to determine the value of the coefficients.) Once the coefficients are determined make a plot that shows the function and the data points.

6.6 The following data is given:

x	1	2	3	5	8
y	0.8	1.9	2.2	3	3.5

Determine the coefficients m and b in the function $y = [m\sqrt{x} + b]^{1/2}$ that best fit the data. Write the equation in a linear form (Section 6.3), and use linear least-squares regression to determine the value of the coefficients.

6.7 To measure g (the acceleration due to gravity), the following experiment is carried out. A ball is dropped from the top of a 100-m-tall building. As the object is falling down, the time t when it passes sensors mounted on the building wall is recorded. The data measured in the experiment is given in the table.



h (m)	100	80	60	40	20	0
t (s)	0	2.02	2.86	3.50	4.04	4.51

In terms of the coordinates shown in the figure, the position of the ball h as a function of the time t is given by $h = h_0 - \frac{1}{2}gt^2$, where $h_0 = 100$ m is the initial position of the ball. Use linear regression to best fit the equation to the data and determine the experimental value of g.

6.8 Water solubility in jet fuel , W_S, as a function of temperature, T, can be modeled by an exponential function of the form $W_S = be^{mT}$. The following are values of water solubility measured at different temperatures. Using linear regression, determine the constants m and b that best fit the data. Use the equation to estimate the water solubility at a temperature of 10° C. Make a plot that shows the function and the data points.

T (°C)	–40	–20	0	20	40
W_S (% wt.)	0.0012	0.002	0.0032	0.006	0.0118

6.9 In an electrophoretic fiber-making process, the diameter of the fiber, d, is related to the current flow, I. The following are measured during production:

I (nA)	300	300	350	400	400	500	500	650	650
d (μm)	22	26	27	30	34	33	33.5	37	42

The relationship between the current and the diameter can be modeled with an equation of the form $d = a + b\sqrt{I}$. Use the data to determine the constants a and b that best fit the data.

6.10 Determine the coefficients of the polynomial $y = a_2x^2 + a_1x + a_0$ that best fit the data given in Problem 6.5.

6.11 Using the method in Section 6.8, determine the coefficients of the equation $y = ax + b/x^2$ that best fit the following data:

x	0.8	1.6	2.4	3.2	4.0
y	6	3.6	4.1	5.1	6.2

6.12 Using the method in Section 6.8, determine the coefficients of the equation $y = Ae^{x/2} + B\sqrt{x} + Cx^2$ that best fit the following data:

x	0.4	1.0	1.6	2.2	2.8
y	5.1	7.1	8	8.1	7.8

6.13 The power generated by a windmill varies with the wind speed. In an experiment, the following five measurements were obtained:

Wind Speed (mph)	14	22	30	38	46
Electric Power (W)	320	490	540	500	480

Determine the fourth-order polynomial in the Lagrange form that passes through the points. Use the polynomial to calculate the power at a wind speed of 26 mph.

6.14 Determine the fourth-order Newton's interpolating polynomial that passes through the data points given in Problem 6.13. Use the polynomial to calculate the power at a wind speed of 26 mph.

6.15 The following data is given:

x	1	2.2	3.4	4.8	6	7
y	2	2.8	3	3.2	4	5

(*a*) Write the polynomial in Lagrange form that passes through the points; then use it to calculate the interpolated value of y at $x = 5.4$.

(*b*) Write the polynomial in Newton's form that passes through the points; then use it to calculate the interpolated value of y at $x = 5.4$.

6.16 Use linear splines interpolation with the data in Problem 6.13 to calculate the power at the following wind speeds.
(*a*) 24 mph (*b*) 35 mph.

6.17 Use quadratic splines interpolation with the data in Problem 6.13 to calculate the power at the following wind speeds.
(*a*) 24 mph (*b*) 35 mph.

6.18 Use natural cubic splines interpolation (based on Lagrange-form polynomials [Eqs. (6.86)–(6.89)]) with the data in Problem 6.13; to calculate the power at the following wind speeds.
(*a*) 24 mph (*b*) 35 mph.

Problems to be programmed in MATLAB

Solve the following problems using MATLAB environment. Do not use MATLAB's built-in functions for curve fitting and interpolation.

6.19 Modify the MATLAB user-defined function `LinearRegression` in Program 6-1. In addition to determining the constants a_1 and a_0, the modified function should also calculate the overall error E according to Eq. (6.6). Name the function `[a, Er] = LinReg(x,y)`. The input arguments `x` and `y` are vectors with the coordinates of the data points. The output argument `a` is a two-element vector with the values of the constants a_1 and a_0. The output argument `Er` is the value of the overall error.
(*a*) Use the function to solve Example 6-1.
(*b*) Use the function to solve Problem 6.2.

6.20 Write a MATLAB user-defined function that determines the best fit of an exponential function of the form $y = be^{mx}$ to a given set of data points. Name the function `[b m]= ExpoFit(x,y)`, where the input arguments `x` and `y` are vectors with the coordinates of the data points, and the output arguments `b` and `m` are the values of the coefficients. The function `ExpoFit` should use the approach that is described in Section 6.3 for determining the value of the coefficients. Use the function to solve Problem 6.8.

6.21 Write a MATLAB user-defined function that determines the best fit of a power function of the form $y = bx^m$ to a given set of data points. Name the function `[b m]= PowerFit(x,y)`, where the input arguments `x` and `y` are vectors with the coordinates of the data points, and the output arguments `b` and `m` are the values of the coefficients. The function `PowerFit` should use the approach that is described in Section 6.3 for determining the value of the coefficients. Use the function to solve Problem 6.3.

6.22 Write a MATLAB user-defined function that determines the coefficients of a quadratic polynomial, $f(x) = a_2x^2 + a_1x + a_0$, that best fits a given set of data points. Name the function `a = QuadFit(x,y)`, where the input arguments `x` and `y` are vectors with the coordinates of the data points, and the output argument `a` is a three-element vector with the values of the coefficients a_2, a_1 and a_0.
(*a*) Use the function to find the quadratic polynomial that best fits the data in Example 6-2.
(*b*) Write a program in a script file that plots the data points and the curve of the quadratic polynomial that best fits the data.

6.23 Write a MATLAB user-defined function that determines the coefficients of a cubic polynomial, $f(x) = a_3x^3 + a_2x^2 + a_1x + a_0$, that best fits a given set of data points. The function should also calculate the overall error E according to Eq. (6.21). Name the function `[a,Er] = CubicPolyFit(x,y)`, where the input arguments `x` and `y` are vectors with the coordinates of the data points, and the output argument `a` is a four-element vector with the values of the coefficients a_3, a_2, a_1, and a_0. The output argument `Er` is the value of the overall error.

(*a*) Use `CubicPolyFit` to determine the cubic polynomial that best fits the data in Example 6-3.

(*b*) Write a program in a script file that plots the data points and the curve of the cubic polynomial that best fits the data.

6.24 Write a MATLAB user-defined function for interpolation with natural cubic splines. Name the function `Yint = CubicSplines(x,y,xint)`, where the input arguments `x` and `y` are vectors with the coordinates of the data points, and `xint` is the x coordinate of the interpolated point. The output argument `Yint` is the y value of the interpolated point.

(*a*) Use the function with the data in Example 6-8 for calculating the interpolated value at $x = 12.7$.

(*b*) Use the function with the data in Problem 6.39 for calculating the enthalpy per unit mass at $T = 14000$ K and at $T = 24000$ K.

6.25 Write a MATLAB user-defined function for spline interpolation that uses third-order Lagrange polynomials. Name the function `Yint = CubicLagSplines(x,y,xint)`, where the input arguments `x` and `y` are vectors with the coordinates of the data points, and `xint` is the x coordinate of the interpolated point. The output argument `Yint` is the y value of the interpolated point. The function uses the following scheme for the interpolation. If `xint` is in the first interval of the data points, the function uses a second-order polynomial that passes through the first three data points. If `xint` is in the last interval of the data points, the function uses a second-order polynomial that passes through the last three data points. If `xint` is in any other interval, lets say interval i between point x_i and point x_{i+1}, the function uses a third-order polynomial for the interpolation. The third-order polynomial is written such that it passes through the data points: x_{i-1}, x_i, x_{i+1}, and x_{i+2}.

(*a*) Use the `CubicLagSplines` function with the data in Problem 6.13 to calculate the power at wind speeds of 26 mph and 42 mph.

(*b*) Use the `CubicLagSplines` function with the data in Example 6-3 to calculate the stress at strains of 0.2 and 3.

6.26 A linear combination of three functions that is used for curve fitting has the form (see Section 6.8):

$$F(x) = C_1 f_1(x) + C_2 f_2(x) + C_3 f_3(x)$$

Write a MATLAB user-defined function that determines the coefficients C_1, C_2, and C_3 that best fits a given set of data points. Name the function `C = NonLinCombFit(F1,F2,F3,x,y)`, where the input arguments `F1`, `F2`, and `F3` are handles of the three functions (user-defined or anonymous) $f_1(x)$, $f_2(x)$, and $f_3(x)$, and `x` and `y` are vectors with the coordinates of the data points. The output argument `C` is a three-element row vector with the values of the coefficients C_1, C_2, and C_3.

Use `NonLinCombFit` to solve Problem 6.40 (*a*). Write a program in a script file that uses `NonLinCombFit` and plots the data points and the curve of $F(x)$ that best fits the data.

Problems in math, science, and engineering

Solve the following problems using the MATLAB environment. As stated, use the MATLAB programs that are presented in the chapter, programs developed in previously solved problems, or MATLAB's built-in functions.

6.27 The resistance R of a tungsten wire as a function of temperature can be modeled with the equation $R = R_0[1 + \alpha(T - T_0)]$, where R_0 is the resistance corresponding to temperature T_0, and α is the temperature coefficient of resistance. Determine R_0 and α such that the equation will best fit the following data. Use $T_0 = 20°\,\text{C}$.

T (°C)	20	100	180	260	340	420	500
R (Ω)	500	676	870	1060	1205	1410	1565

(*a*) Use the user-defined function `LinReg` developed in Problem 6.19.
(*b*) Use MATLAB's built-in function `polyfit`.

6.28 Bacteria growth rate can be modeled with the equation $\ln N_t - \ln N_0 = \mu(t - t_0)$, where μ is the growth rate constant, and N_t and N_0 are the numbers of bacteria at times t and t_0, respectively. Determine μ and N_0 such that the equation will best fit the following data. Use $t_0 = 0$.

t (h)	0	2	4	6	8
N (cells/ml)	35	1990	70,800	2,810,000	141,250,000

(*a*) Use the user-defined function `LinReg` developed in Problem 6.19.
(*b*) Use MATLAB's built-in function `polyfit`.

6.29 The amount of water in air measured at various temperatures at 100% humidity is displayed in the following table:

T (°C)	0	10	20	30	40	50
m_{water} (g/Kg of air)	5	8	15	28	51	94

(*a*) Determines the coefficients of an exponential function of the form $m_{water} = be^{mT}$ that best fits the data. Use the function to estimate the amount of water at 35° C. If available, use the user-defined function `ExpoFit` developed in Problem 6.20. In one figure, plot the exponential function and the data points.
(*b*) Use MATLAB's built-in function `polyfit` to determine the coefficients of a second-order polynomial that best fits the data. Use the polynomial to estimate the amount of water at 35° C. In one figure, plot the polynomial and the data points.

6.30 In a uniaxial tension test, a dog-bone-shaped specimen is pulled in a machine. During the test, the force applied to the specimen, F, and the length of a gage section, L, are measured. The true stress, σ_t, and the true strain, ε_t, are defined by:

$$\sigma_t = \frac{F}{A_0}\frac{L}{L_0} \quad \text{and} \quad \varepsilon_t = \ln\frac{L}{L_0}$$

where A_0 and L_0 are the initial cross-sectional area and gage length, respectively. The true stress–strain curve in the region beyond the yield stress is often modeled by:

$$\sigma_t = K\varepsilon_t^m$$

The following are values of F and L measured in an experiment. Use the approach from Section 6.3 for determining the values of the coefficients K and m that best fit the data. The initial cross-sectional area and gage length are $A_0 = 1.25 \times 10^{-4}\,\text{m}^2$, and $L_0 = 0.0125$ m.

F (kN)	24.6	29.3	31.5	33.3	34.8	35.7	36.6	37.5	38.8	39.6	40.4
L (mm)	12.58	12.82	12.91	12.95	13.05	13.21	13.35	13.49	14.08	14.21	14.48

6.31 The percent of households that own at least one computer in selected years from 1981 to 2010, according to the U.S. census bureau, is listed in the following table:

Year	1981	1984	1989	1993	1997	2000	2001	2003	2004	2010
Household with computer [%]	0.5	8.2	15	22.9	36.6	51	56.3	61.8	65	76.7

The data can be modeled with a function in the form $H_C = C/(1 + Ae^{-Bx})$ (logistic equation), where H_C is percent of households that own at least one computer, C is a maximum value for H_C, A and B are constants, and x is the number of years after 1981. By using the method described in Section 6.3 and assuming that $C = 90$, determine the constants A and B such that the function best fit the data. Use the function to estimate the percent of ownership in 2008 and in 2013. In one figure, plot the function and the data points.

6.32 Use MATLAB's built-in functions to determine the coefficients of the third-order polynomial, $H_C = a_3x^3 + a_2x^2 + a_1x + a_0$ (where x is the number of years after 1981) that best fits the data in Problem 6.31. Use the polynomial to estimate the percent of computer ownership in 2008 and in 2013. In one figure, plot the polynomial and the data points.

6.33 The following data was obtained when the stopping distance d of a car on a wet road was measured as a function of the speed v when the brakes were applied:

v (mi/h)	12.5	25	37.5	50	62.5	75
d (ft)	20	59	118	197	299	420

Determine the coefficients of a quadratic polynomial $d = a_2v^2 + a_1v + a_0$ that best fits the data. Make a plot that show the data points (asterisk marker) and polynomial (solid line).
(*a*) Use the user-defined function `QuadFit` developed in Problem 6.22.
(*b*) Use MATLAB's built-in function `polyfit`.

6.34 Measurements of thermal conductivity, k (W/m K), of silicon at various temperatures, T (K), are:

T (°K)	50	100	150	200	400	600	800	1000
k (W/m K)	28	9.1	4.0	2.7	1.1	0.6	0.4	0.3

The data is to be fitted with a function of the form $k = f(T)$. Determine which of the nonlinear equations that are listed in Table 6-2 can best fit the data and determine its coefficients. Make a plot that shows the data points (asterisk marker) and the equation (solid line).

6.35 Thermistors are resistors that are used for measuring temperature. The relationship between temperature and resistance is given by the Steinhart-Hart equation:

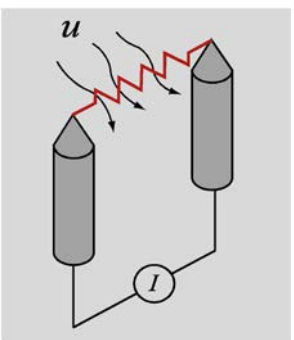
$$\frac{1}{T + 273.15} = C_1 + C_2 \ln(R) + C_3 \ln^3(R)$$

where T is the temperature in degrees Celsius, R is the thermistor resistance in Ω, and C_1, C_2, and C_3, are constants. In an experiment for characterizing a thermistor, the following data was measured:

T (C)	360	320	305	298	295	290	284	282	279	276
R (Ω)	950	3100	4950	6960	9020	10930	13100	14950	17200	18950

Determine the constants C_1, C_2, and C_3 such that the Steinhart-Hart equation will best fit the data.

6.36 A hot-wire anemometer is a device for measuring flow velocity, by measuring the cooling effect of the flow on the resistance of a hot wire. The following data are obtained in calibration tests:



u (ft/s)	4.72	12.49	20.03	28.33	37.47	41.43	48.38	55.06
V (Volt)	7.18	7.3	7.37	7.42	7.47	7.5	7.53	7.55

u (ft/s)	66.77	59.16	54.45	47.21	42.75	32.71	25.43	8.18
V (Volt)	7.58	7.56	7.55	7.53	7.51	7.47	7.44	7.28

Determine the coefficients of the exponential function $u = Ae^{BV}$ that best fit the data.

(*a*) Use the user-defined function `ExpoFit` developed in Problem 6.20.

(*b*) Use MATLAB built-in functions.

In each part make a plot that shows the data points (asterisk marker) and the equation (solid line).

6.37 The data given is to be curve-fitted with the equation $y = axe^{mx}$. Transform the equation to a linear form and determine the constants a and m by using linear least-square regression. (Hint: substitute $v = \ln(y/x)$ and $u = x$.) Make a plot that shows the points (circle markers) and the equation (solid line).

x	0	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
y	0	2.40	3.00	2.55	2.24	1.72	1.18	0.82	0.56	0.42	0.25

6.38 The yield stress of many metals, σ_y, varies with the size of the grains. Often, the relationship between the grain size, d, and the yield stress is modeled with the Hall–Petch equation:

$$\sigma_y = \sigma_0 + kd^{\left(-\frac{1}{2}\right)}$$

The following are results from measurements of average grain size and yield stress:

d (mm)	0.0018	0.0025	0.004	0.007	0.016	0.060	0.25
σ_y (MPa)	530	450	380	300	230	155	115

(*a*) Determine the constants σ_0 and k such that the Hall–Petch equation will best fit the data. Plot the data points (circle markers) and the Hall–Petch equation as a solid line. Use the Hall–Petch equation to estimate the yield stress of a specimen with a grain size of 0.003 mm.

(*b*) Use the user-defined function `QuadFit` from Problem 6.22 to find the quadratic function that best fits the data. Plot the data points (circle markers) and the quadratic equation as a solid line. Use the quadratic equation to estimate the yield stress of a specimen with a grain size of 0.003 mm.

6.39 Values of enthalpy per unit mass, h, of an equilibrium Argon plasma (Ar, Ar^+, A^{++}, A^{+++} ions and electrons) versus temperature are:

$T \times 10^3$ (K)	5	7.5	10	12.5	15	17.5	20	22.5	25	27.5	30
h (MJ/kg)	3.3	7.5	41.8	51.8	61	101.1	132.9	145.5	171.4	225.8	260.9

Write a program in a script file that uses interpolation to calculate h at temperatures ranging from 5000 K to 30000 K in increments of 500 K. The program should generate a plot that shows the interpolated points, and the data points from the table (use an asterisk marker).

(*a*) For interpolation use the user-defined function `CubicSplines` from Problem 6.24.

(*b*) For interpolation use MATLAB's built-in function `interp1` with the `spline` option.

6.40 The following are measurements of the rate coefficient, k, for the reaction $CH_4 + O \rightarrow CH_3 + OH$ at different temperatures, T:

T (K)	595	623	761	849	989	1076	1146	1202	1382	1445	1562
$k \times 10^{20}$ (m^3/s)	2.12	3.12	14.4	30.6	80.3	131	186	240	489	604	868

(*a*) Use the method of least-squares to best fit a function of the form $\ln(k) = C + b\ln(T) - \frac{D}{T}$ to the data. Determine the constants C, b, and D by curve fitting a linear combination of the functions $f_1(T) = 1$, $f_2(T) = \ln(T)$, and $f_3(T) = \frac{-1}{T}$ to the given data (Section 6.8).

(*b*) Usually, the rate coefficient is expressed in the form of an Arrhenius equation $k = AT^b e^{-E_a/(RT)}$, where A and b are constants, $R = 8.314$ J/mole/K is the universal gas constant, and E_a is the activation energy for the reaction. Having determined the constants C, b, and D in part (*a*), deduce the values of A (m^3/s) and E_a (J/mole) in the Arrhenius expression.

6.41 The following measurements were recorded in a study on the growth of trees.

Age (year)	5	10	15	20	25	30	35
Height (m)	5.2	7.8	9	10	10.6	10.9	11.2

The data is used for deriving an equation $H = H(Age)$ that can predict the height of the trees as a function of their age. Determine which of the nonlinear equations that are listed in Table 6-2 can best fit the data and determine its coefficients. Make a plot that shows the data points (asterisk marker) and the equation (solid line).

6.42 The following data present ocean water salinity at different depths:

Depth (m)	0	100	200	300	400	500	600	700	800	900	1100	1400	2000	3000
Salinity (ppt)	35.5	35.35	35.06	34.65	34.48	34.39	34.34	34.32	34.33	34.36	34.45	34.58	34.73	34.79

(*a*) Use interpolation to estimate the water salinity at depths of 250 m, 750 m, and 1800 m by using the user-defined function `CubicSplines` developed in Problem 6.24.
(*b*) Use interpolation to estimate the water salinity at depths of 250 m, 750 m, and 1800 m by using the user-defined function `CubicLagSplines` developed in Problem 6.25.
(*c*) Use MATLAB to create a vector with values of depth ranging from 0 to 3000 m with spacing of 10 m. Then use the MATLAB's built-defined function `interp1` with the option `spline` to calculate corresponding interpolated values of salinity. Make a plot that shows the data points and interpolated points.

6.43 The following data present the power of a diesel engine at different engine speeds:

Engine Speed (rpm)	1200	1500	2000	2500	3000	3250	3500	3750	4000	4400
Engine Power (hp)	65	130	185	225	255	266	275	272	260	230

(*a*) Estimate the engine power at speeds of 2300 rpm and 3650 rpm by using the user-defined function `CubicSplines` developed in Problem 6.24.
(*b*) Estimate the engine power at speeds of 2300 rpm and 3650 rpm by using the user-defined function `CubicLagSplines` developed in Problem 6.25.
(*c*) Use MATLAB to create a vector with values of engine speeds ranging from 1200 to 4400 rpm with spacing of 10 rpm. Use the MATLAB's built-defined function `interp1` with the option `spline` to calculate corresponding interpolated values of engine power. Make a plot that shows the data points and interpolated points.