

Section 5

Splines

Up to this point, we have restricted the approximating functions that were used for interpolation to a single polynomial with a unique definition over the interval corresponding to the set of data points. The results can be less than satisfactory for a number of reasons related to either the order of the polynomial or the general profile of the data points. When the order is too low or too high, relative to the variability and number of data points, the interpolating polynomial may exhibit significant errors.

Figure 5.1 shows a set of nine data points from a function $f(x)$, a second order polynomial $f_2(x)$ through the middle and two end data points, and finally the eighth order polynomial $f_8(x)$ through all the data points. A visual inspection of the data points suggests $f(x)$ approaches a limiting value in an asymptotic fashion. High order polynomials are not well suited when it comes to approximating this type of behavior.

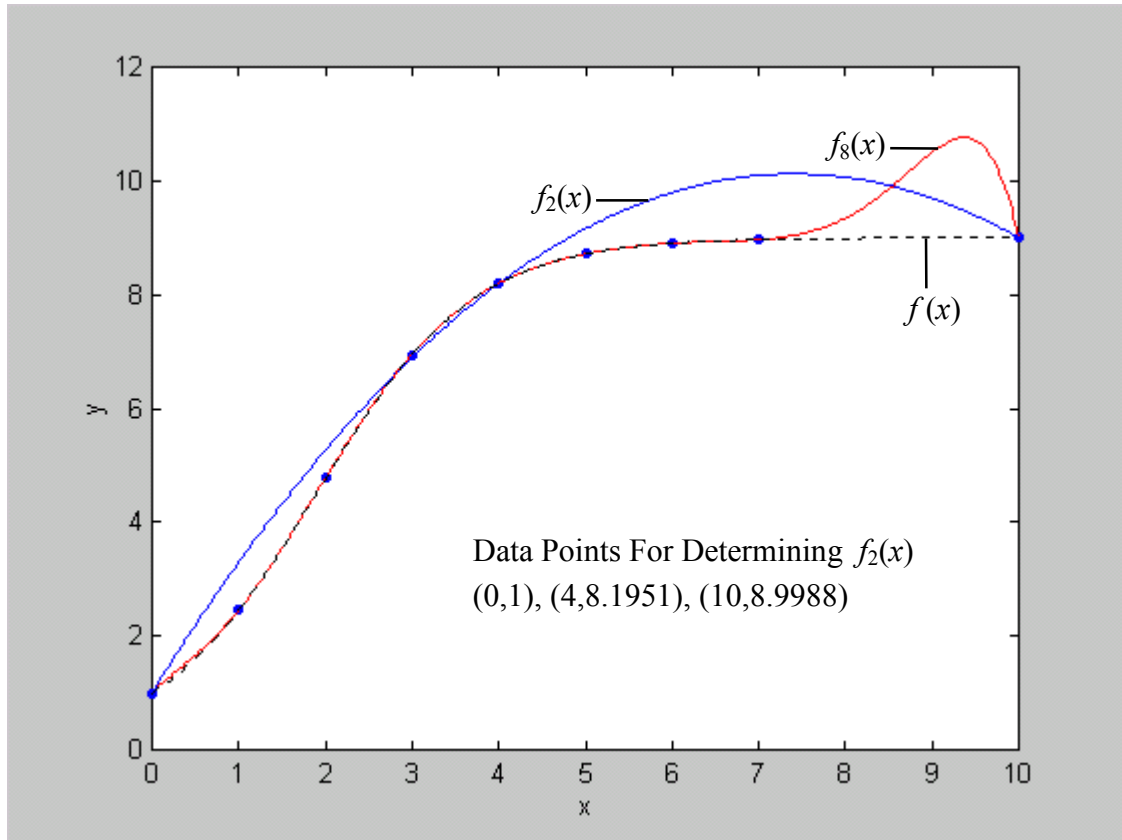


Figure 5.1 Example of a Function $f(x)$ Not Well Suited for Polynomial Approximation

Another approach to obtain an approximating function for interpolation is based on the idea of formulating the approximating function in a piecewise fashion. In fact, we have already done this in a somewhat elementary way. Figure 1.1 in Section 1 illustrates the use of piecewise linear segments to define the approximating function. This is

equivalent to using multiple first order interpolating polynomials restricted to intervals between adjacent data points.

Suppose we wish to approximate a function $f(x)$, defined in tabular form by a set of data points $[x_i, f(x_i)]$, $i = 0, 1, 2, \dots, n$ with lower order polynomials, each valid for a limited portion of the interval $x_0 \leq x \leq x_n$. These lower order polynomials are referred to as spline functions. The term originates from drafting where a smooth curve is drawn through a set of control points using a flexible strip called a "spline". The spline is weighted at various points along its length to keep it in contact with the drafting surface as the curve is drawn. The simplest implementation is to use linear splines in the intervals between consecutive data points. Doing this, we denote the approximating function $F_1(x)$ and the "n" linear splines which comprise it as $f_{1,i}(x)$. This leads to

$$F_1(x) = \begin{cases} f_{1,1}(x), & x_0 \leq x \leq x_1 \\ \vdots \\ f_{1,i}(x), & x_{i-1} \leq x \leq x_i \\ \vdots \\ f_{1,n}(x), & x_{n-1} \leq x \leq x_n \end{cases} \quad (5.1)$$

Using Newton first order divided-difference interpolating polynomials for the linear splines,

$$F_1(x) = \begin{cases} a_1 + b_1(x - x_0), & x_0 \leq x \leq x_1 \\ \vdots \\ a_i + b_i(x - x_{i-1}), & x_{i-1} \leq x \leq x_i \\ \vdots \\ a_n + b_n(x - x_{n-1}), & x_{n-1} \leq x \leq x_n \end{cases} \quad (5.2)$$

The coefficients a_i and b_i define the individual splines $f_{1,i}(x)$ and are given by

$$a_i = f(x_{i-1}), \quad i = 1, 2, 3, \dots, n \quad (5.3)$$

$$b_i = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}, \quad i = 1, 2, 3, \dots, n \quad (5.3a)$$

In the following example, linear splines are applied to the data points in Figure 5.1.

Example 5.1

Suppose $y_i = f(x_i)$ in Figure 5.1 represents the population (in millions) of a country after x_i decades following the initial recorded population. Table 5.1 contains nine population data points.

i	x_i	$y_i = f(x_i)$
0	0	1.0000
1	1	2.4570
2	2	4.7709
3	3	6.9495
4	4	8.1951
5	5	8.7151
6	6	8.9031
7	7	8.9675
8	10	8.9988

Table 5.1 Data Points of $f(x)$ in Figure 5.1

The composite function consisting of linear splines is easily graphed in MATLAB because the 'plot' function by default connects data points by linear segments. Hence, the following MATLAB commands will produce the graph shown in Figure 5.2.

```
>>x=[0:7 10];  
>>v=[0 10 0 10];  
>>F_x=[1 2.4570 4.7709 6.9495 8.1951 8.7151 8.9031 8.9675 8.9988];  
>>plot(x,F_x,'-')  
>>xlabel('x'),ylabel('y')  
>>axis(v)  
>>hold on  
>>axis manual  
>>plot(x,F_x,'-')
```

The graph in Figure 5.2 can provide rough estimates of the interpolated values using $F_1(x)$, especially when the 'zoom' feature of MATLAB is used. However, a direct approach utilizes the 'interp1' function call which returns a single value or array of linearly interpolated values. For example, if we wish to estimate the population at the middle of each decade, the additional MATLAB commands will do the job.

```

>>xint=0.5:1:9.5
>>Fint=interp1(x,F_x,xint)

xint = 0.5000 1.5000 2.5000 3.5000 4.5000 5.5000 6.5000 7.5000 8.5000
       9.5000

Fint = 1.7285 3.6140 5.8602 7.5723 8.4551 8.8091 8.9353 8.9727 8.9831 8.9936

```

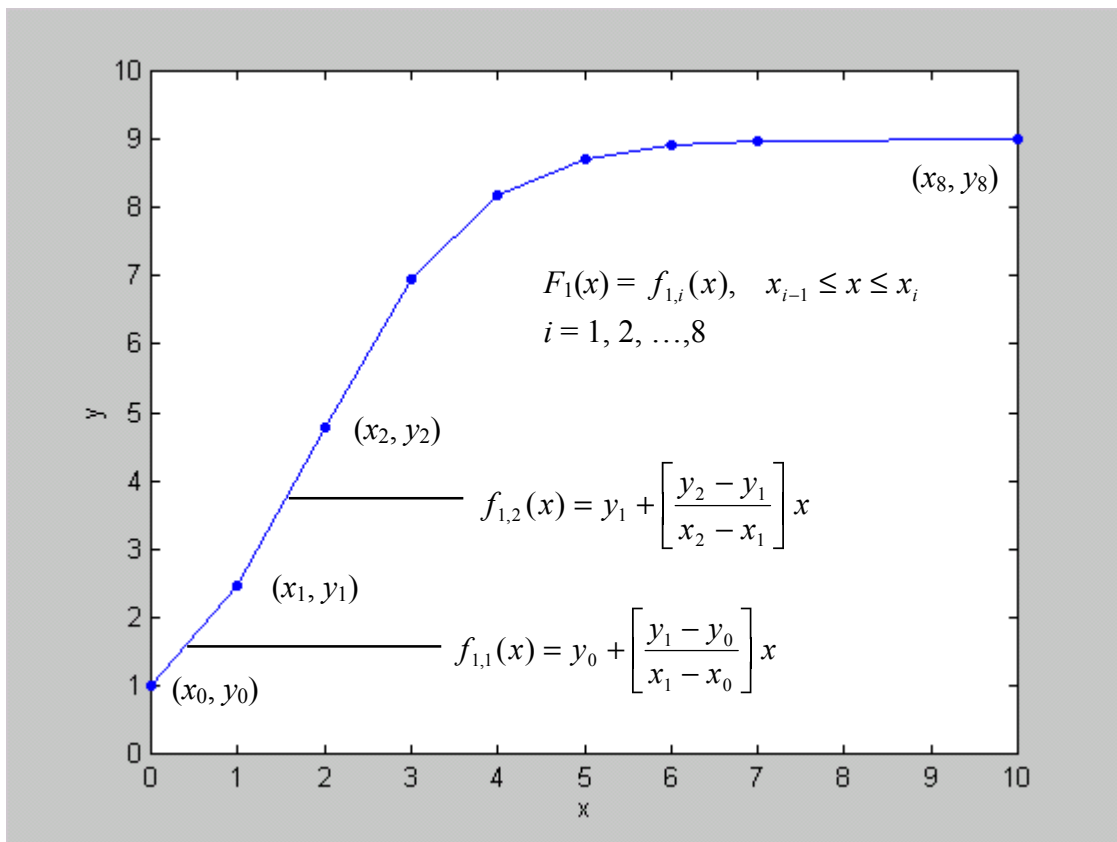


Figure 5.2 Approximating Function for Data in Table 5.1 Consisting of Linear Splines

There are situations when the approximating function is required to pass through each data point and appear smooth at the same time. A composite function consisting of linear splines will satisfy the first requirement. However, unless the data points are very tightly spaced (See Figure 1.3), the combined function will have a zig-zag look as opposed to a smooth appearance. Piecewise continuous functions like $F_1(x)$ in Figure 5.2 are not smooth because of the changes in slope at each of the internal data points, or knots as they are often referred to. Linear spline functions are "tied down" at these knots.

Smoothness refers to the continuity of a function and its derivatives. A function is infinitely smooth at a point if the derivatives of every order are continuous at that point. Clearly, the first derivative of $F_1(x)$ in Figure 5.2 is discontinuous at each knot. Unless there are three adjacent data points which happen to be collinear, piecewise linear functions like $F_1(x)$ will always exhibit a change in slope at each knot.

The use of piecewise second order polynomials offers a partial solution to the problem if we restrict the quadratic polynomials to the intervals between adjacent data points. That is, with $n+1$ data points, $[x_i, f(x_i)]$, $i = 0, 1, 2, \dots, n$ there will be n quadratic splines, one per interval. If you're wondering why each quadratic spline passes through only two consecutive data points even though we could fit a quadratic function across subintervals comprised of three data points, look at Figure 5.3.

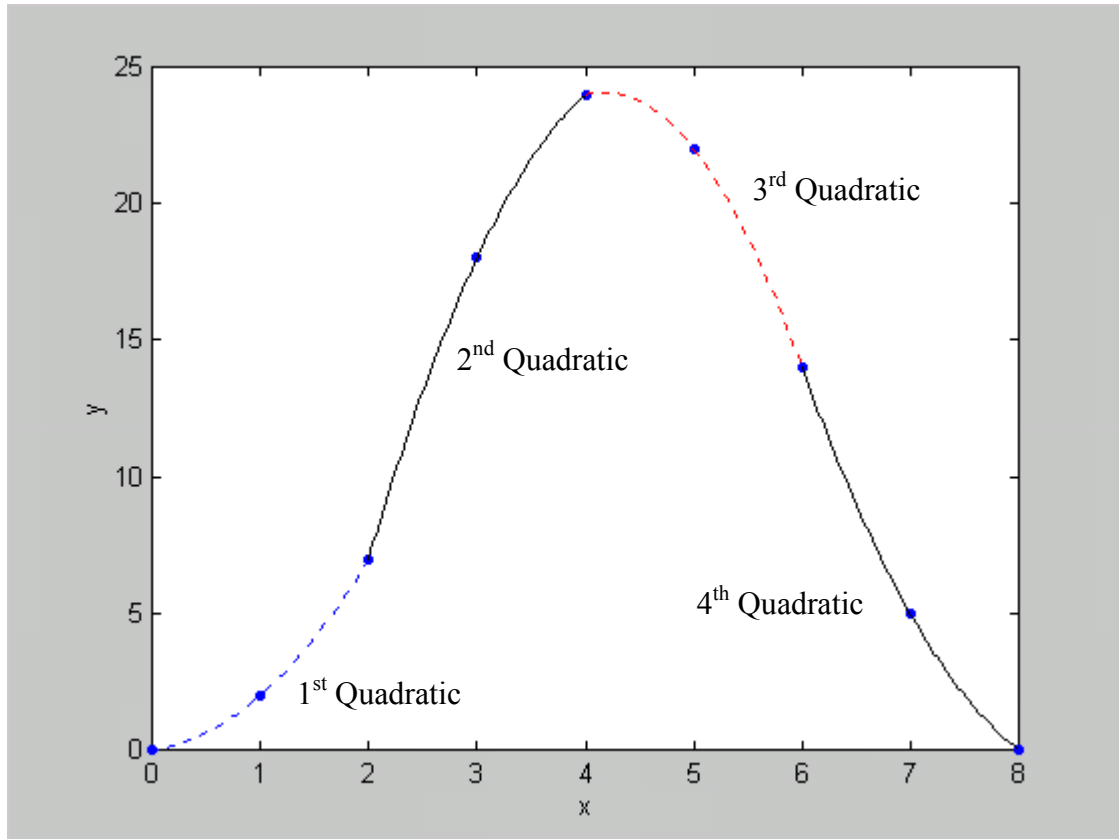


Figure 5.3 Piecewise Quadratic Polynomials Through Subintervals of Three Data Points

The composite function in Figure 5.3 consisting of quadratic polynomials fitted to subintervals of three data points each is not a smooth function. There is a discontinuity in slopes at the transition between the first and second quadratic as well as the second and third quadratics. Table 5.2 lists the four quadratic polynomials and their first derivatives. The MATLAB 'polyfit' function was used to obtain the quadratic polynomial coefficients

Interval	Quadratic	First Derivative
$0 \leq x \leq 2$	$1.5x^2 + 0.5x$	$3x + 0.5$
$2 \leq x \leq 4$	$-2.5x^2 + 23.5x - 30$	$-5x + 23.5$
$4 \leq x \leq 6$	$-3x^2 + 25x - 28$	$-6x + 25$

$6 \leq x \leq 8$	$2x^2 - 35x + 153$	$4x - 35$
-------------------	--------------------	-----------

Table 5.2 Quadratic Functions in Figure 5.3 and Their First Derivatives

Figure 5.4 is a graph of the first derivatives showing the discontinuities in slope at $x = 2$ and $x = 4$.

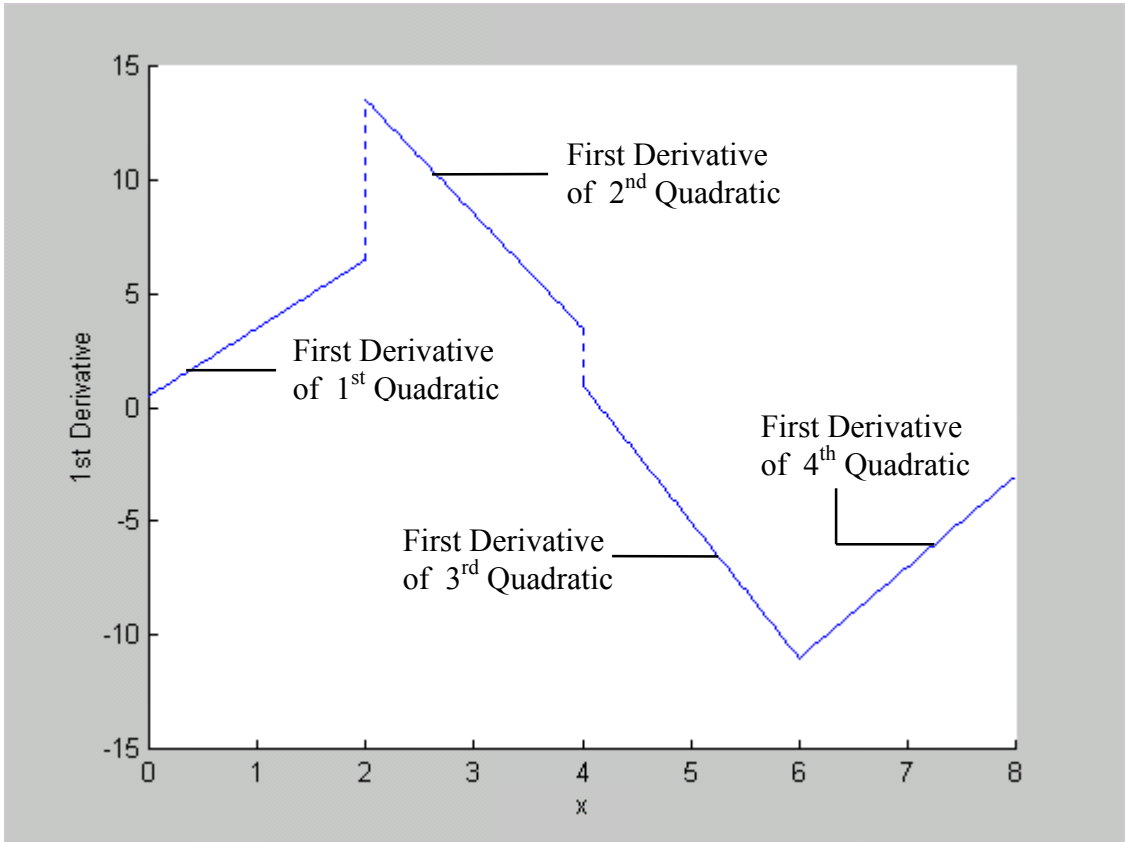


Figure 5.4 First Derivative of Composite Function Shown in Figure 5.3

Suppose we constrain each quadratic polynomial to a single interval between adjacent data points. There are three coefficients to be specified for each quadratic, yet only two constraints, namely that the quadratic pass through both end points. To assure a reasonably smooth fit to the data points, the slopes of each interior quadratic should be continuous at both end points as well, resulting in two more constraints. It would appear that the problem of determining each quadratic spline is now over constrained by virtue of there being four constraints (at least for the interior splines) with only three degrees of freedom represented by the three polynomial coefficients.

The situation is illustrated in Figure 5.5 where quadratic splines are required over four intervals defined by five data points. There are a total of 12 coefficients (3 for each of 4 quadratic polynomials) to be determined; hence there must be exactly 12 independent constraints to generate 12 linearly independent equations. There are 8 constraining equations to assure the 4 quadratic functions assume the correct values at the 2 endpoints

of each interval. Each of the 3 interior points produces only a single smoothness constraint, equality of the first derivatives of adjacent quadratic splines, increasing the total number of constraints to 11. Consequently, the system of linear equations for determining the 12 coefficients is not over constrained. On the contrary, an additional constraint, i.e. equation is needed to produce a consistent system of equations with a unique solution for the 12 coefficients. The missing equation will be revealed in the general discussion of quadratic splines which follows, although you may have figured it out on your own if you looked carefully at the first quadratic spline.

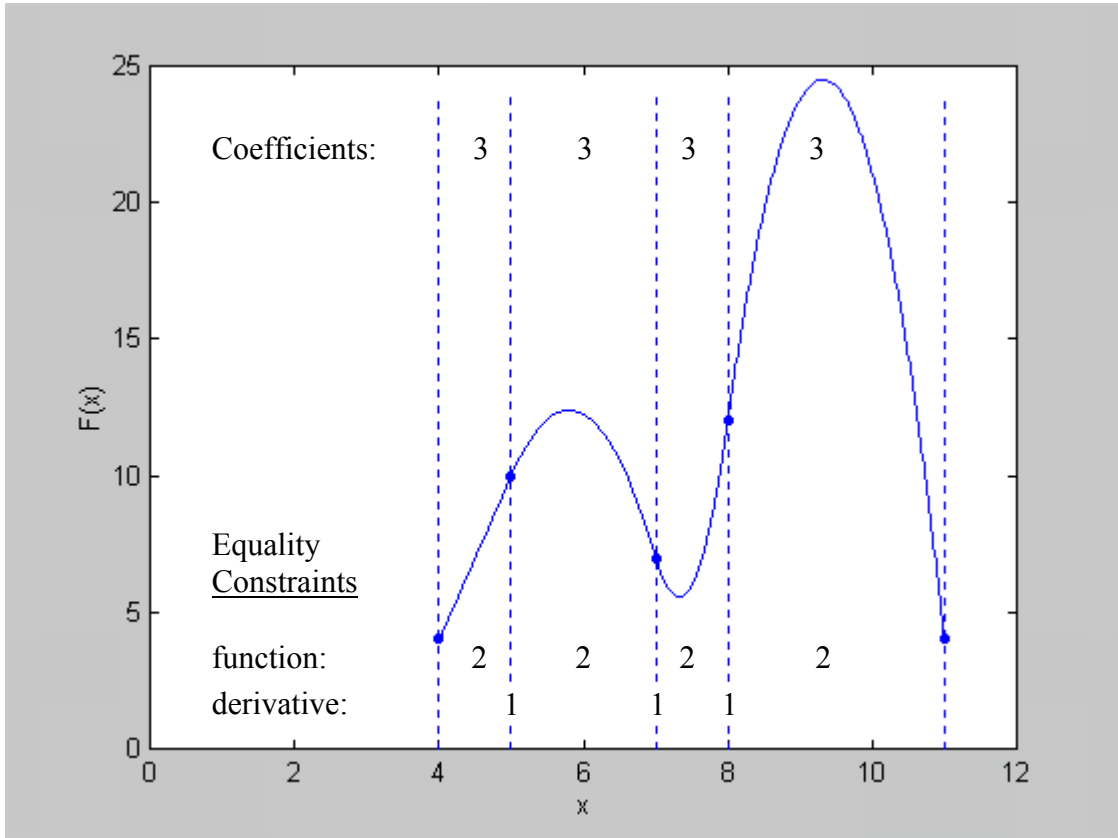


Figure 5.5 A Composite Function $F(x)$ of Piecewise Quadratic Splines

Given a set of data points $[x_i, f(x_i)]$, $i = 0, 1, 2, \dots, n$, there are n quadratic splines to be determined, one per interval. The splines may be represented in either the standard form, i.e. a linear combination of the monomials 1 , x , and x^2 as in Equation (2.7), or in the Newton divided-difference form of Equation (3.7). Since the smoothness constraint involves the first derivative, it is not advisable to employ the Lagrange form which is the most cumbersome to differentiate. Lagrange polynomials are better suited in applications where the constraints are based solely on the function values, as in Figure 5.3, where each piecewise quadratic was required to fit a subset of three data points.

Choosing the Newton divided-difference form, the function $F_2(x)$ comprised of quadratic splines is expressed as shown in Equation (5.4). Referring to Figure 5.6, there are 3 coefficients to be determined for each of n splines, hence a total of $3n$ coefficients.

Equations to determine these coefficients are obtained by imposing a like number of constraints on the splines.

$$F_2(x) = \begin{cases} a_1 + b_1(x-x_0) + c_1(x-x_0)(x-x_1), & x_0 \leq x \leq x_1 \\ a_2 + b_2(x-x_1) + c_2(x-x_1)(x-x_2), & x_1 \leq x \leq x_2 \\ \vdots & \vdots \\ a_{i-1} + b_{i-1}(x-x_{i-2}) + c_{i-1}(x-x_{i-2})(x-x_{i-1}), & x_{i-2} \leq x \leq x_{i-1} \\ a_i + b_i(x-x_{i-1}) + c_i(x-x_{i-1})(x-x_i), & x_{i-1} \leq x \leq x_i \\ a_{i+1} + b_{i+1}(x-x_i) + c_{i+1}(x-x_i)(x-x_{i+1}), & x_i \leq x \leq x_{i+1} \\ \vdots & \vdots \\ a_{n-1} + b_{n-1}(x-x_{n-2}) + c_{n-1}(x-x_{n-2})(x-x_{n-1}), & x_{n-2} \leq x \leq x_{n-1} \\ a_n + b_n(x-x_{n-1}) + c_n(x-x_{n-1})(x-x_n), & x_{n-1} \leq x \leq x_n \end{cases} \quad (5.4)$$

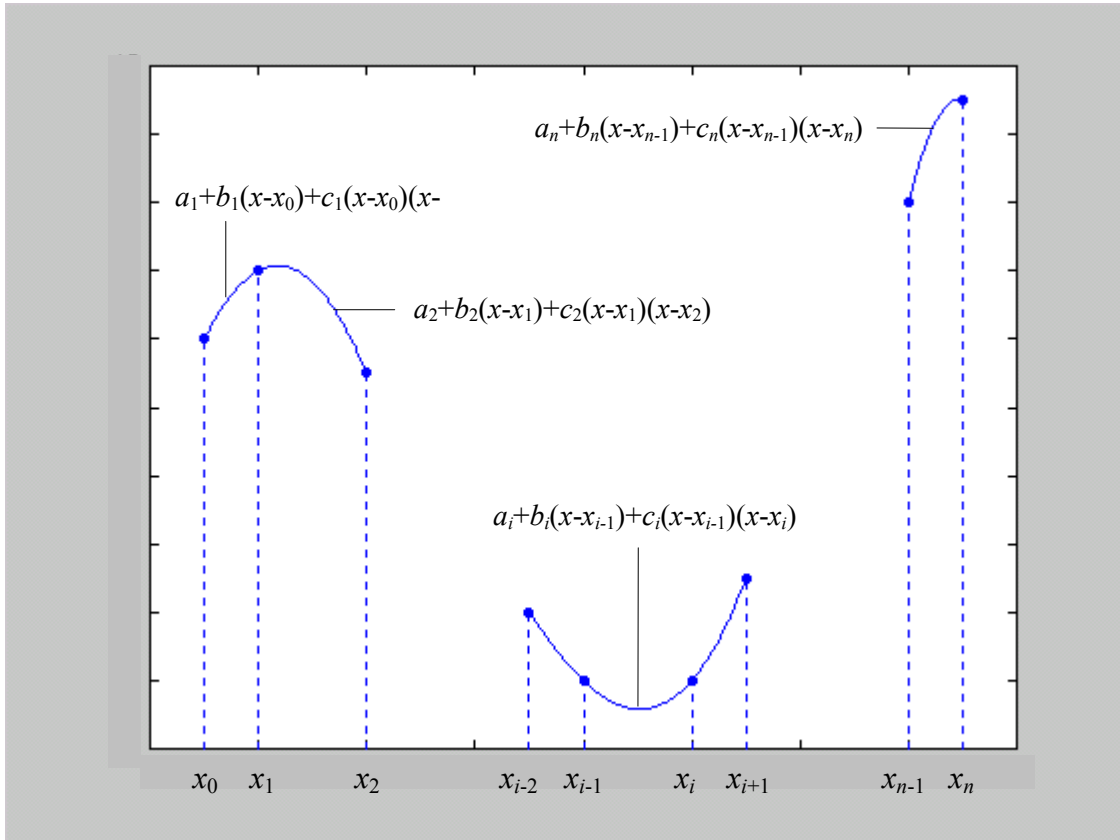


Figure 5.6 Quadratic Splines Represented in Newton Divided-Difference Form

The first set of equations constrain the quadratic splines to the data point function values at the end points of their respective intervals. This assures $F_2(x)$ will be a piecewise continuous function through the entire set of data points.

I. Interval End Point Function Values Must Equal Data Point Values:

$$F_2(x_i) = f(x_i), \quad i = 0, 1, 2, \dots, n \quad (5.5)$$

For the first interval,

$$F_2(x_0) = a_1 + b_1(x_0 - x_0) + c_1(x_0 - x_0)(x_0 - x_1) \quad (5.6)$$

$$f(x_0) = a_1 \quad (5.7)$$

$$F_2(x_1) = a_1 + b_1(x_1 - x_0) + c_1(x_1 - x_0)(x_1 - x_1) \quad (5.8)$$

$$f(x_1) = f(x_0) + b_1(x_1 - x_0) \quad (5.9)$$

$$b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \quad (5.10)$$

Evaluating the second quadratic spline at the end points of its interval gives

$$F_2(x_1) = a_2 + b_2(x_1 - x_1) + c_2(x_1 - x_1)(x_1 - x_2) \quad (5.11)$$

$$f(x_1) = a_2 \quad (5.12)$$

$$F_2(x_2) = a_2 + b_2(x_2 - x_1) + c_2(x_2 - x_1)(x_2 - x_2) \quad (5.13)$$

$$f(x_2) = f(x_1) + b_2(x_2 - x_1) \quad (5.14)$$

$$b_2 = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \quad (5.15)$$

Similar constraints apply to the remaining splines. The general form of the equations are given below.

$$F_2(x_{i-1}) = a_i + b_i(x_{i-1} - x_{i-1}) + c_i(x_{i-1} - x_{i-1})(x_{i-1} - x_i) \quad (5.16)$$

$$a_i = f(x_{i-1}), \quad i = 1, 2, 3, \dots, n \quad (5.17)$$

$$F_2(x_i) = a_i + b_i(x_i - x_{i-1}) + c_i(x_i - x_{i-1})(x_i - x_i) \quad (5.18)$$

$$f(x_i) = f(x_{i-1}) + b_i(x_i - x_{i-1}) \quad (5.19)$$

$$b_i = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \quad i = 1, 2, 3, \dots, n \quad (5.20)$$

Equations (5.17) and (5.20) are solutions for a_i and b_i of each spline, accounting for $2n$ out of the $3n$ coefficients. Thus, n equations involving c_i , $i = 1, 2, 3, \dots, n$ are required before we can uniquely determine the set of quadratic splines.

We now impose the smoothness constraint at the knots (interior data points) which produces $n-1$ new equations.

II. Continuity of First Derivative at Interior Data Points:

$$\lim_{x \rightarrow x_i^-} \frac{d}{dx} F_2(x) = \lim_{x \rightarrow x_i^+} \frac{d}{dx} F_2(x), \quad i = 1, 2, 3, \dots, n-1 \quad (5.21)$$

At the first interior point,

$$\frac{d}{dx} [a_1 + b_1(x - x_0) + c_1(x - x_0)(x - x_1)] \Big|_{x=x_1} = \frac{d}{dx} [a_2 + b_2(x - x_1) + c_2(x - x_1)(x - x_2)] \Big|_{x=x_1} \quad (5.22)$$

$$b_1 + c_1(2x - x_0 - x_1) \Big|_{x=x_1} = b_2 + c_2(2x - x_1 - x_2) \Big|_{x=x_1} \quad (5.23)$$

$$b_1 + c_1(x_1 - x_0) = b_2 + c_2(x_1 - x_2) \quad (5.24)$$

$$c_2 = \left(\frac{x_1 - x_0}{x_1 - x_2} \right) c_1 + \frac{b_1 - b_2}{x_1 - x_2} \quad (5.25)$$

A similar condition applies at each of the $n-1$ knots. That is,

$$c_{i+1} = \left(\frac{x_i - x_{i-1}}{x_i - x_{i+1}} \right) c_i + \frac{b_i - b_{i+1}}{x_i - x_{i+1}}, \quad i = 1, 2, 3, \dots, n-1 \quad (5.26)$$

Equation (5.26) represents a system of $n-1$ equations in n unknowns once the coefficients b_i are computed. Since there is no other obvious condition involving one or more c_i to impose, we are left with a single degree of freedom to choose one of the coefficients c_i arbitrarily. It is customary to let either c_1 or c_n be zero, meaning either the first or last spline will be a linear rather than quadratic function. If we choose $c_1=0$, Equation (5.26) can be solved sequentially for c_i , $i = 2, 3, \dots, c_n$. An example is now given to demonstrate the relative ease in finding the quadratic splines corresponding to a set of data points.

Example 5.2

A highway engineer must design a new road which will connect two existing roads. In order to maintain safe clearances from a lake and several structures, measurements were taken to determine several points along the centerline of the proposed road. The geographical locations of these points, relative to some point of reference, are given in Table 5.3. Quadratic splines are to be used to specify the equations of the centerlines for each road segment.

i	x_i (miles)	y_i (miles)
0	0.0	1.0
1	0.2	1.0
2	0.6	1.2
3	1.0	1.1
4	1.4	0.6
5	1.6	0.5
6	1.8	0.5
7	2.0	0.5

Table 5.3 End Point Data for Segments of Proposed Roadway

The following MATLAB m-file was written to plot the given centerline points, determine the quadratic splines, and plot the continuous centerline comprised of those splines. The results are illustrated in Figure 5.7 which shows the smooth centerline trajectory of the proposed connector between the existing roads. (Despite the smooth appearance of the connecting road, the curvature at certain points may not conform to existing civil engineering standards for roadway design.)

```
% Example 5.3
x=[0 0.2 0.6 1 1.4 1.6 1.8 2]; % Centerline
y=[1 1 1.2 1.1 0.6 0.5 0.5 0.5]; % Data Points
v=[0 2 0 1.6];
plot(x,y, '.') % Plot Centerline Data Points
xlabel('x (miles)')
ylabel('y (miles)')
axis(v)
hold on
axis manual
n=8;
% Find Newton Polynomial Coefficients
for i=1:n-1
    a(i)=y(i);
    b(i)=(y(i+1)-y(i))/(x(i+1)-x(i));
end
```

```

c(1)=0;
for i=2:n-1
    factor1=(x(i)-x(i-1))/(x(i)-x(i+1));
    factor2=(b(i-1)-b(i))/(x(i)-x(i+1));
    c(i)=factor1*c(i-1)+factor2;
end
% Plot Quadratic Splines
for i=1:n-1
    xi=linspace(x(i),x(i+1),100);
    f2=a(i)+b(i)*(xi-x(i))+c(i)*(xi-x(i)).*(xi-x(i+1));
    plot(xi,f2,'-')
end

```

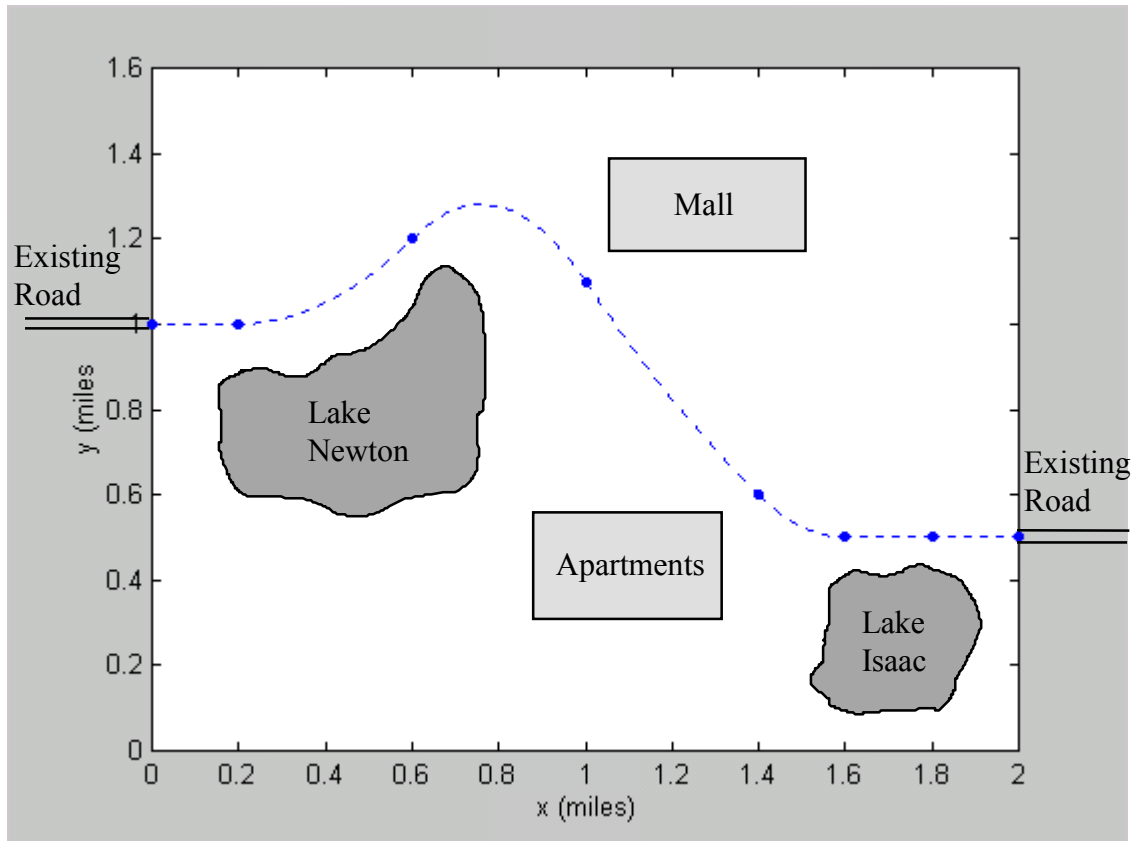


Figure 5.7 Roadway Centerline Consisting of Quadratic Splines

When the data points are equally spaced, i.e.

$$x_{i+1} - x_i = \Delta, \quad i = 0, 1, 2, \dots, n-1 \quad (5.27)$$

Equations (5.20) and (5.26) simplify to

$$b_i = \frac{f(x_i) - f(x_{i-1})}{\Delta} \quad i = 1, 2, 3, \dots, n \quad (5.28)$$

$$c_{i+1} = -c_i + \frac{1}{\Delta^2} [f(x_{i+1}) - 2f(x_i) + f(x_{i-1}))], \quad i = 1, 2, 3, \dots, n-1 \quad (5.29)$$

which is readily solved sequentially given the starting value for c_1 .

A composite function consisting of quadratic splines has a visually smooth appearance since the first derivative is continuous at the knots. The second derivative, however is not continuous at the knots. What are the implications of the second derivative being discontinuous at these points? To answer this question, we focus our discussion on the curvature of a function. In precise mathematical terms, curvature is the instantaneous rate of change of the direction of the tangent vector at a point on the curve with respect to the arc length. Intuitively, it makes sense that the faster the tangent vector is changing direction at a point, the greater will be the curvature of the function at the point.

From Calculus, the curvature κ at a point on the function $y = f(x)$ is given by

$$\kappa = \frac{|f''(x)|}{(1 + f'^2(x))^{3/2}} \quad (5.30)$$

Consider the simple polynomial function $y = f(x) = x^2$ shown in Figure 5.8. From Equation (5.30) the curvature of $f(x)$ at the origin is

$$\kappa = \frac{2}{(1 + (2x)^2)^{3/2}} \Big|_{x=0} = 2 \quad (5.31)$$

The circle of curvature is the circle with center (of curvature) along the normal to the curve and radius (of curvature) ρ equal to $1/\kappa$. The curvature properties of $f(x)$ are shown at several points along the function in the quadrants of Figure 5.8. Note the curvature of $f(x) = x^2$ is a maximum at the origin and diminishes as x increases.

Returning to our discussion of splines, it is evident that piecewise functions composed of quadratic splines may be smooth (based on the first derivative being continuous) at the knots but the curvature may change in a stepwise manner at these points. Equation (5.30) implies the change in curvature at a knot is proportional to the discontinuity of the second derivative since the first derivative and hence the denominator is continuous there.

There are indeed applications where sudden changes in the curvature of a smooth looking function are unacceptable. Jarring forces can result on a roller coaster ride when the radius of curvature changes abruptly. Highways consist of sections of road where a smooth transition requires a continuous curvature.

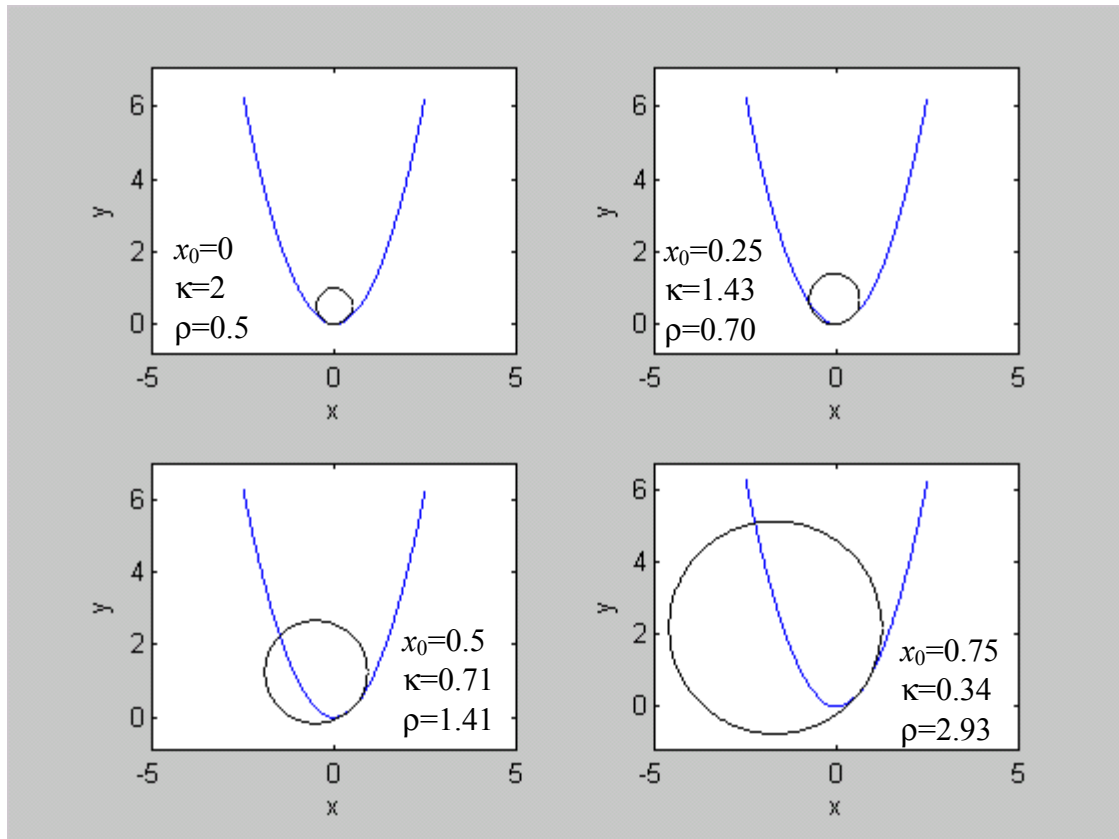


Figure 5.8 Curvature and Radius of Curvature of a Function at Several Points

The MATLAB script file used to determine as well as plot the circle of curvature is listed below.

```

for i=1:4
x=linspace(-2.5,2.5,500);
y=x.*x;
v=[-5 5 0 10];
subplot(2,2,i)
plot(x,y,'-')
xlabel('x'),ylabel('y')
axis(v)
hold on
axis square
axis equal
x0=(i-1)/4; % x value of f(x)where curvature is determined
xc=-4*x0^3; % x coord for center of circle of curvature
yc=0.5+3*x0^2; % y coord for center of circle of curvature
r=0.5*(1+4*x0^2)^(1.5); % radius of curvature of f(x) at x0
x=linspace(xc-r,xc+r,200);
disc=(r^2-(x-xc).^2).^0.5;
ycir_upper=yc+disc; % upper half of circle of curvature
ycir_lower=yc-disc; % lower half of circle of curvature
plot(x,ycir_upper,'g-')
plot(x,ycir_lower,'g-')
end

```

Cubic splines are used when the composite function is required to exhibit minimum curvature. Both first and second derivatives will be continuous at the interior data points. When there are $n+1$ data points (x_i, y_i) , $i = 0, 1, 2, \dots, n$ the composite function $F(x)$ is expressed as

$$F(x) = f_i(x), \quad x_{i-1} \leq x \leq x_i \quad i = 1, 2, 3, \dots, n \quad (5.32)$$

where each $f_i(x)$ is a cubic spline in the interval (x_i, x_{i-1}) . Expressing $f_i(x)$ in standard form,

$$f_i(x) = a_i + b_i x + c_i x^2 + d_i x^3, \quad i = 1, 2, 3, \dots, n \quad (5.33)$$

There are 4 coefficients for each cubic spline, hence a total of $4n$ coefficients to completely define $F(x)$. Each spline is constrained at its endpoints, i.e.

$$\left. \begin{array}{l} f_i(x_{i-1}) = y_{i-1} \\ f_i(x_i) = y_i \end{array} \right\} \quad i = 1, 2, 3, \dots, n \quad (5.34)$$

resulting in $2n$ equations involving the $4n$ coefficients. The first and second derivatives are continuous at the $n-1$ interior data points,

$$f'_{i-1}(x_{i-1}) = f'_i(x_{i-1}) \quad i = 2, 3, \dots, n \quad (5.35)$$

$$f''_{i-1}(x_{i-1}) = f''_i(x_i) \quad i = 2, 3, \dots, n \quad (5.36)$$

generating an additional $2(n-1)$ equations. At this point there are $4n-2$ equations, so two more are required to uniquely determine the coefficients of each cubic spline. Since the system is underdetermined, we must assign numerical values for two of the coefficients. This is analogous to the situation involving quadratic splines where a single coefficient was arbitrarily assigned a value of zero resulting in a linear spline at one of the ends. More will be said about this later on.

The procedure outlined above for determining the cubic splines is not the most efficient due to the necessity of having to solve a system of $4n$ simultaneous equations. An alternate method is now presented which ultimately results in far fewer as well as simpler equations to solve for the cubic splines.

Figure 5.9 shows three interior cubic splines of a composite function $F(x)$. First and second derivatives are also shown. Differentiating the cubic splines twice generates linear functions for the second derivatives in each interval. A Newton divided-difference representation for the second derivative $f''_i(x)$ in the interval $[x_{i-1}, x_i]$ is

$$f_i''(x) = A_{i-1} + (x - x_{i-1}) \frac{A_i - A_{i-1}}{x_i - x_{i-1}} \quad (5.37)$$

where A_{i-1} and A_i are the numerical values, yet to be determined, of the second derivative at x_{i-1} and x_i respectively, (See Figure 5.9).

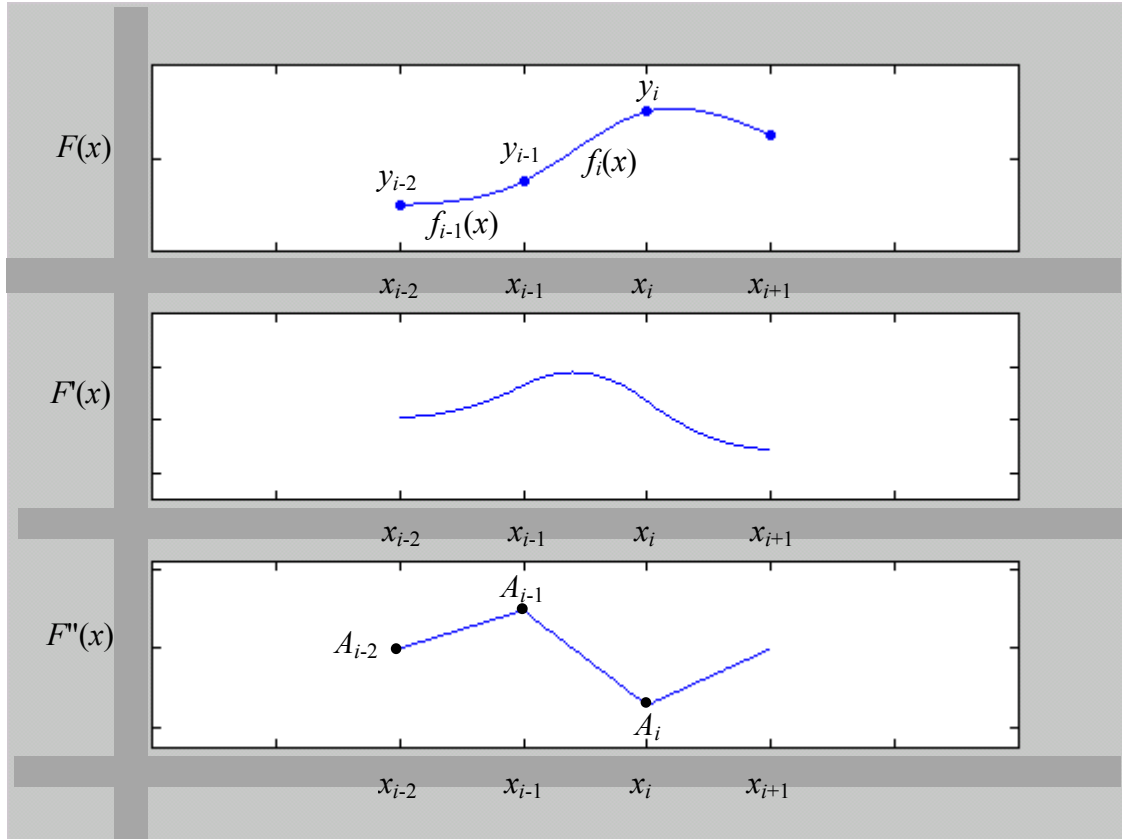


Figure 5.9 Cubic Splines and Their First Two Derivatives

Integrating the second derivative in Equation (5.37) twice, yields expressions for $f_i'(x)$ and $f_i(x)$ with constants of integration present.

$$f_i'(x) = \left(\frac{A_i - A_{i-1}}{x_i - x_{i-1}} \right) \frac{(x - x_{i-1})^2}{2} + A_{i-1}(x - x_{i-1}) + \alpha_i \quad (5.38)$$

$$f_i(x) = \left(\frac{A_i - A_{i-1}}{x_i - x_{i-1}} \right) \frac{(x - x_{i-1})^3}{6} + A_{i-1} \frac{(x - x_{i-1})^2}{2} + \alpha_i (x - x_{i-1}) + \beta_i \quad (5.39)$$

The constants of integration are determined from the endpoint constraints of Equation (5.34). The result is

$$\beta_i = y_{i-1}, \quad i = 1, 2, 3, \dots, n \quad (5.40)$$

$$\alpha_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} - \left(\frac{A_i + 2A_{i-1}}{6} \right) (x_i - x_{i-1}), \quad i = 1, 2, 3, \dots, n \quad (5.41)$$

Substituting Equations (5.40) and (5.41) into Equation (5.39) yields expressions for the cubic splines which pass through the data points (x_i, y_i) , $i = 0, 1, 2, \dots, n$.

$$\begin{aligned} f_i(x) = & \left[\frac{A_i - A_{i-1}}{6(x_i - x_{i-1})} \right] (x - x_{i-1})^3 + \frac{A_{i-1}}{2} (x - x_{i-1})^2 \\ & + \left[\frac{y_i - y_{i-1}}{x_i - x_{i-1}} - \left(\frac{A_i + 2A_{i-1}}{6} \right) (x_i - x_{i-1}) \right] (x - x_{i-1}) + y_{i-1}, \quad i = 1, 2, 3, \dots, n \end{aligned} \quad (5.42)$$

At first glance it appears that we may have complicated things unnecessarily due to the presence of the second derivative terms A_i and A_{i-1} in Equation (5.42). On the positive side, however, we are not confronted with a system of $4n$ simultaneous equations as would be the case if we employed the standard polynomial form to represent the cubic splines. In fact, all that remains is a way of determining the constants A_i , $i = 0, 1, 2, \dots, n$.

Equations for these constants are obtained by imposing the continuity requirement of the first derivative at the knots, Equation (5.35). Note, continuity of the second derivative at each knot is assured by virtue of Equation (5.37). The process is straightforward but tedious due to the manipulation of numerous terms appearing in the expressions for the first derivatives $f'_{i-1}(x_{i-1})$ and $f'_i(x_{i-1})$. After much simplification, the result is

$$\left(\frac{x_i - x_{i-1}}{6} \right) A_i + \left(\frac{x_i - x_{i-2}}{3} \right) A_{i-1} + \left(\frac{x_{i-1} - x_{i-2}}{6} \right) A_{i-2} = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} - \frac{y_{i-1} - y_{i-2}}{x_{i-1} - x_{i-2}} \quad (5.43)$$

Equation (5.43) applies for $i = 2, 3, \dots, n$ which accounts for $n-1$ equations relating the $n+1$ constants A_i , $i = 0, 1, 2, \dots, n$. It is customary to select or compute numerical values for A_0 and A_n based on additional information or requirements imposed on the endmost splines. For example, choosing $A_0 = A_n = 0$, produces what is referred to as a "natural" cubic spline. In this case, the spline is relaxed at both ends approaching straight lines at the endpoints. A "clamped" cubic spline results when the endpoint constants A_0 and A_n are calculated based on known values for the first derivatives at the endpoints, i.e. $f'_1(x_0)$ and $f'_n(x_n)$. Alternatively, A_0 and A_n can be extrapolated values based on A_1, A_2 and A_{n-2}, A_{n-1} respectively. Finally, selecting $A_0 = A_1$ and $A_n = A_{n-1}$ forces the two end splines to reduce to quadratics (see Equation (5.37)).

Composite functions made up of cubic splines are reasonably smooth in appearance because the curvature of the individual splines connecting the data points is moderate in comparison to that of a single high order polynomial defined over the entire range of data points. Recall from Equation (5.30) that the curvature of a function is directly related to the second derivative. Clearly, large second derivatives imply the first derivative is changing rapidly and therefore the slope of the tangent vector is doing likewise, resulting in significant curvature. In other words, given the choice of two interpolating functions, the one with the smaller (in some average sense) second derivative is preferable. It can be shown that of all possible interpolating functions which are twice differentiable, the natural cubic spline has the least curvature in the mean square sense. In mathematical terms, the integral $\int_{x_0}^{x_n} [g''(x)]^2 dx$ is a minimum when the integrand $g''(x)$ is a natural cubic spline.

Example 5.3

In Example 5.2, a company is planning on bidding for the contract to build the proposed road. In order to estimate the cost, it must know the length of road to be constructed. Cubic splines are to be used to describe the road geometry and calculate its length.

An m-file was written to produce a natural spline fit through the data points from Table 5.3. From Equation (5.43), a 6×6 coefficient matrix \mathbf{X} and 6×1 constant vector \mathbf{b} are created. The 6×1 vector \mathbf{a} of second derivatives is the solution to the system of equations $\mathbf{X}\mathbf{a} = \mathbf{b}$ where

$$\mathbf{X} = \begin{bmatrix} \frac{1}{5} & \frac{1}{15} & 0 & 0 & 0 & 0 \\ \frac{1}{15} & \frac{4}{15} & \frac{1}{15} & 0 & 0 & 0 \\ 0 & \frac{1}{15} & \frac{4}{15} & \frac{1}{15} & 0 & 0 \\ 0 & 0 & \frac{1}{15} & \frac{1}{5} & \frac{1}{30} & 0 \\ 0 & 0 & 0 & \frac{1}{30} & \frac{2}{15} & \frac{1}{30} \\ 0 & 0 & 0 & 0 & \frac{1}{15} & \frac{2}{15} \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} \frac{1}{2} \\ -\frac{3}{4} \\ -1 \\ \frac{3}{4} \\ \frac{1}{2} \\ 0 \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \end{bmatrix}$$

The cubic splines are then determined from Equation (5.42) and plotted in Figure 5.10 along with the data points. For comparison, the road using quadratic splines determined in Example 5.2 and plotted in Figure 5.7 is redrawn. The m-file listing follows.

```
% Example 5.3 and Figure 5.10
% A natural cubic spline is fit to the data points defined
% by vectors x and y. A0 and An in Equation (5.43) are both zero.
x=[0 0.2 0.6 1 1.4 1.6 1.8 2];
y=[1 1 1.2 1.1 0.6 0.5 0.5 0.5];
v=[0 2 0 1.6];
```

```

plot(x,y,'-.')
axis(v)
axis manual
hold on
n=7; % Number of data points
X=zeros(n-1,n-1); % X is (n-1) by (n-1)
%Create first row of X matrix and b vector
X(1,1)=(x(3)-x(1))/3;
X(1,2)=(x(3)-x(2))/6;
b(1)=(y(3)-y(2))/(x(3)-x(2)) - (y(2)-y(1))/(x(2)-x(1));
% Create rows 2 thru n-2 of X matrix and b vector
for i=2:n-2
    X(i,i-1)=(x(i+1)-x(i))/6;
    X(i,i)=(x(i+2)-x(i))/3;
    X(i,i+1)=(x(i+2)-x(i+1))/6;
    b(i)=(y(i+2)-y(i+1))/(x(i+2)-x(i+1)) - (y(i+1)-y(i))/(x(i+1)-x(i));
end
% Create (n-1)st row of X matrix and b vector
X(n-1,n-2)=(x(n)-x(n-2))/6;
X(n-1,n-1)=(x(n+1)-x(n-1))/3;
b(n-1)=(y(n+1)-y(n))/(x(n+1)-x(n)) - (y(n)-y(n-1))/(x(n)-x(n-1));
a=inv(X)*b'; % Find a=[A(1)...A(n-1)]
for i=1:n
    dx=x(i+1)-x(i);
    dy=y(i+1)-y(i);
    xspace=linspace(x(i),x(i+1),75);
    xx=xspace-x(i);
    if (i==1)
        T1=(1/6)*(a(1)/dx)*xx.^3;
        T3=((dy/dx)-(a(1)/6)*dx)*xx + y(1);
        f_1=T1+T3;
        plot(xspace,f_1,'-') % Plot 1st cubic spline
    elseif(i~=1&i~=n)
        T1=(1/6)*((a(i)-a(i-1))/dx)*xx.^3;
        T2=(a(i-1)/2)*xx.^2;
        T3=((dy/dx)-((a(i)+2*a(i-1))/6)*dx)*xx + y(i);
        f_i=T1+T2+T3;
        plot(xspace,f_i,'-') % Plot 2nd thru (n-1)st cubic spline
    elseif(i==n)
        T1=(1/6)*(-a(n-1)/dx)*xx.^3;
        T2=(a(i-1)/2)*xx.^2;
        T3=((dy/dx)-(a(n-1)/3)*dx)*xx + y(n);
        f_n=T1+T2+T3;
        plot(xspace,f_n,'-') % Plot nth cubic spline
    end
end
end

```

As you might expect MATLAB has its own built-in routine for cubic splines. The statement $y_i = \text{spline}(x, y, x_i)$ invokes the cubic spline function where arrays x and y define the data points while (x_i, y_i) are the smoothed point(s) in between them. Instead of a natural spline, MATLAB constrains the third derivatives of the first two and last two cubic splines to be equal. The constants A_0 and A_n are not zero; however the vector $\mathbf{a} = [A_0 A_1 \dots A_n]^T$ is uniquely determined. The 'spline' function exploits the sparse nature of the coefficient matrix obtained from Equation (5.43) and is therefore more efficient than matrix inversion to solve the equations.

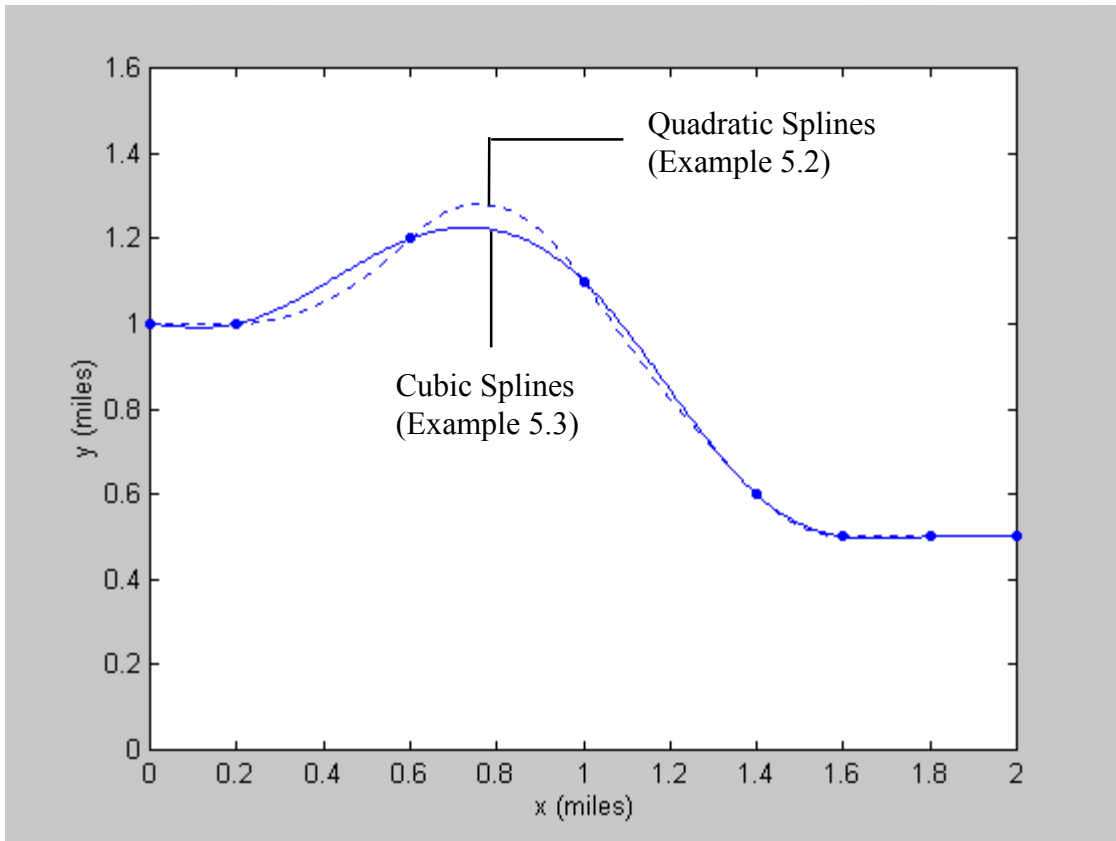


Figure 5.10 Roadway Centerline Determined From Quadratic and Cubic Splines

If you're curious about the coefficients for each cubic spline, you can call 'spline' with arguments x and y only and it returns a 'pp' or piecewise polynomial array form which can be decomposed using 'unmkpp' to reveal the polynomial coefficients and other useful information. To illustrate, suppose we use the data points from Example 5.3.

```

»x=[0 0.2 0.6 1 1.4 1.6 1.8 2];
»y=[1 1 1.2 1.1 0.6 0.5 0.5 0.5];
pp=spline(x,y);
[x_values,spline_coefs,num_splines,num_coefs]=unmkpp(pp)

```

The following output results.

```

x _values = 0 0.2000 0.6000 1.0000 1.4000 1.6000 1.8000 2.0000

spline_coefs = -2.0032 2.4359 -0.4070 1.0000
               -2.0032 1.2340 0.3269 1.0000
               -0.8416 -1.1698 0.3526 1.2000
               3.8069 -2.1797 -0.9872 1.1000
               -1.8512 2.3886 -0.9037 0.6000
               -2.1298 1.2779 -0.1704 0.5000
               -2.1298 0.0000 0.0852 0.5000

num_splines = 7

```

```
num_coefs = 4
```

Note, the 'pp' array can be used multiple times for interpolation using 'ppval' without having to recompute the cubic spline coefficients, in the same way the array resulting from a 'polyfit' call can be used repeatedly with 'polyval'. For example, if we need the second cubic spline evaluated at 10 equally spaced points or the composite function evaluated at 20 equally spaced points, we could write

```
»yi=ppval(pp,linspace(0.2,0.6,10))
»yi=ppval(pp,linspace(0,2,20))

yi=1.0000 1.0168 1.0374 1.0608 1.0859 1.1116 1.1369 1.1608 1.1822 1.2000

yi=1.0000 0.9818 1.0036 1.0513 1.1109 1.1685 1.2099 1.2242 1.2048 1.1460
    1.0426 0.9048 0.7586 0.6308 0.5456 0.5042 0.4934 0.4991 0.5063 0.5000
```

We have yet to calculate the road length as required in Example 5.3. Since 'pp' describes the entire composite function of cubic splines, its ideal for computing the length of any section or the entire length of the roadway. A simple procedure for estimating a length of section from $x = a$ to $x = b$ is now presented.

Imagine walking along the section from $x = a$ to $x = b$ such that each step is the same length in the x -direction, i.e. $\Delta x = x_{i+1} - x_i = x_i - x_{i-1}$. For each step from x_{i-1} to x_i , an incremental section of road Δs_i is traversed as shown in Figure 5.11.

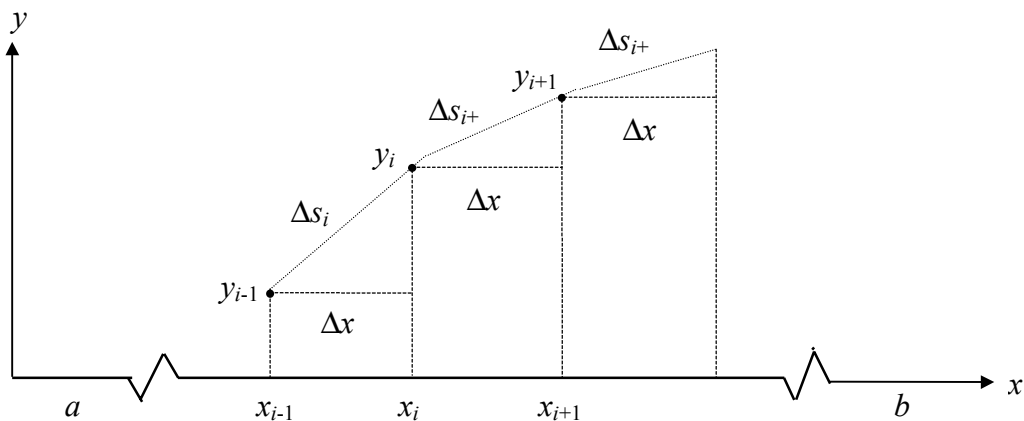


Figure 5.11 Road Length Calculation Using Summation of Incremental Steps

The length of the road can be approximated by summing the incremental steps, i.e.

$$S \approx \sum_i \Delta s_i \quad (5.44)$$

$$S \approx \sum_i [(\Delta x)^2 + (\Delta y_i)^2]^{1/2} \quad (5.45)$$

Note how the 'pp' array is used with the function 'ppval' in the following m-file listing to find the y coordinates of each step. Since $a = 0$ and $b = 2$ correspond to the endpoints, the value for "length" represents the length of the entire road. Also note, the file was executed twice with different values of "n" to ensure the accuracy of the answer.

```
%Road Length Calculation
x=[0 0.2 0.6 1 1.4 1.6 1.8 2];
y=[1 1 1.2 1.1 0.6 0.5 0.5 0.5];
pp=spline(x,y);
a=0;
b=2;
n=1000 % Number of points. There are n-1 steps.
xi=linspace(a,b,n);
dx=xi(2)-xi(1);
yi=ppval(pp,xi);
s=0;
for i=1:n-1
    dyi=yi(i+1)-yi(i);
    dsi=(dx*dx + dyi.*dyi).^0.5;
    s=s+dsi;
end
length=s

n = 100
length = 2.3597

n = 1000
length = 2.3599
```

You probably recognize Equation (5.45) as a step along the way to obtaining an integral expression. Further discussion of integration and approximation methods is deferred until the chapter on Numerical Integration.

The next example illustrates the use of splines and low order interpolating polynomials as a way of representing the thermodynamic properties of a substance.

Example 5.4

The refrigerant in refrigeration system circulates in a closed system existing in different states with specific thermodynamic properties. The refrigeration cycle is best understood when viewed on a diagram which portrays the refrigerant's properties of pressure (P) and enthalpy (h) in the liquid and vapor states. The table below contains the properties of Refrigerant 22 in its liquid and vapor states. (Source: Refrigerating and Air-Conditioning, ARI Institute, Prentice-Hall)

Our first step will be to draw a portion of the P vs h chart, also called a Mollier diagram, for R-22 from the data in Table 5.4.

T , (°F)	P , (psia)	h_l (Btu/lb)	h_v (Btu/lb)	T , (°F)	P , (psia)	h_l (Btu/lb)	h_v (Btu/lb)
20	57.727	15.837	106.383	80	158.33	33.109	111.052
25	63.450	17.219	106.839	85	170.38	34.626	111.345
30	69.591	18.609	107.284	90	183.09	36.158	111.619
35	76.170	20.010	107.719	95	196.50	37.704	111.873
40	83.206	21.422	108.142	100	210.60	39.267	112.105
45	90.719	22.843	108.553	105	225.45	40.847	112.314
50	98.727	24.275	108.953	110	241.04	42.446	112.498
55	107.25	25.718	109.339	115	257.42	44.065	112.655
60	116.31	27.172	109.712	120	274.60	45.705	112.782
65	125.93	28.638	110.070	125	292.62	47.369	112.877
70	136.12	30.116	110.414	130	311.50	49.059	112.936
75	146.91	31.606	110.741	135	331.26	50.778	112.956

Table 5.4 Thermodynamic Properties of Refrigerant 22

Two composite cubic splines are required, one for saturated liquid properties and the other for the saturated vapor. The m-file for producing a segment of the Mollier diagram from the data in Table 5.4 is listed below.

```

%Mollier Diagram for R-22 from Thermodynamic Properties
T=[20:5:135];
P=[57.727 63.450 69.591 76.170 83.206 90.719 98.727 107.25 116.31...
    125.93 136.12 146.91 158.33 170.38 183.09 196.50 210.60 225.45...
    241.04 257.42 274.60 292.62 311.50 331.26];
hl=[15.837 17.2129 18.609 20.010 21.422 22.843 24.275 25.718 27.172...
    28.638 30.116 31.606 33.109 34.626 36.158 37.704 39.267 40.847...
    42.446 44.065 45.705 47.369 49.059 50.778];
hv=[106.383 106.839 107.284 107.719 108.142 108.533 108.953 109.339...
    109.712 110.070 110.414 110.741 111.052 111.345 111.619 111.873...
    112.105 112.314 112.498 112.655 112.782 112.877 112.936 112.956];
hli=linspace(15.837,50.778,250);
pli=spline(hl,P,hli);
v=[0 140 0 350];
plot(hli,pli,'-')
xlabel('Enthalpy, h (Btu/lb)')
ylabel('Absolute Pressure, P (psia)')
axis(v)
hold on
axis manual
hvi=linspace(106.383,112.956,250);
pvi=spline(hv,P,hvi);
plot(hvi,pvi,'-')

```

The graph, shown in Figure 5.12, consists of the saturated liquid line and saturated vapor line. The cycle begins at pt. A with a saturated liquid at a high temperature and pressure leaving the condenser. After expanding to a low pressure, low temperature liquid and vapor mixture (pt. B) it flows through an evaporator where it completely evaporates, absorbing heat in the process, and enters the compressor (pt. C). It exits the compressor as a high pressure, superheated vapor (pt. D). It is cooled to a saturated vapor (pt. E) and then fully condensed to a saturated liquid (pt. A) in the condenser where the heat absorbed in the evaporator and the heat of compression is rejected. The heat absorbed in the evaporator from pt. B to pt. C occurs at a constant low pressure and low temperature. The heat rejected in the condenser from pt. D to pt. E to pt. A occurs at a constant high pressure with the temperature constant from pt. E to pt. A.

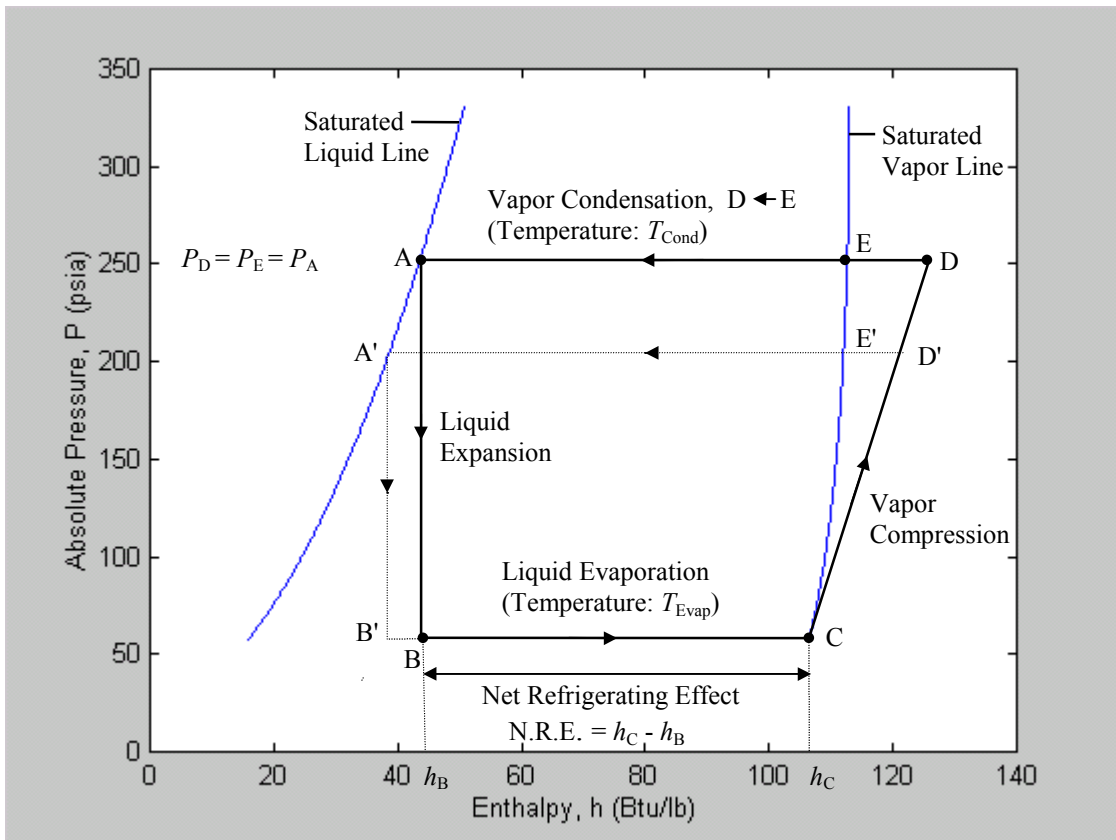


Figure 5.12 Spline Representation of R-22 Pressure-Enthalpy Diagram with Key Points of the Refrigeration Cycle Shown

The Net Refrigerating Effect (N.R.E.) is the difference in heat content of the refrigerant entering and leaving the evaporator. The N.R.E. is a measure of the heat (in Btu) absorbed from the conditioned space per lb of circulating refrigerant. It is given by $N.R.E. = h_C - h_B$.

The N.R.E. depends on two temperatures, the constant evaporator temperature, T_{Evap} and the condensing temperature T_{Cond} . For purposes of this example, the evaporator temperature T_{Evap} will be fixed at 25°F. Our objective is to investigate two relationships.

The first is between the condenser pressure P_{Cond} (constant from pt. E to pt. A) and the condensing temperature T_{Cond} (constant from pt. E to pt. A) which depends on the outside air or water (for water cooled systems) temperature. The second is how the N.R.E. varies with the condensing temperature T_{Cond} .

The exhaust pressure of the compressor P_D is the same as condenser pressure P_{Cond} and is determined by T_{Cond} , the condensing temperature. Therefore, changes in T_{Cond} result in the line D-E-A moving up and down (D'-E'-A') as shown in Figure 5.12. A graph of P_{Cond} vs. T_{Cond} is shown in Figure 5.13. It was obtained by fitting a second order polynomial through 3 data points (shown as asterisks) in Table 5.4 corresponding to condensing temperatures of 100°F, 115°F and 130°F. Additional data points are also shown indicating the high accuracy obtained using a second order interpolating polynomial. Figure 5.13 is significant because more electrical energy is required by the compressor to compress the refrigerant to a higher pressure.

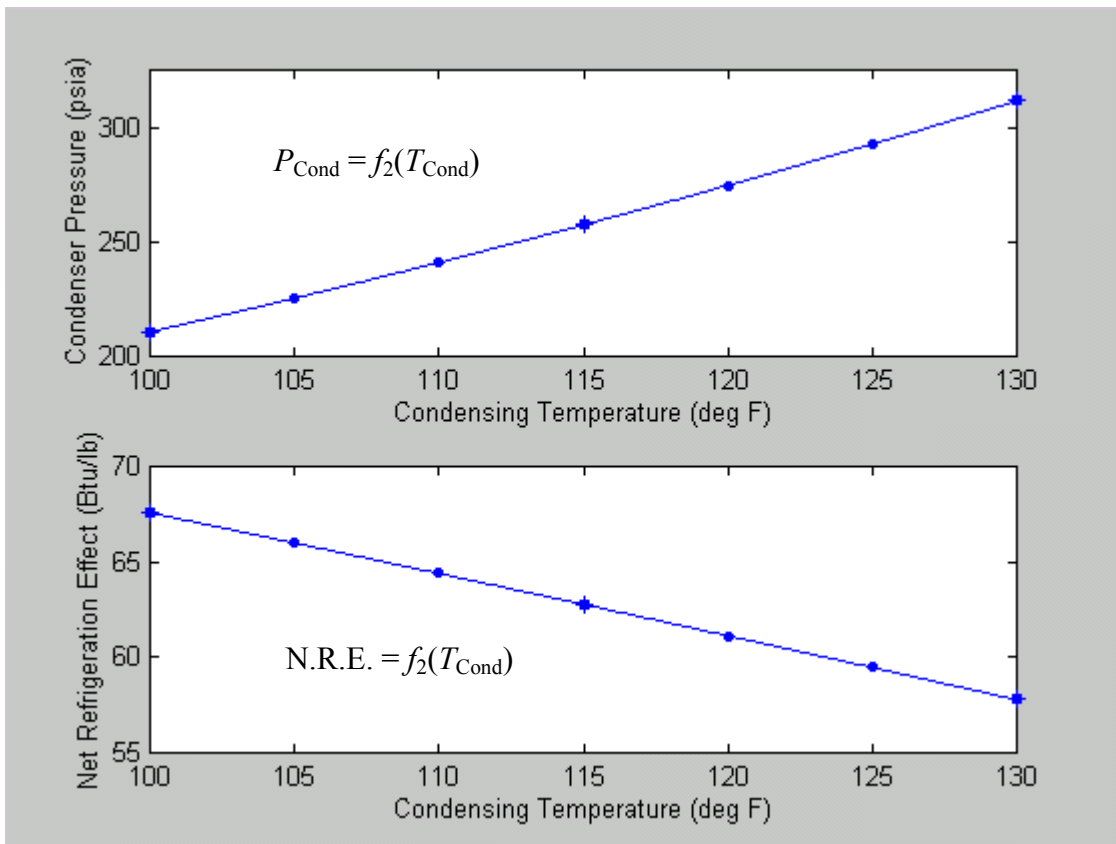


Figure 5.13 Interpolating Polynomials for Condenser Pressure and Net Refrigeration Effect in Terms of Condensing Temperature for R-22 refrigeration Cycle

The second relationship of interest, N.R.E. vs. T_{Cond} , demonstrates the influence of condensing temperature on the cooling capacity of the refrigerant. Referring to Figure 5.12 it can be seen that variations in T_{Cond} result in different paths from pt. D to pt. B because of the changes in condenser pressure. The N.R.E. is affected because the enthalpy of the refrigerant entering the evaporator, h_B is changing. For a constant

evaporator temperature $T_{\text{Evap}} = 25^\circ\text{F}$, the saturated vapor enthalpy $h_C = 106.839$ Btu/lb (see Table 5.4) and the N.R.E. reduces to

$$\text{N.R.E.}(T_{\text{Cond}}) = f(T_{\text{Cond}}) = 106.839 - h_B(T_{\text{Cond}}) \quad (5.46)$$

$$\text{N.R.E.}(T_{\text{Cond}}) = 106.839 - h_A(T_{\text{Cond}}) \quad \text{Btu / lb} \quad (5.47)$$

From Table 5.4, the saturated liquid enthalpy h_A varies from 39.267 Btu/lb when $T_{\text{Cond}} = 100^\circ\text{F}$ up to 49.059 Btu/lb when $T_{\text{Cond}} = 130^\circ\text{F}$. Using the 7 data points relating h_A and T_{Cond} from Table 5.4 in combination with Equation (5.47) gives the following table

$T_{\text{Cond}} (^\circ\text{F})$	h_A (Btu/lb)	N.R.E. (Btu/lb)
100	39.267	67.572
105	40.847	65.992
110	42.446	64.393
115	44.065	62.774
120	45.705	61.134
125	47.369	59.470
130	49.059	57.780

Table 5.5 Data Points for Generating Interpolating Polynomial $\text{N.R.E.} = f_2(T_{\text{Cond}})$

A second order interpolating polynomial $\text{N.R.E.} = f_2(T_{\text{Cond}})$ through the bold data points in Table 5.5 was obtained using MATLAB. It is plotted in the lower half of Figure 5.13 which includes all 7 data points and the bold ones with asterisks. The lower graph in Figure 5.13 could be used to determine the cooling capacity, in tons (1 ton = 12,000 Btu/hr) at various condensing temperatures if the refrigerant flow rate in lb/hr were known.

Strictly speaking, splines are more useful in graphics applications requiring smooth curve and surface designs than they are for pure interpolation to approximate function values. Figure 5.14 illustrates the use of splines to represent a simple graphic image. All the features except for the eyes are composed of composite splines through a set of digitized points obtained from the graphic. The sampled points along the spline describing the dinosaur's back are shown. There are a total of fourteen splines in all. All sharp corners where the slope is discontinuous, such as the tip of the tail, must be on the boundary of two (or more) splines. Choosing the best set of points for composing the splines involves a fair amount of trial and error.

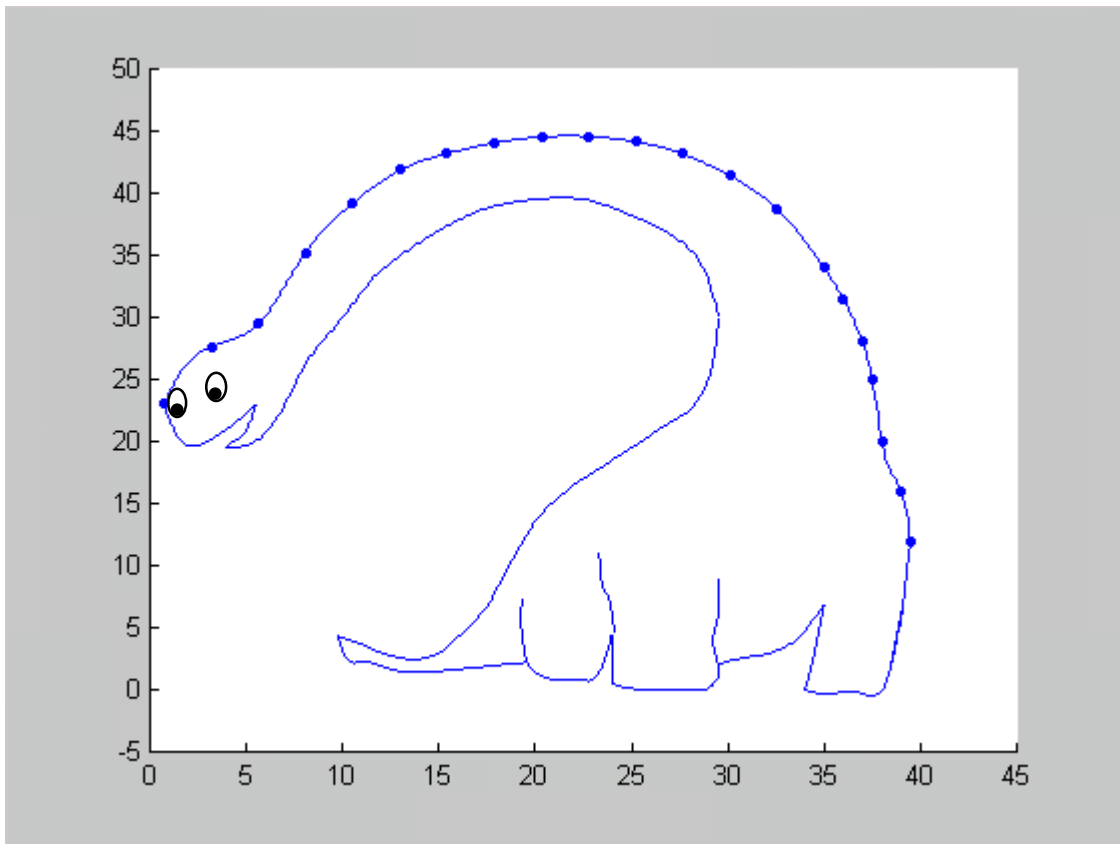


Figure 5.14 Creating a Graphic From a Digitized Image Using Spline Functions

Integration and differentiation of complex functions or functions only available in tabular form is often accomplished by approximating the function with a composite spline because of the relative ease in differentiating and integrating polynomials. Popular graphics and CAD software rely extensively on drawing splines in 2D and 3D applications. Animation is another area where splines are utilized to describe smooth paths for inanimate objects to follow. The Spline Toolbox is a compilation of routines for use with MATLAB to extend its capability for spline construction and manipulation.

Exercises

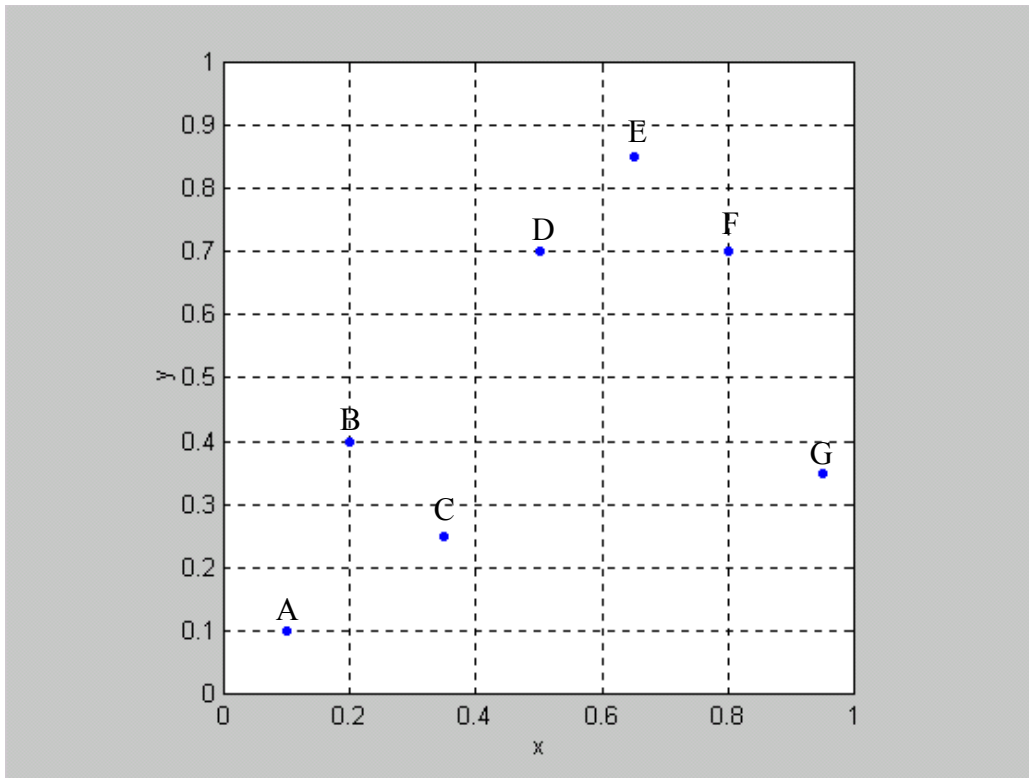
1. A quadratic spline is to be fit through the following points: (0,1), (1,2), (3,3), (4,2). The spline is given by

$$f(x) = \begin{cases} a_1 + b_1(x - x_0) & 0 \leq x \leq 1 \\ a_2 + b_2(x - x_1) + c_2(x - x_1)^2 & 1 \leq x \leq 3 \\ a_3 + b_3(x - x_2) + c_3(x - x_2)^2 & 3 \leq x \leq 4 \end{cases}$$

A system of 8 equations in 8 unknowns has been generated, i.e. $\mathbf{Ax} = \mathbf{b}$. The vector $\mathbf{x} = [a_1 \ b_1 \ a_2 \ b_2 \ c_2 \ a_3 \ b_3 \ c_3]^T$. Find the coefficient matrix \mathbf{A} and the vector \mathbf{x} .

2. The road in Examples 5.2 and 5.3 cost \$2.75 million per mile to construct. Find the difference in cost if the road is designed to follow a natural cubic spline instead of a quadratic spline.
3. Find the maximum curvature of each road (Example 5.2 and 5.3) and the x, y coordinates where the curvature is maximum.
4. Derive the additional equations to be used in conjunction with Equation (5.43) to produce a unique solution for the vector of second derivatives $\mathbf{a} = [A_0 \ A_1 \ \dots \ A_n]^T$ when the cubic spline $F(x)$ through the data points $(x_i, y_i), i = 0, 1, 2, \dots, n$ is
- A) Clamped at its end points [$F'(x_0) = D_0, F'(x_n) = D_n$]
 - B) Parabolically terminated [$F'''(x) \equiv 0$ for $x_0 \leq x \leq x_1$ and $x_{n-1} \leq x \leq x_n$]
 - C) Obtained using MATLAB [$F'''(x)$ for $x_0 \leq x \leq x_1 = F'''(x)$ for $x_1 \leq x \leq x_2$ and $F'''(x)$ for $x_{n-2} \leq x \leq x_{n-1} = F'''(x)$ for $x_{n-1} \leq x \leq x_n$]
5. Modify the m-file for creating natural cubic splines (or write your own) to accommodate the case where A_0 and A_n are linearly extrapolated from A_1, A_2 and A_{n-1}, A_{n-2} respectively.
6. A video camera is mounted on the roof of a vehicle which is traveling at constant speed. The path of the vehicle begins at point A, terminates at point G and must include the intermediate points shown in the diagram below. Design the path so that the frame sequence for the motion is smooth over the entire route.

Pt	A	B	C	D	E	F	G
x	0.1	0.2	0.35	0.5	0.65	0.8	0.95
y	0.1	0.4	0.25	0.7	0.85	0.7	0.35



Data Points Describing Route Traveled in Problem 6

7. A logistic growth curve is defined by the equation

$$P(t) = \frac{MP_0}{\left[P_0 + (M - P_0)e^{-rM(t-t_0)} \right]}$$

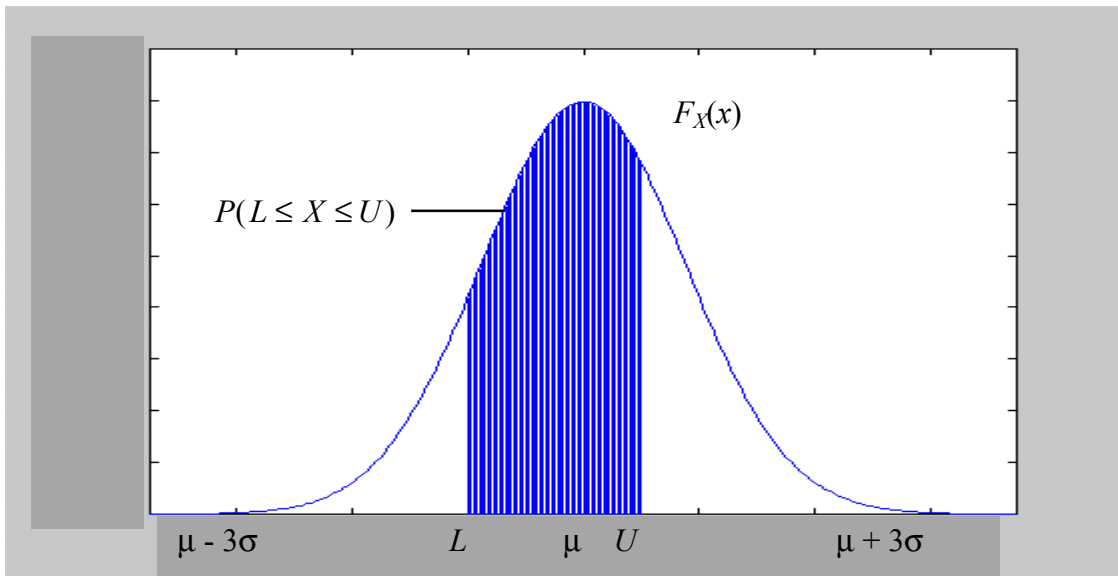
where P_0 is the initial population at time t_0 while r and M are parameters related to the growth rate and limiting population. Consider a logistic growth model for a population with the following parameter values: $t_0 = 0$, $P_0 = 50000$ people, $r = 0.125 \times 10^{-6}$ (people-year) $^{-1}$ and $M = 900000$ people.

- Graph the population $P(t)$ for $0 \leq t \leq 80$.
- Generate a set of data points $[(t_i, P(t_i))]$, $i = 0, 1, 2, \dots, 8$ to include the time and population at the end of every 10 years.
- Find the natural cubic spline to interpolate the data points and graph it over the same time period $0 \leq t \leq 80$.
- Use the knowledge of $P(t)$ to find $P'(t_0)$ and $P'(t_8)$ and repeat part c) using a clamped spline instead.

8. The useful life X of a tire (in miles) is known to follow a Normal probability density function with mean μ miles and standard deviation σ miles (see graph below). The probability of a single tire wearing out between L and U miles is the shaded area under the function between those limits. It is given by

$$\Pr(L \leq X \leq U) = \int_L^U f_X(x) dx$$

$$\text{where } f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{x-\mu}{\sigma}\right)^2} dx$$



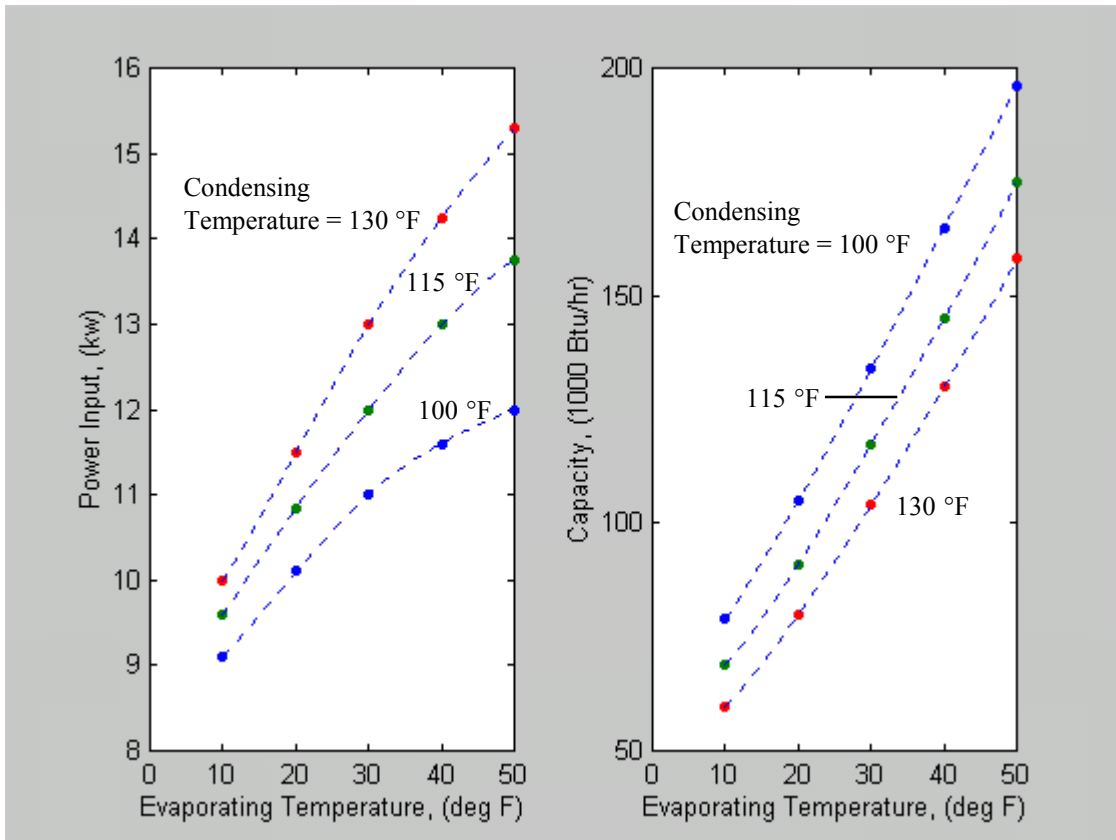
The Normal Probability Density Function

Write a MATLAB m-file to compute the probability above given values for μ and σ as well as L and U .

- First generate equally spaced data points $[x_i, f(x_i)]$, $i = 0, 1, 2, \dots, n$ where $x_0 = L$ and $x_n = U$.
- Approximate $f_X(x)$ from $x_0 = L$ to $x_n = U$ by a composite function $F(x)$ consisting of n linear splines.
- Integrate $F(x)$ from $x_0 = L$ to $x_n = U$ by integrating the n linear splines from x_i to x_{i+1} where $i = 0, 1, 2, \dots, n-1$.
- Repeat steps 1 and 2 using cubic splines.

Test your program with $\mu = 30000$, $\sigma = 2500$, $L = 25000$ and $U = 35000$. Vary n until an acceptable value (based on the estimated probability) is obtained.

9. The graphs shown summarizes test results of a hermetic reciprocating compressor using R-22 running at a constant speed. Capacity refers to the refrigeration effect that can be achieved by the compressor. Power is the electrical power required by the compressor motor.



Graphs for Problem 5.9

- a) On a new set of graphs (one for Power and one for Capacity) plot the data points and composite spline functions for each condensing temperature similar to the dotted curves shown.
- b) The EER (Energy Efficiency Ratio) of a refrigeration system is a measure of performance defined as

$$\text{EER} = \frac{\text{Capacity, in Btu / hr}}{\text{Power Input, in watts}}$$

Use the results of Part a) to prepare a set of performance curves (one for each condensing temperature) relating the EER and the evaporating temperature.

- c) Use the original set of data points to plot on separate graphs composite splines curves of Capacity vs. Condensing Temperature and Power Input vs. Condensing

Temperature. Each graph contains five curves corresponding to Evaporator temperatures of 10 °F, 20 °F, 30°F, 40 °F and 50 °F.

- d) Use the results of Part c) to prepare a set of performance curves (one for each evaporating temperature) relating the EER and the condensing temperature.
10. Import a clip art graphic or scan a simple figure. Digitize it and represent it with a set of composite cubic splines. Display the original graphic and the cubic spline representation along side each other.