Section 1 Introduction

It is not uncommon to be working with two variables e.g. x and y that obey a fixed but unknown relationship $y = f(x)$ between them. The function $f(x)$ relating the two variables may not be as important as merely having the ability to generate (by measurement or experimental observation) numerical values (x_i, y_i) where $y_i = f(x_i)$. For example, the performance of a component may be related to a design parameter in a complicated fashion which is difficult to describe in mathematical terms. In a controlled environment, the lone parameter can be varied and the resulting performance monitored.

Suppose a consumer testing magazine is interested in the repair costs for a particular luxury passenger vehicle after its been in an accident. The test procedure calls for head-on collisions of the vehicle with a stationary object at various speeds. The cost of restoring the vehicle to its pre-crash condition is determined. Tabulated results after several collisions are given below.

Speed (S) mph	Damages (D) \$
10	4500
20	19750
30	43500
40	55000

Table 1.1 Experimental Vehicle Crash Test Data

The table represents a sample of four data points obtained from an unknown function, $D = f(S)$. The experimental process is rather expensive and further experimentation might well be cost prohibitive. The magazine would like to publish a graph for its readers to estimate the cost of repairing the same vehicle resulting from collisions occurring over a range of speeds.

One approach to approximating the unknown function $D = f(S)$ is illustrated in Figure 1.1. The approximating function is the result of connecting the data points from Table 1.1 by straight line segments. The dotted curve represents the actual (unknown) function $D = f(S)$. To estimate the damages from a 15 mph collision, the approximating function is evaluated at this speed as shown.

The process of approximating a function $f(x)$ by another function and subsequent use to estimate numerical values of $f(x)$ is referred to as interpolation. The original function $f(x)$ may be known in analytical form or possibly only through tabulated values that come from it. The approximating function, hereafter referred to as the interpolating function, is ordinarily much simpler than the underlying function $f(x)$. In Figure 1.1, the interpolating function is a piecewise linear function defined by the data points of $f(x)$.

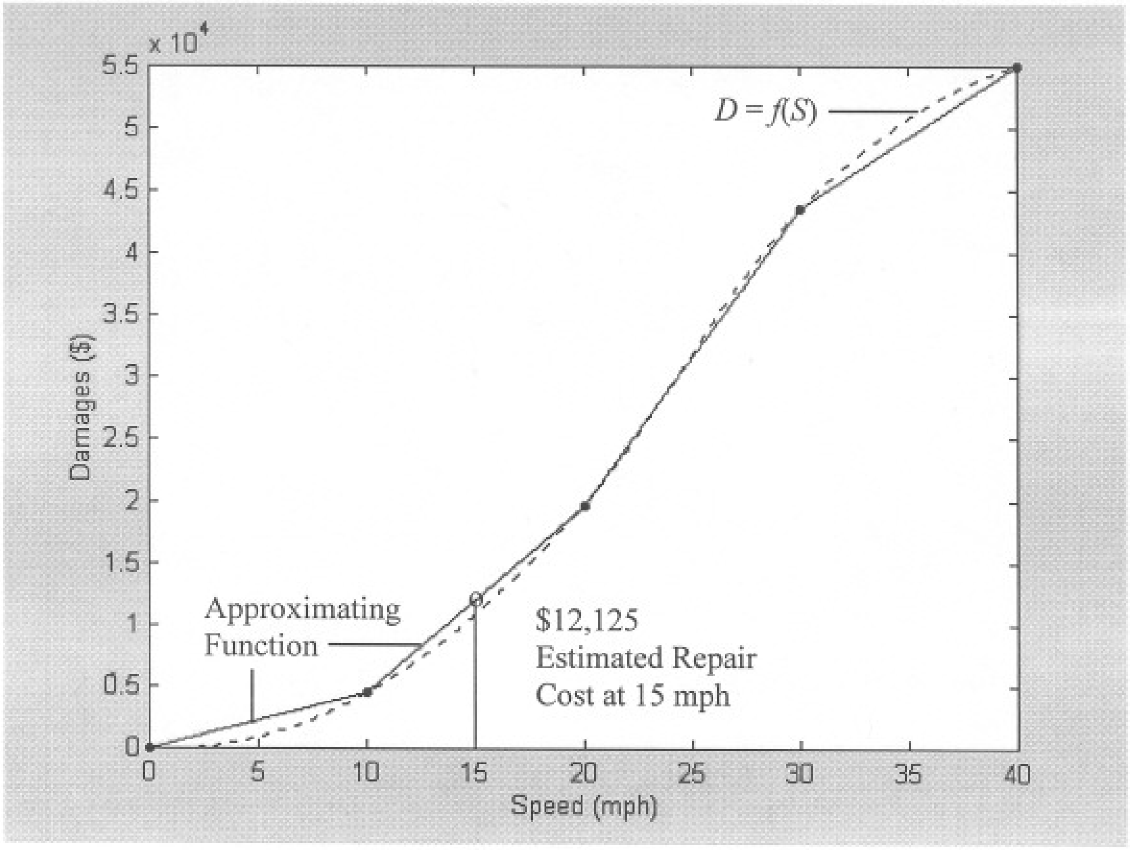


Figure 1.1 An Approximating Function For the Data in Table 1.1

Interpolating functions are generally restricted to an interval containing the data points used to generate them. In this example, estimated repair costs should be confined to speeds from 0 to 40 mph (an additional data point can be assumed at $S = 0$ mph, $D =$ \$0). Attempting to estimate function values outside the range of the data points is termed extrapolation. Caution should be used when extrapolating since erroneous and even nonsensical results are possible.

Interpolation of data representing future predictions is not extrapolation. The two sets of data points graphed in Figure 1.2 represent historical data as well as forecasts of the future behavior of two economic variables. Presumably, sound economic forecasting methods were employed in the process of extrapolating the future values. Regardless, estimating either quantity from 1997 through 2025 is an example of interpolation.

The piecewise linear interpolating function is a suitable approximating function under the right conditions. By in large, if the underlying function $f(x)$ is relatively smooth and well behaved, and the data points are reasonably spaced, then a "connect the dots" approach using piecewise linear interpolation will produce satisfactory results.

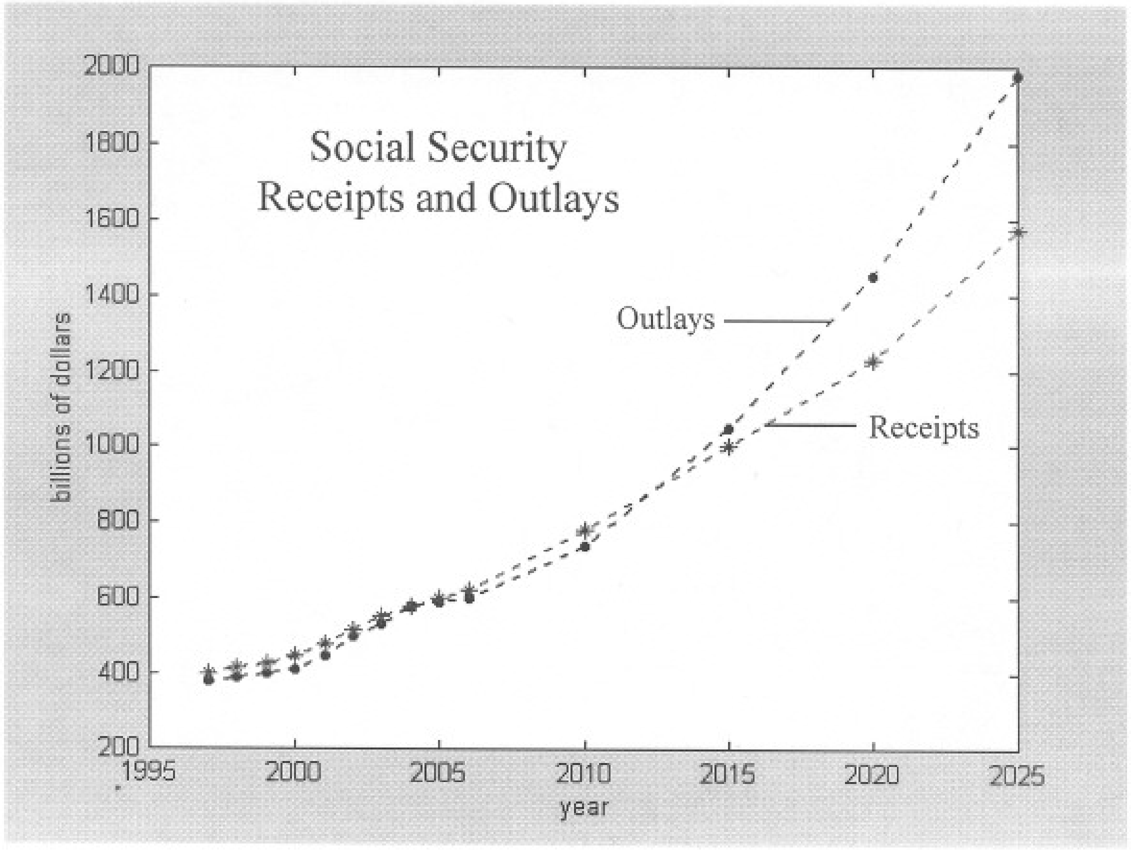


Figure 1.2 Data Points from Functions Predicting Future Values

As an example, consider the case where the function to be approximated is actually $y = f(x) = \sin x$. The table below contains equally spaced values of $\sin x$ from 1 to 2 rad.

i	x_i	$y_i = f(x_i) = \sin x_i$
0	1.00	0.8415
1	1.25	0.9490
2	1.50	0.9975
3	1.75	0.9840
4	2.00	0.9093

Table 1.2 Several Points From the Function $f(x) = \sin x$

The piecewise linear interpolating function connecting the five data points and the function $\sin x$ are shown in Figure 1.3. By observation, it appears that the approximating function is within a few percent of the sine function over the entire interval.

Suppose it is necessary to estimate the sine of $\pi/2$ rad. The y coordinate of point P_1 in Figure 1.3 is the estimate, a reasonably close approximation to the actual value of $\sin \pi/2$. On the other hand, if $\sin \pi/2$ was estimated from the line joining points (1, 0.8415) and (2, 0.9093) then the result would be the y coordinate of point P_2, which differs significantly from the correct value.

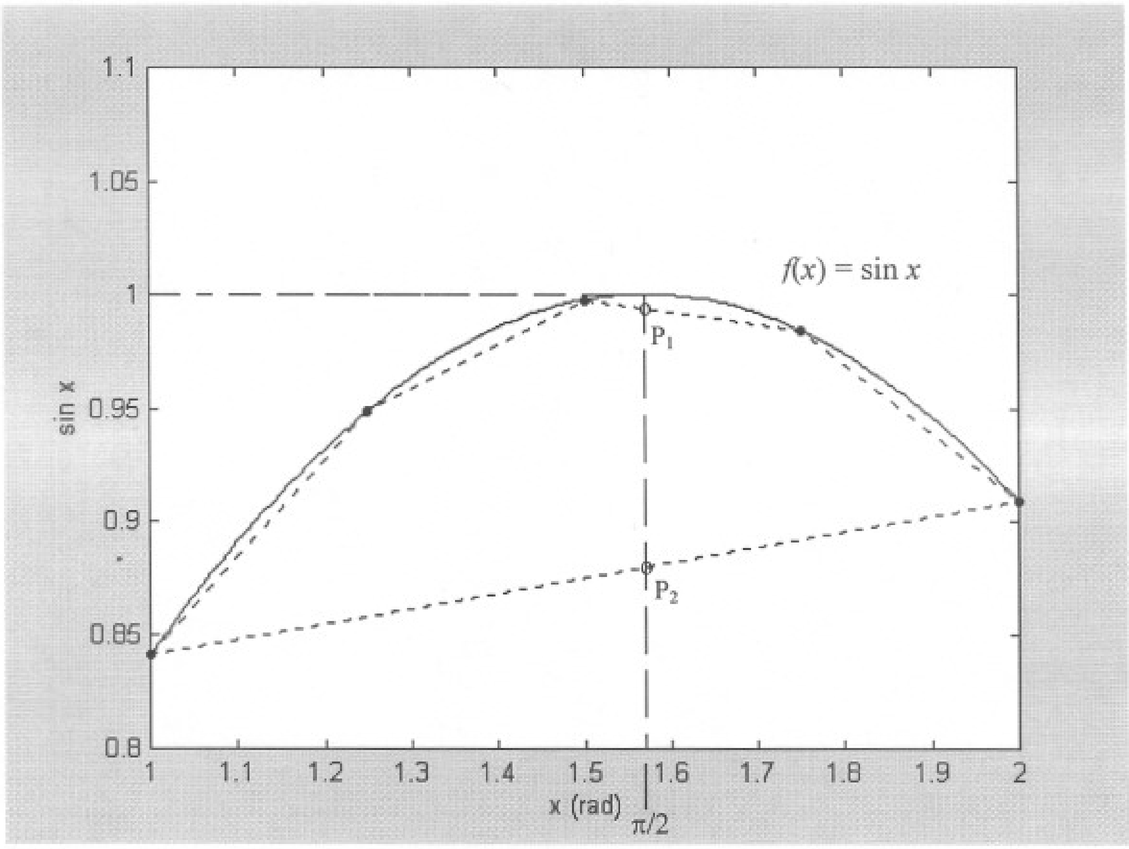


Figure 1.3 Linear Interpolation of $f(x) = \sin x$ to Estimate $\sin \pi/2$

This piecewise linear approach to approximation of functions is the method used in graphing software to plot known analytical functions. The function is evaluated at equally spaced points which are then connected by line segments. The graph appears as a smooth curve when the spacing of points is relatively close. Figure 1.4 illustrates how the approximating function becomes smooth as the points where the function is evaluated become more dense. The function being approximated, i.e. graphed is $f(x) = e^x$.

Piecewise approximating functions are not limited to the case where a linear function approximates the true function between data points. Later in the chapter we consider other polynomial approximations to the underlying function $f(x)$ between consecutive data points.

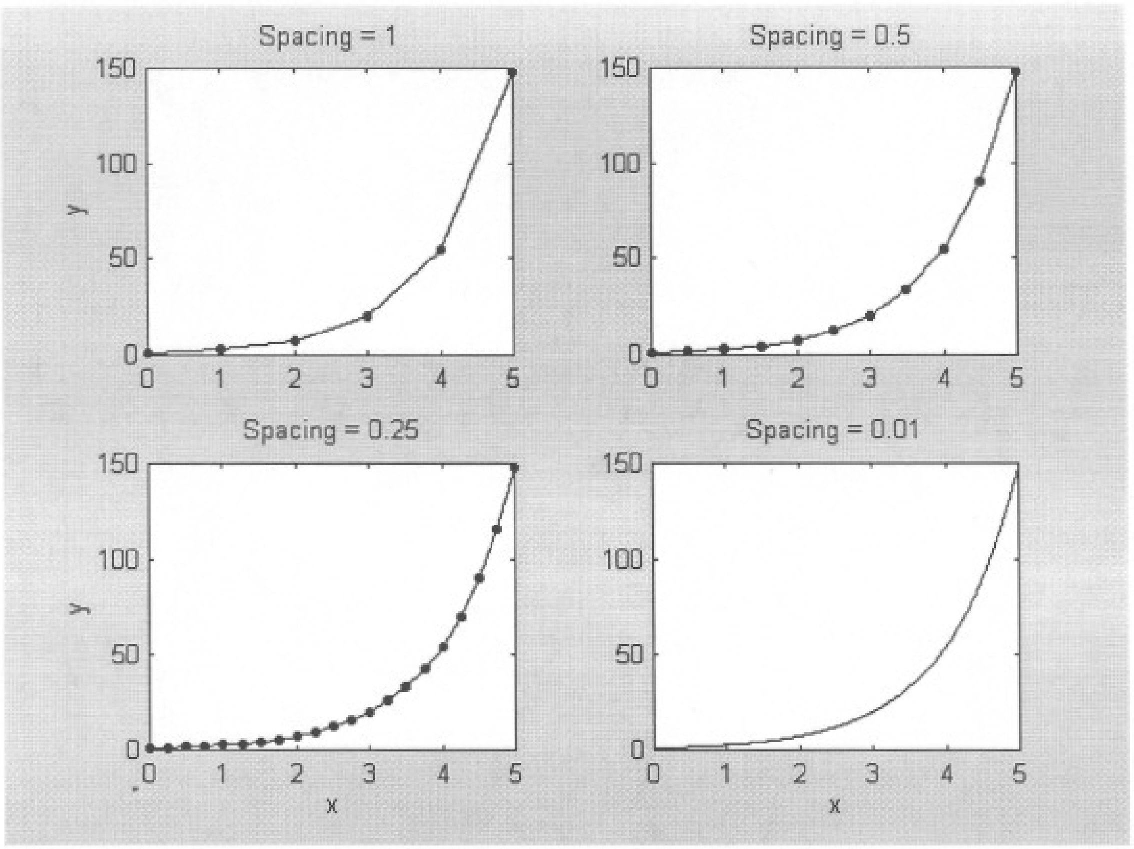


Figure 1.4 Graph of $f(x) = e^x$ Using Linear Interpolation Between Sampled Points

In contrast to the piecewise approximation of a function, interpolating functions are frequently composed of a linear combination of elementary functions $\phi_i(x)$, $i = 0,1,$, n. That is, a function $f(x)$ is approximated by an interpolating function $I(x)$ given by

$$I(x) = a_0\phi_0(x) + a_1\phi_1(x) + \ldots + a_n\phi_n(x) \tag{1.1}$$

A common choice for the elementary functions $\phi_i(x)$ is the monomial function x^i. In this case , the interpolating function is a polynomial, denoted by $f_n(x)$ where

$$f_n(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \tag{1.2}$$

The nth order Taylor Series expansion of a function $f(x)$ introduced in Chapter 1 is a good example of the use of polynomial functions intended for approximation purposes. Interpolating polynomials are discussed in detail in the remaining sections.

There are several reasons why interpolating polynomials are popular when it comes to approximation of functions. There are efficient algorithms for evaluating polynomials when "n" is large or the polynomial must be computed numerous times with different arguments. Differentiation and integration of complex functions is often required. In some instances the function is only available in tabular form. Polynomial approximations of these functions are easily differentiated and integrated.

Section 2 Interpolating Polynomials in Standard Form

Before we can begin to approximate a function $f(x)$, some information about the function has to be known. In the most common situation, a set of N+1 data points (x_i, y_i), $i = 0, 1, 2, \ldots, N$ is obtained in some fashion, where $y_i = f(x_i)$. An nth order ($n \le N$) polynomial can be selected to approximate $f(x)$. The standard form of the interpolating polynomial is

$$f_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots\ldots + a_n x^n \tag{2.1}$$

This polynomial is not guaranteed to pass through the entire set of N+1 data points unless the order n is the same as N and the x_i values are discrete. When the polynomial order n is less than N, it will pass through the $n + 1$ data points used to determine the n + 1 coefficients a_i, $i = 0, 1, 2, \ldots, n$. The $n + 1$ equations are generated from

$$f_n(x_i) = f(x_i), \quad i = 0, 1, 2, \ldots, n \tag{2.2}$$

Figure 2.1 illustrates the case where two different second order polynomials $f_2(x)$ are used to approximate a function $f(x)$ from which five points are known.

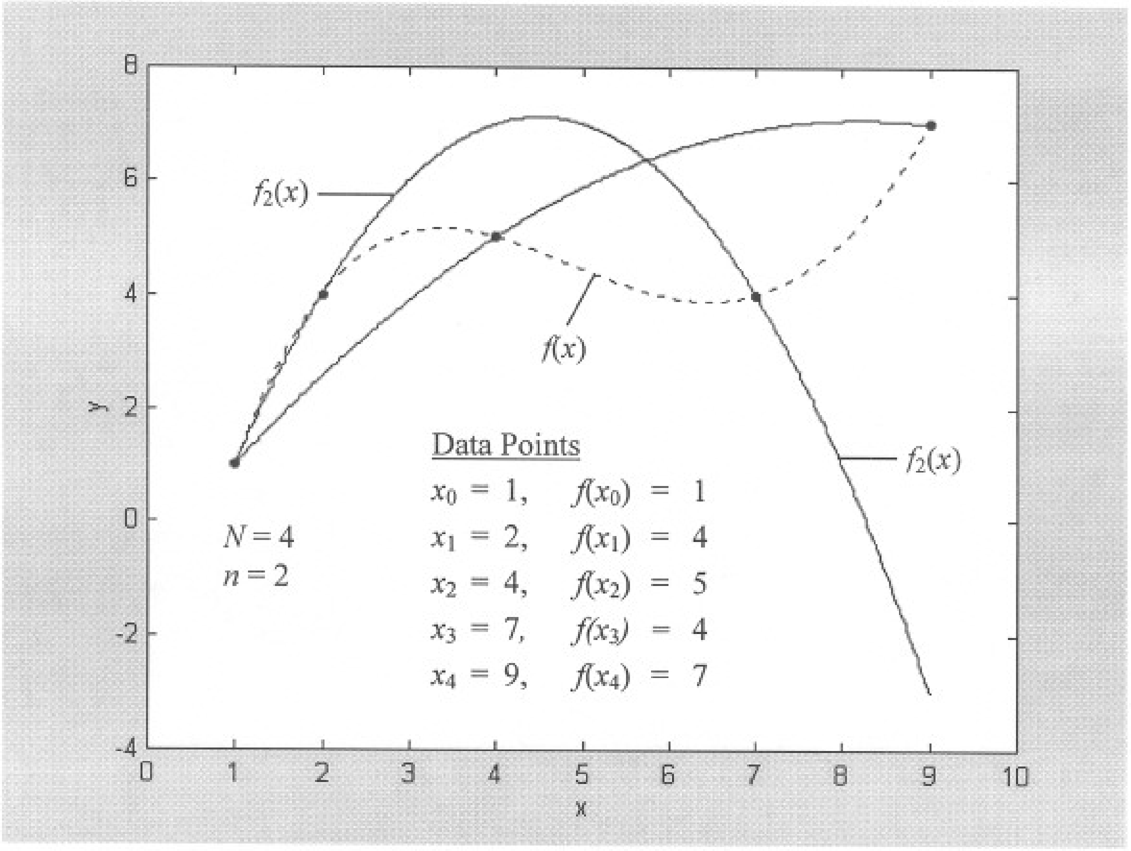


Figure 2.1 Two Second Order Polynomial Approximations to a Function with Five Known Data Points ($N = 4$, $n = 2$)

In general, with $n + 1$ data points specified, the leading coefficient $a_n \neq 0$, and the interpolating polynomial is nth order. Under certain conditions, the highest order term(s) may vanish and the resulting polynomial is less than nth order. For example, if three of the data points in Figure 2.1 happen to be collinear, the coefficient a_2 of the quadratic passing through all three points would be zero and the interpolating polynomial reduces to a linear function.

Equation (2.2) results in a system of $n + 1$ simultaneous equations which can be solved for the coefficients a_i, $i = 0, 1, 2, \ldots, n$. The matrix form of the equations are as follows:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & . & . & . & x_0^{n-1} & x_0^n \\ 1 & x_1 & x_1^2 & . & . & . & x_1^{n-1} & x_1^n \\ 1 & x_2 & x_2^2 & . & . & . & x_2^{n-1} & x_2^n \\ . & . & . & & & & . & . \\ . & . & . & & & & . & . \\ . & . & . & & & & . & . \\ 1 & x_{n-1} & x_{n-1}^2 & . & . & . & x_{n-1}^{n-1} & x_{n-1}^n \\ 1 & x_n & x_n^2 & . & . & . & x_n^{n-1} & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ . \\ . \\ . \\ a_{n-1} \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ . \\ . \\ . \\ f(x_{n-1}) \\ f(x_n) \end{pmatrix} \tag{2.3}$$

The coefficient matrix above is called the Vandermonde matrix and its nonsingular as long as all the x_i values are different. Since a unique solution exists when $x_i \neq x_j$, $i \neq j$, the resulting polynomial in Equation (2.1) is unique, although as we will see later, it can be represented in different forms.

The case of linear interpolation is considered first. The objective is to find a linear approximation to a function when two or more data points of the function are known. The linear approximation becomes the basis for estimation of function values.

The linear approximation $f_1(x)$, from Equation (2.1) with $n = 1$ is,

$$f_1(x) = a_0 + a_1 x \tag{2.4}$$

where the coefficients a_0 and a_1 come from solution of Equation (2.3) with $n = 1$. That is,

$$\begin{pmatrix} 1 & x_0 \\ 1 & x_1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \end{pmatrix} \tag{2.5}$$

Equation (2.5) is easily solved for a_0 and a_1. Doing so, and substituting the results into Equation (2.4) yields the familiar equation of a line

$$f_1(x) = f(x_0) + \left[\frac{f(x_1) - f(x_0)}{x_1 - x_0}\right](x - x_0) \qquad (2.6)$$

Figure 2.2 shows two different linear approximations to an unknown function $f(x)$. In the top graph, $(x_0, f(x_0)) = (1,4)$ and $(x_1, f(x_1)) = (4,11)$ were used to find $f_1(x)$. In the lower graph, the data points from $f(x)$ are renumbered so that $(x_0, f(x_0)) = (2,9)$ and $(x_1, f(x_1)) = (7,24)$.

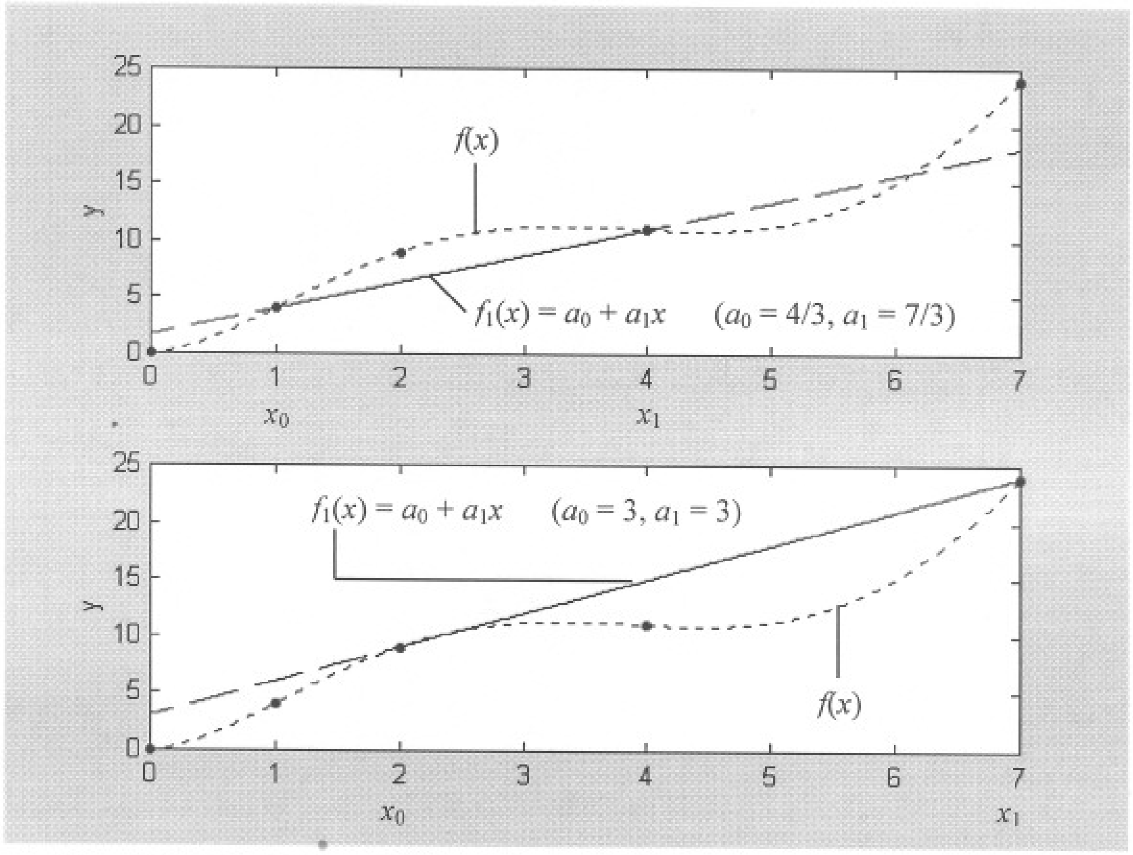


Figure 2.2 Two Different Linear Approximations to an Unknown Function $f(x)$

Keep in mind, interpolation is valid only for x values within the range of the two points used to determine the linear function. Therefore, $f_1(x)$ in the top graph should be used for interpolation only when $1 \leq x \leq 4$ and $f_1(x)$ in the bottom graph for $2 \leq x \leq 7$.

Understanding the limitations imposed on the approximating functions by the use of interpolation is important. We are certainly free to use the approximating functions $f_1(x)$ outside the interval $x_0 \leq x \leq x_1$, however by doing so we may not be using the "best" linear approximation of the function $f(x)$. After all, both linear approximations are based solely on two specific points from $f(x)$ and disregard the remaining (known) data points. In this next chapter we consider methods for obtaining low order polynomial approximating functions that utilize all the known data points. For now we must restrict the domain of approximating functions used for interpolation.

An example of linear interpolation is presented below.

Example 2.1

The table of values from the function $f(x) = \sin x$, Table 1.2, is repeated below. Estimate the value of sin (1.15 rad) using the data point at $x_0 = 1.00$, $y_0 = 0.8415$ and choose the second data point to be each of the four remaining points.

i	x_i	$y_i = f(x_i) = \sin x_i$
0	1.00	0.8415
1	1.25	0.9490
1	1.50	0.9975
1	1.75	0.9840
1	2.00	0.9093

Table 2.1 Several Points From the Function $f(x) = \sin x$

a) $(x_0, y_0) = (1.00, 0.8415)$, $(x_1, y_1) = (1.25, 0.9490)$

$$f_1(x) = f(x_0) + \left[\frac{f(x_1) - f(x_0)}{x_1 - x_0}\right](x - x_0)$$

$$f_1(1.15) = f(1.00) + \left[\frac{f(1.25) - \mathrm{f}(1.00)}{1.25 - 1.00}\right](1.15 - 1.00)$$

$$= 0.8415 + \left[\frac{0.9490 - 0.8415}{1.25 - 1.00}\right](1.15 - 1.00)$$

$$= 0.9060$$

b) $(x_0, y_0) = (1.00, 0.8415)$, $(x_1, y_1) = (1.50, 0.9975)$

$$f_1(1.15) = f(1.00) + \left[\frac{f(1.50) - f(1.00)}{1.50 - 1.00}\right](1.15 - 1.00)$$

$$= 0.8415 + \left[\frac{0.9975 - 0.8415}{1.50 - 1.00}\right](1.15 - 1.00)$$

$$= 0.8883$$

Performing similar calculations for the remaining two end points produces the results,

c) $(x_0, y_0) = (1.00, 0.8415)$, $(x_1, y_1) = (1.75, 0.9840)$

$f_1(1.15) = 0.8700$

d) $(x_0, y_0) = (1.00, 0.8415)$, $(x_1, y_1) = (2.00, 0.9093)$

$f_1(1.15) = 0.8517$

A graphical illustration of the previous calculations is shown in Figure 2.3. It is clear that the accuracy of the interpolated value diminishes as the right end point moves further from the *x* value where the interpolation is performed. Table 2.2 summarizes the results.

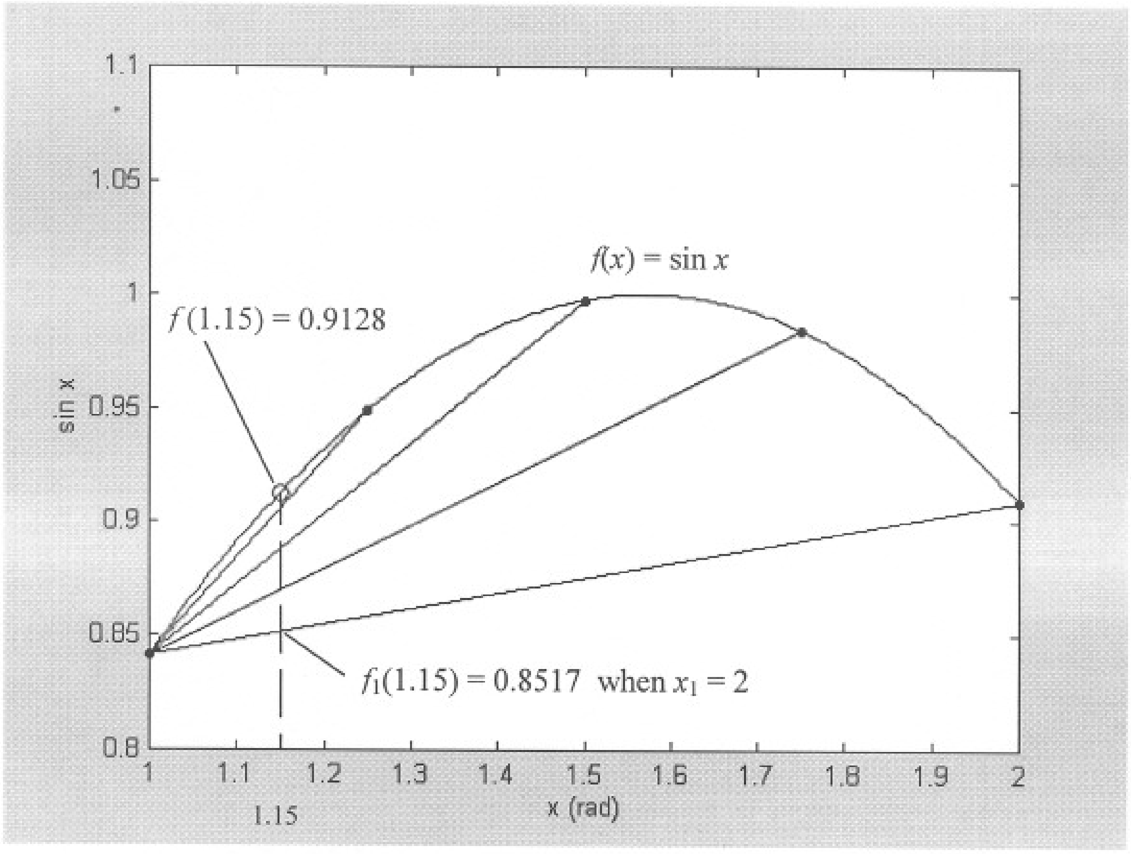


Figure 2.3 Several Linear Interpolating Polynomials for Estimating sin(1.15 rad)

x_0	x_1	$f_1(1.15)$	E_T	e_T, %
1.00	1.25	0.9060	0.0068	0.74
1.00	1.50	0.8883	0.0245	2.68
1.00	1.75	0.8700	0.0428	4.69
1.00	2.00	0.8517	0.0611	6.69

Table 2.2 Accuracy of Linear Interpolation in Example 2.1

Equation (2.6) for linear interpolation is easily implemented with the MATLAB function "interp1". For example, MATLAB statements to generate $f_1(1.15)$ in the first two rows of Table 2.2 are

```
» x0=1;
» f0=sin(x0);
» x1=1.25;
» f1=sin(x1);
» x=[x0 x1];
» f=[f0 f1];
» y=interp1(x,f,1.15)
y =    0.9060
» x1=1.5;
» f1=sin(x1);
» x=[x0 x1];
» f=[f0 f1];
» y=interp1(x,f,1.15)
y =    0.8883
```

The true error, $E_T = f(a) - f_1(a)$ at $x = a$, with linear interpolation will always be zero when the underlying function $f(x)$ is itself a linear function. Of course, it would be pointless to implement linear interpolation to estimate values from a known linear function. In general, linear interpolation produces results which deviate from the true function value, i.e. the true error is ordinarily nonzero. Nonetheless, its possible that the true error could actually be zero or extremely small depending on the actual function $f(x)$ and the location of the point a. Referring to Figure 2.2, if the two end points (0,0) and (7,24) were used to fix $f_1(x)$, there would be two intermediate points where the true error would be zero. Both points would lie at the intersection of the linear interpolating polynomial $f_1(x)$ and the actual function $f(x)$.

Higher order interpolating polynomials are required when the results of linear interpolation may be suspect, either because the spacing between the two end points is too great or the suspected behavior of the function is highly nonlinear in the region of interest. In reality, a combination of both conditions warrants the use of higher order interpolating polynomials.

When 3 data points from a function $f(x)$ are available, the additional data point provides useful information about the curvature of the function in the neighborhood of the points. For example, in Figure 2.4 with only points $P_0(x_0, y_0)$ and $P_1(x_1, y_1)$ obtained from a function $y = f(x)$, it is impossible to account for any nonlinearity inherent in the function. The interpolating polynomial $f_2(x)$ is a quadratic function which passes through all three points thereby capturing some of the nonlinear behavior of the function. Keep in mind that the function $f(x)$ is generally unknown. It is drawn as a dotted curve in Figure 2.4 to emphasize this point.

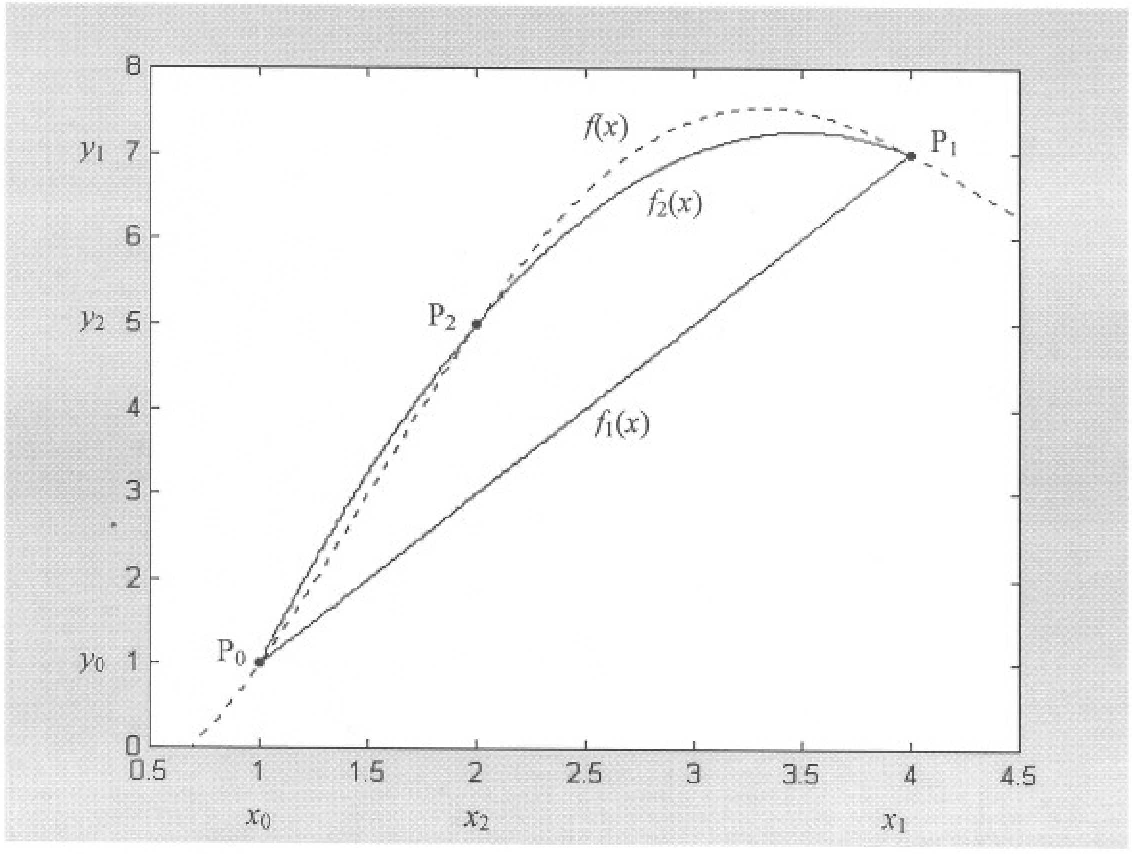


Figure 2.4 First and Second Order Interpolating Polynomials with 3 Data Points

The second order interpolating polynomial $f_2(x)$ is uniquely determined by the three data points $(x_i, f(x_i))$, $i = 0, 1, 2$ which lie on both the function $y = f(x)$ and $f_2(x)$. In standard form $f_2(x)$ is

$$f_2(x) = a_0 + a_1 x + a_2 x^2 \tag{2.7}$$

where the coefficients a_0, a_1, and a_2 are determined from the three data points (x_0, y_0), (x_1, y_1) and (x_2, y_2) which the polynomial must pass through. From Equation (2.7),

$$f_2(x_0) = a_0 + a_1 x_0 + a_2 x_0^2 \tag{2.8}$$

$$f_2(x_1) = a_0 + a_1 x_1 + a_2 x_1^2 \tag{2.8a}$$

$$f_2(x_2) = a_0 + a_1 x_2 + a_2 x_2^{\;2} \qquad (2.8b)$$

From Equation (2.2) with $n = 2$, the interpolating polynomial $f_2(x)$ and the function $f(x)$ are identical when x is either x_0, x_1 or x_2 (refer to Figure 2.4). Replacing $f_2(x_0)$ with $f(x_0)$, $f_2(x_1)$ with $f(x_1)$, and $f_2(x_2)$ with $f(x_2)$ in Equation (2.8) yields

$$f(x_0) = a_0 + a_1 x_0 + a_2 x_0^{\;2} \qquad (2.9)$$

$$f(x_1) = a_0 + a_1 x_1 + a_2 x_1^{\;2} \qquad (2.9a)$$

$$f(x_2) = a_0 + a_1 x_2 + a_2 x_2^{\;2} \qquad (2.9b)$$

Note, we could have used y_0 in place of $f(x_0)$ and the same for y_1, y_2 instead of $f(x_1)$ and $f(x_2)$ in Equation (2.8). Notation aside, what's important is that the constraints represented by Equations (2.9) allow us to solve for the unknown coefficients a_0, a_1 and a_2. Equations (2.9) in matrix form are

$$\begin{pmatrix} 1 & x_0 & x_0^{\;2} \\ 1 & x_1 & x_1^{\;2} \\ 1 & x_2 & x_2^{\;2} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \mathrm{f}(x_0) \\ \mathrm{f}(x_1) \\ \mathrm{f}(x_2) \end{pmatrix} \qquad (2.10)$$

where the 3 by 3 coefficient matrix is the Vandermonde matrix previously introduced in Equation (2.3).

Example 2.2

Find the second order interpolating polynomial that fits through the last 3 points in Table 1.1 of vehicle crash data. Estimate the damages for a crash at 35 mph and compare the result to the same estimate based on linear interpolation using the data points at 30 and 40 mph.

The last 3 data points are (20, 19750), (30, 43500) and (40, 55000). The second order interpolating polynomial is written

$$f_2(s) = a_0 + a_1 s + a_2 s^2 \qquad (2.11)$$

Substituting the data points into Equation (2.11),

$$\begin{pmatrix} 1 & 20 & 20^2 \\ 1 & 30 & 30^2 \\ 1 & 40 & 40^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 19750 \\ 43500 \\ 55000 \end{pmatrix} \qquad (2.12)$$

Using MATLAB to obtain the solution,

```
»x0=20;
»x1=30;
»x2=40;
»b=[19750 43500 55000];
»A=[1 x0 x0*x0; 1 x1 x1*x1; 1 x2 x2*x2];
»a=inv(A)*b';
»a'

a' =    1.0e+004 *

  -6.450000   0.543750   -0.006125
```

and the interpolating second order polynomial is

$$f_2(s) = -64500 + 5437.5\, s - 61.25\, s^2 \tag{2.13}$$

which can be verified by substituting in the 3 data points. Note, there is a Vandermonde matrix function in MATLAB 'vander (**x**)' where **x** is the vector of x_i values. The Vandermonde matrix returned however is slightly different than the standard form defined in Equation (2.3). The difference is that the MATLAB Vandermonde matrix columns are in reverse order compared to its definition in Equation (2.3). The MATLAB Vandermonde matrix can still be used as the coefficient matrix if we simply rewrite Equation (2.12) as follows,

$$\begin{pmatrix} 20^2 & 20 & 1 \\ 30^2 & 30 & 1 \\ 40^2 & 40 & 1 \end{pmatrix} \begin{pmatrix} a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} 19750 \\ 43500 \\ 55000 \end{pmatrix} \tag{2.14}$$

Hence, when the MATLAB Vandermonde matrix is used, the solution vector **a** is [a_2 a_1 a_0] and not the reverse as in the example above. Use of the MATLAB Vandermonde matrix function is illustrated below.

```
»x0=20;
»x1=30;
»x2=40;
»x=[x0 x1 x2];
»b=[19750 43500 55000];
»A=vander(x)
»a=inv(A)*b';
»a'

A =     400           20           1
        900           30           1
       1600           40           1

a' =   1.0e+004 *

  -0.006125   0.543750    -6.450000
```

The first order interpolating polynomial $f_1(s)$ based on the data points (30, 43500) and (40, 55000) and the second order interpolating polynomial $f_2(s)$ using the data points (20, 19750), (30, 43500) and (40, 55000) are shown in Figure 2.5. Using second order polynomial interpolation, the estimated damages for a 35 mph collision is obtained from $f_2(35)$ in Equation (2.13). The result is $50781. The estimated value using linear interpolation is $49250.

MATLAB functions 'polyfit' and 'polyval' will produce the same results. Coefficients for the linear and quadratic interpolating polynomials are computed as well as the interpolated value of damages resulting from a collision at 35 mph.

```
»s=[30 40];
»d=[43500 55000];
»a=polyfit(s,d,1);
»a

a = 1150    9000

»f1_35=polyval(a,35)

f1_35 = 49250

»s=[20 30 40];
»d=[19750 43500 55000];
»a=polyfit(s,d,2)
»a

a = -61.25    5437.5     -6450

»f2_35=polyval(a,35)

f2_35 = 50781.25
```

The use of interpolating polynomials is straightforward; however caution is necessary when dealing with higher order polynomials. This is because high order polynomials can fluctuate dramatically between the sampled data points. Consequently, estimates of the function, which itself may have exhibit a smooth behavior between sampled points, can be notoriously inaccurate when high order polynomials are used. Fortunately, it is readily apparent when this occurs.

Example 2.3 illustrates the tendency of polynomials to vary significantly between data points.

Example 2.3

The liquid flowrate into a tank is adjustable and measurable. Each time the inflow is changed, the output flowrate eventually reaches a new steady-state or equilibrium value as does the height of liquid in the tank. The relationship between output flowrate and height of liquid in the tank at steady-state is of interest. Table 2.3 contains measurements of the tank output flowrate (same as the input flowrate) and height of liquid for various steady-state operating conditions.

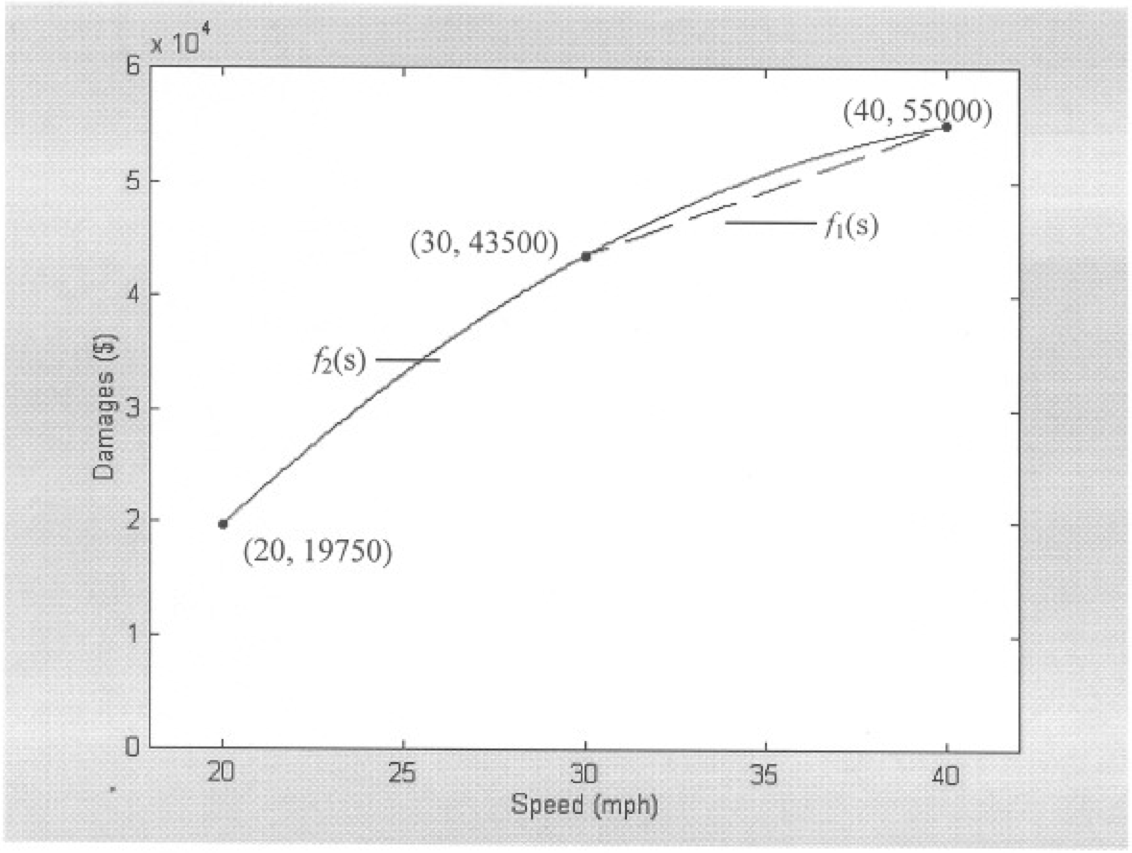


Figure 2.5 Linear and Quadratic Interpolation of Speed Crash Data in Example 3.2

A deterministic relationship exists between the two variables H and F. There is a function $F = f(H)$ that could be evaluated at a specific value of H to determine the corresponding steady-state output flow F. For now, let's assume the actual function $f(H)$ is unknown. The tabulated data is a sampling of points from this function.

Height of Liquid in Tank	Output Flowrate
H (ft)	F (gpm)
0	0
1	50
4	100
9	150
16	200
25	250
36	300

Table 2.3 A Sample of Steady-State Operating Conditions for a Liquid Tank

The 7 data points (H_i, F_i), $i = 0, 1, 2, \ldots, 6$ will uniquely determine a 6th order interpolating polynomial, $f_6(H)$. The MATLAB function 'polyfit' is the quickest way to obtain the vector of coefficients $\mathbf{a} = (a_6, a_5, a_4, a_3, a_2, a_1, a_0)$ that defines the interpolating polynomial $f_6(H)$ below.

$$f_6(H) = a_0 + a_1 H + a_2 H^2 + a_3 H^3 + a_4 H^4 + a_5 H^5 + a_6 H^6 \qquad (2.15)$$

```
H = [0 1 4 9 16 25 36];
F = 0:50:300;
a = polyfit(H,F,6)

a = -5.2609e-005   4.8225e-003 -1.6129e-001       2.4566e+000
   -1.7621e+001    6.5321e+001 -2.9717e-011
```

Figure 2.6 shows the data points and the interpolating polynomial. As expected, all the data points lie on the interpolating polynomial. However, it may be somewhat unsettling to discover the estimated outflow with 20 ft of liquid in the tank is considerably less than the outflow measured when the liquid level was 16 ft. (Refer to Figure 2.6) Even worse, suppose you rely on the interpolating polynomial to predict what the outflow would be when there was 30 ft. of liquid pushing the fluid out. From MATLAB, the results are:

```
»H=[0 1 4 9 16 25 36];
»F=0:50:300;
»a=polyfit(H,F,6)
»f6_20=polyval(a,20)
»f6_30=polyval(a,30)

f6_20  =    168.96

f6_30  =    616.64
```

Of course, one look at the graph of $f_6(H)$ is enough to eliminate its consideration as an interpolating polynomial over the range of fluid levels $0 \le H \le 36$.

In case you're wondering, the dotted function in Figure 2.6 is the correct function $F = f(H)$ for determining the outflow F for any fluid level H. Using some simple principles from Physics, it can be shown that

$$F = f(H) = c\sqrt{H} \qquad (2.16)$$

where the constant of proportionality c is easily computed by substituting in any one of the measured data points (other than $H = 0$, $F = 0$). Using $H = 25$ ft., $F = 250$ gpm, gives $250 = c\sqrt{25}$ from which $c = 50\ \text{gpm}/\text{ft}^{\frac{1}{2}}$. Since we know the true function $f(H)$, let's evaluate the correct outflow for several different fluid levels and compare the results to the estimated outflows using the sixth order interpolating polynomial.

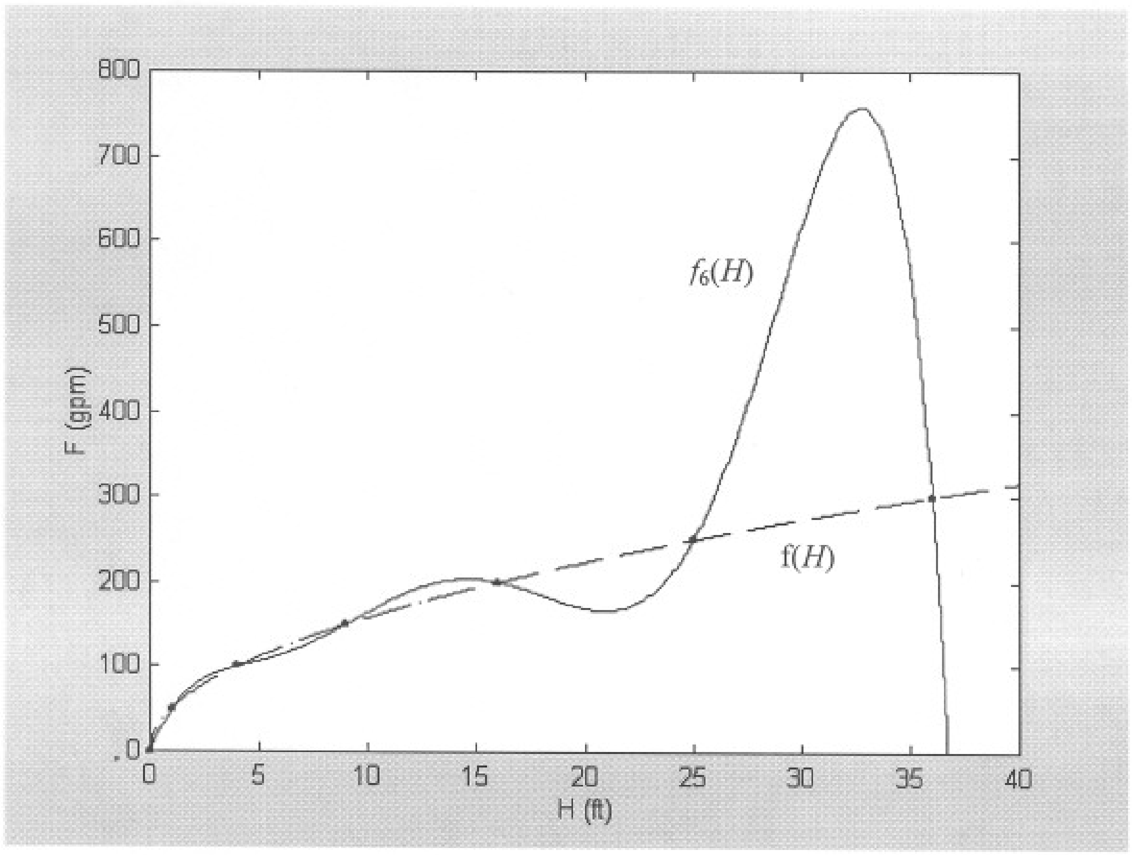


Figure 2.6 Interpolating Polynomial for Data in Table 3.3

Using MATLAB,

```
»for H = 5:5:35
»     f6_H = polyval(a,H)
»     F = 50*H.^0.5;
»     Diff = F - f6_H
»     fprintf('H = %.1f, F(est) = %.1f, F(true) = %.1f,
»     F(true) - F(est) = %.1f\n',H,f6_H, F, Diff)
»end
```

```
H =  5.0, F(est) = 106.6, F(true) = 111.8, F(true) - F(est) = 5.2
H = 10.0, F(est) = 164.4, F(true) = 158.1, F(true) - F(est) = -6.3
H = 15.0, F(est) = 203.5, F(true) = 193.6, F(true) - F(est) = -9.8
H = 20.0, F(est) = 169.0  F(true) = 223.6, F(true) - F(est) = 54.6
H = 25.0, F(est) = 250.0, F(true) = 250.0, F(true) - F(est) = 0.0
H = 30.0, F(est) = 616.6, F(true) = 273.9, F(true) - F(est) = -342.8
H = 35.0, F(est) = 564.9, F(true) = 295.8, F(true) - F(est) = -269.1
```

Notice that the estimated flow from the interpolating polynomial, F(est) is within 5% of the correct flow, F(true), up to roughly 15 ft. of liquid in the tank.

High order polynomials should not be dismissed entirely when it comes to interpolation. Its possible to obtain reliable estimates in certain situations. For example, we might try obtaining additional data points in the interval between $H = 16$ ft. and $H = 36$ ft. The order of the interpolating polynomial would increase by one for each additional point, however the resulting polynomial is constrained to pass through the added points leaving less "wiggle room" to behave as it did previously.

The higher order polynomials could still exhibit fluctuations in the original interval where the inaccuracies were most pronounced ($16 \le H \le 36$). To make matters worse, serious errors could be introduced where the original interpolating polynomial $f_6(H)$ was fairly accurate, i.e. from 0 ft. to 16 ft. Figure 2.7 shows what happens when the original set of data points in Table 2.3 is supplemented with three additional points from $f(H)$ in Equation (2.16). The top graph is the same as Figure 2.6 and the lower one shows the dramatic improvement resulting from the use of $f_9(H)$ as the interpolating polynomial.

It should be readily apparent at this point why interpolating functions should not, as a rule, be used for extrapolation. Indeed, the sixth order interpolating polynomial $f_6(H)$ has a zero near $H = 37$ ft. and the ninth order polynomial is grossly inaccurate as the height increases above $H = 36$ ft.

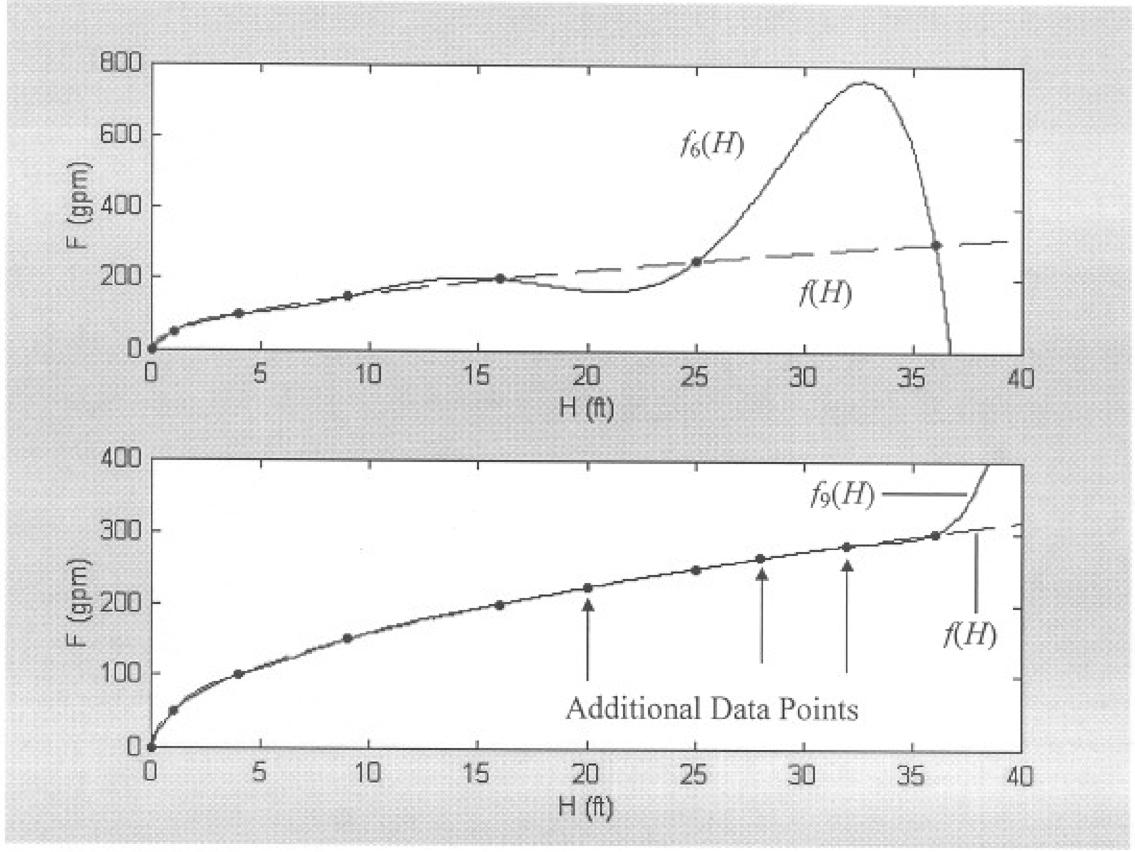


Figure 2.7 Sixth and Ninth Order Interpolating Polynomials for Interpolation of Flowrates Based on Fluid Level

Under the right conditions, interpolating polynomials can be used to optimize functions described by discrete data points. That is, minima and maxima of interpolating polynomials can be used to estimate local or global extreme values of the underlying function $f(x)$ despite the absence of an analytical form. In a specific application, an analyst familiar with the inherent relationship among the variables will ordinarily be able to discriminate between the existence of a true optimal condition and a spurious one resulting from the use of polynomial interpolation. For example, the sixth order polynomial in Figure 2.6 indicates the presence of a local maximum between $H = 30$ ft and $H = 35$ ft. Physically this makes no sense and in fact is sufficient reason to reject using $f_6(H)$ over that interval.

The location(s) of extreme points of an interpolating polynomial can be found by looking for critical values, i.e. zeros of the first derivative or by observation of the interpolating function in the region of interest. The following example demonstrates how interpolating polynomials can be used to solve for an optimal condition.

Example 2.4

An Expressway Authority must determine the access toll charge for a road under its jurisdiction. Based on preliminary studies the authority has obtained the following data relating monthly traffic will the toll charged.

x, Toll (\$)	\$ 0.25	\$ 0.50	\$ 0.75	\$1. 00	\$1.25
y, Monthly Traffic	600,000	450,000	200,000	100,000	40,000

Monthly payments to bond holders to pay off the long term bonds used to finance construction of the expressway is \$ 150,000. The toll will be a multiple of 5 cents.

a) Use a fourth order interpolating polynomial $y = f_4(x)$ to approximate the true relationship $y = f(x)$ between monthly traffic and the access toll charge. Plot the data points and the interpolating polynomial evaluated at 5 cent toll increments on the same graph.

b) The monthly profit P is the difference between the monthly revenue, $R = xy$, and the fixed bond payment $c =$ \$ 150,000. Use the interpolating polynomial $y = f_4(x)$ to generate a second curve to approximate the relationship between P and x.

c) Estimate the maximum profit and the corresponding toll charge to generate the optimum profit.

d) What should the toll be set at if the authority wishes to maximize ridership without losing money each month?

A MATLAB m-file was written to generate the graphs below. From the bottom graph, a toll of \$ 0.55 will result in optimized monthly profits of \$50,460. Zero profit is expected at tolls of \$ 0.30 and \$ 0.80 with maximum ridership of nearly 581,000 vehicles per month at the lesser toll.

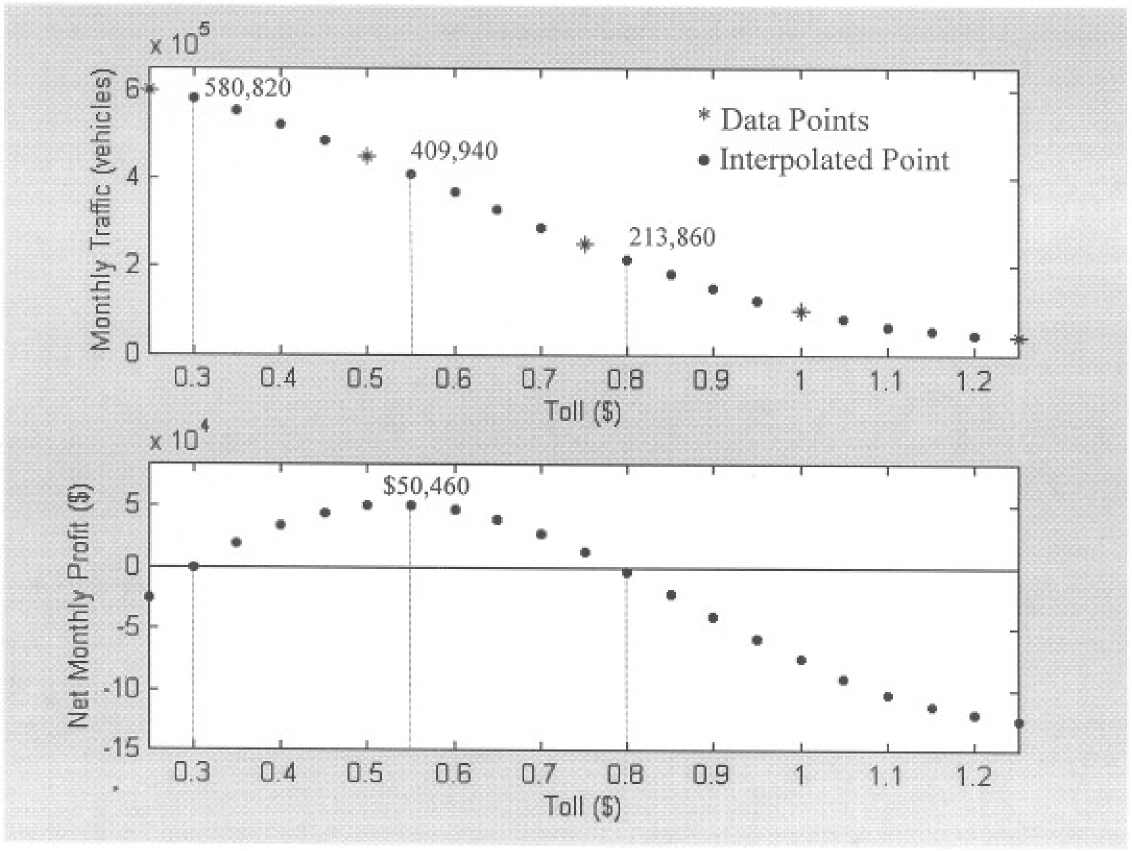


Figure 2.8 Optimum Toll Selection Based on Interpolation of Traffic vs. Toll Data

Section 3 Newton Divided-Difference Interpolating Polynomials

The standard form for representing an *n*th order interpolating polynomial is straightforward. There are however, other representations of *n*th order polynomials which on the surface may seem a bit more unwieldy, but require less manipulation to arrive at. One such form is the subject of this section.

Given a set of $n+1$ data points (x_i, y_i), $i = 0, 1, 2, \ldots, n$ where the x_i are all different and the y_i are sampled from an underlying function $y = f(x)$, the *n*th order interpolating polynomial can be expressed as

$$f_n(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \ldots.. + b_n(x - x_0)(x - x_1)\cdot\cdot\cdot\cdot\cdot(x - x_{n-1}) \qquad (3.1)$$

Before we consider how to determine the coefficients, observe that Equation (3.1) is in fact an nth order polynomial as evidenced by the last term which includes the highest power of x, namely x^n. All powers of x are present in Equation (3.1) despite the fact that the coefficients of say x^0, x^1, x^2,...etc. are not as obvious as when the polynomial is expressed in standard form (see Equation 2.1). Since there are $n+1$ independent coefficients b_i available, $i = 0, 1, 2, \ldots, n$ we can be assured there is at most an *n*th order polynomial that passes through the given data points.

The rationale for selecting the analytical form in Equation (3.1) will be apparent after we look at a few simple examples.

A) Given (x_0, y_0), (x_1, y_1) where $y_0 = f(x_0)$ and $y_1 = f(x_1)$

The first order polynomial for the case when $n = 1$ through the two data points is

$$f_1(x) = b_0 + b_1(x - x_0) \qquad (3.2)$$

Substitution of the two data points (x_0, y_0), (x_1, y_1) into Equation (3.2) gives

$$f_1(x_0) = b_0 + b_1(x_0 - x_0) \qquad (3.3)$$

$$f_1(x_1) = b_0 + b_1(x_1 - x_0) \qquad (3.3a)$$

Solving for b_0 and b_1 yields,

$$b_0 = f_1(x_0) \qquad (3.4)$$

$$b_1 = \frac{f_1(x_1) - f_1(x_0)}{x_1 - x_0} \tag{3.4a}$$

By design, the interpolating function $f_1(x)$ and the actual function $f(x)$ from which the data points were obtained are equal at $x = x_0$ and $x = x_1$ (see Figure 2.2). As a result, b_0 and b_1 can be expressed in terms of the given data,

$$b_0 = f(x_0) \tag{3.5}$$

$$b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{3.5a}$$

Example 3.1

The monthly payment on a 30-yr mortgage of \$100,000. for two different annual interest rates is given in Table 3.1 below. Use an interpolation formula in the form of Equation (3.2) to estimate the monthly payment corresponding to an interest rate of 8.25 % per year.

Data Point Number k	Annual Interest Rate i_k	Monthly Payment $A_k = f(i_k)$
0	7 %	\$ 665.30
1	10 %	\$ 877.57

Table 3.1 Monthly Payments for \$100,000 30-yr Mortgage with Different Interest Rates: Two Data Points

The first order interpolating polynomial is written

$$f_1(i) = b_0 + b_1(i - i_0) \tag{3.6}$$

where

$$b_0 = f(i_0) = f(7) = 665.30$$

$$b_1 = \frac{f(i_1) - f(i_0)}{i_1 - i_0} = \frac{f(10) - f(7)}{10 - 7} = \frac{877.57 - 665.3}{10 - 7} = 70.76$$

The estimated monthly payment is therefore

$$f_1(8.25) = 665.3 + 70.76(8.25 - 7) = 753.68$$

B) Given (x_0, y_0), (x_1, y_1), (x_2, y_2) where $y_0 = f(x_0)$, $y_1 = f(x_1)$, and $y_2 = f(x_2)$

The second order polynomial is written

$$f_2(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) \tag{3.7}$$

where b_0, b_1, and b_2 are determined using the same procedure employed in the previous example.

$$f_2(x_0) = b_0 + b_1(x_0 - x_0) + b_2(x_0 - x_0)(x_0 - x_1) \tag{3.8}$$

$$f_2(x_1) = b_0 + b_1(x_1 - x_0) + b_2(x_1 - x_0)(x_1 - x_1) \tag{3.8a}$$

$$f_2(x_2) = b_0 + b_1(x_2 - x_0) + b_2(x_2 - x_0)(x_2 - x_1) \tag{3.8b}$$

Solving for b_0, b_1, and b_2 gives

$$b_0 = f(x_0) \tag{3.9}$$

$$b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{3.9a}$$

$$b_2 = \frac{\left[\dfrac{f(x_2) - f(x_1)}{x_2 - x_1}\right] - \left[\dfrac{f(x_1) - f(x_0)}{x_1 - x_0}\right]}{x_2 - x_0} \tag{3.9b}$$

Once again, each $f_2(x_i)$ was replaced by $f(x_i)$ so the coefficients could be expressed in terms of known quantities.

There are definite advantages to representing interpolating polynomials in the nonstandard form. In the previous example, there is a sharp contrast in the method of solving for the three coefficients between the standard form representation of a second order polynomial in Equation 2.7 and the nonstandard form of Equation (3.7). In standard form, a_0, a_1, and a_2 are obtained by solving the system of simultaneous equations given in Equation 2.10. On the other hand, b_0, b_1, and b_2 are obtained sequentially as indicated in Equations (3.9), (3.9a) and (3.9b).

Given the choice, a sequential solution is usually preferable. The savings in computation is real. It is faster to evaluate explicit formulas for a set of coefficients one at a time than it is to implement a matrix-based solution where the coefficients are obtained in a parallel fashion, i.e. solution to a system of simultaneous equations. This argument may be less convincing when using calculators or computers with programs designed to solve simultaneous equations. Nonetheless, the solutions are obtained in fundamentally different ways.

The second advantage of the nonstandard representation is more compelling. The expressions for b_0 and b_1 in Equations (3.9) and (3.9a) are the same as in Equation (3.5). Why is this important? Suppose you just completed the process of finding an interpolating polynomial for a given set of data points. There may be some doubt in your mind concerning the accuracy of results based on the use of this polynomial. Later on there will be a discussion of quantitative methods for approximating the errors inherent in interpolation. More data points may be required to reduce the estimated errors. Incorporating additional data points is easier with the nonstandard representation for the interpolating polynomials. This is because each additional data point requires the computation of a single coefficient for the new term in the polynomial. This is illustrated in the following example which extends the results obtained in Example 3.1.

Example 3.2

Suppose we obtain two additional data points in the previous example dealing with the estimation of mortgage payments. The new data points correspond to 8 % and 9 % loans. Use one of the two additional points to obtain a second order interpolating polynomial. Estimate the monthly payment for an 8.25 % loan.

We must choose one of the two new data points to find the second order interpolating polynomial $f_2(i)$. Our intuition suggests the data point for an 8 % loan is the wiser choice because its closer to the point where the estimate is required, i.e. 8.25 %. The second order polynomial is obtained from the data points corresponding to $k = 0$, 1, and 2 in the following table. It is given in Equation (3.10).

Data point Number k	Annual Interest Rate i_k	Monthly Payment $A_k = f(i_k)$
0	7 %	\$ 665.30
1	10 %	\$ 877.57
2	8 %	\$ 733.76
3	9 %	\$ 804.62

Table 3.2 Monthly Payments for \$100,000 30-yr Mortgage with Different Interest Rates: 4 Data Points

$$f_2(i) = b_0 + b_1(i - i_0) + b_2(i - i_0)(i - i_1) \tag{3.10}$$

Coefficients b_0 and b_1 were determined in Example 3.1 using the first two data points. The remaining coefficient b_2 is obtained from Equation (3.9b) as

$$b_2 = \frac{\left[\frac{f(i_2)-f(i_1)}{i_2-i_1}\right]-\left[\frac{f(i_1)-f(i_0)}{i_1-i_0}\right]}{i_2-i_0}$$

$$= \frac{\left[\frac{733.76-877.57}{8-10}\right]-\left[\frac{877.57-665.30}{10-7}\right]}{8-7}$$

$$= 1.148$$

The polynomial $f_2(i)$ is therefore,

$$f_2(i) = 665.3 + 70.76(i-i_0) + 1.148(i-i_0)(i-i_1) \qquad (3.11)$$

and the estimate of payments for an 8.25 % mortgage is now

$$f_2(8.25) = 665.3 + 70.76(8.25-7) + 1.148(8.25-7)(8.25-10)$$

$$= 751.24$$

The previous two examples are somewhat academic in nature. Quite obviously, the loan officer at the bank will not be interested in your calculations to estimate the monthly payments. He or she has access to the function $A = f(i)$ from which you obtained several data points in the Sunday paper. Let's see how close your two estimates were to the correct answer. Figure 3.1 includes graphs of the true function $f(i)$ as well as the interpolating polynomials $f_1(i)$ and $f_2(i)$. The upper plot includes the data points as well.

It is clear from looking at the upper plot that either interpolating function $f_1(i)$ or $f_2(i)$ will provide accurate estimates of the true monthly payment function $f(i)$ over the entire range from $i = 7$ % to $i = 10$ % and then some. The lower plot is an enlargement of each graph in the region about $i = 8.25$ %, the interest rate under consideration. Here we see that the quadratic interpolating polynomial $f_2(i)$ and the real function $f(i)$ are virtually indistinguishable. You may be interested in knowing that the true function A = $f(i)$ is given by

$$A = f(i) = P\frac{(i/1200)(1+i/1200)^n}{\left[(1+i/1200)^n - 1\right]} \qquad (3.12)$$

where P is the mortgage amount, n is the loan period in months and i is the annual percent interest rate. Perhaps you are surprised at how close the graphs of the low order interpolating polynomials $f_1(i)$ and $f_2(i)$ are to the true function $f(i)$ which, based on its analytical form in Equation (3.12), does not appear to resemble a polynomial function.

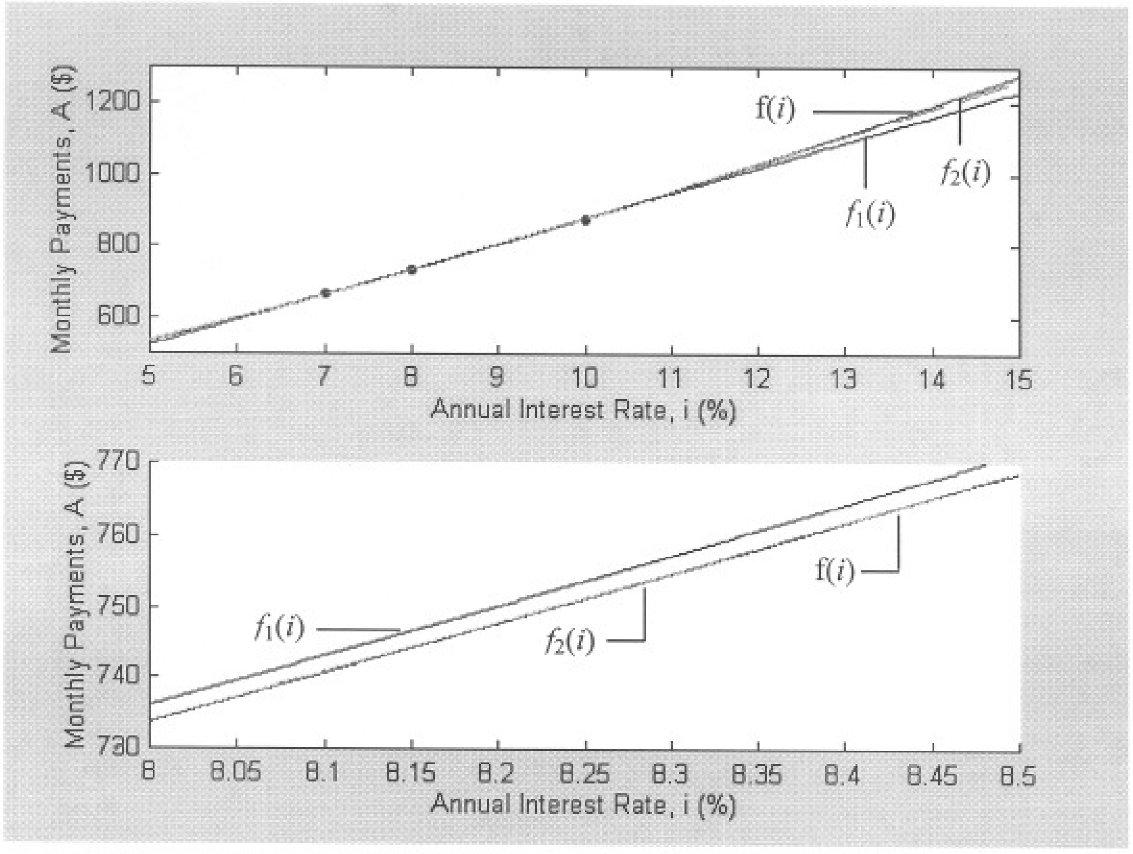


Figure 3.1 First and Second Order Interpolating Polynomials $f_1(i)$, $f_2(i)$ and the True Function $f(i)$

Table 3.3 illustrates the accuracy one can expect when using low order polynomial interpolation over the range of interest rates under consideration.

i (%)	$f(i)$	$f_1(i)$	E_T	$f_2(i)$	E_T
7.5	699.21	700.68	-1.47	699.25	-0.04
8.5	768.91	771.43	-2.52	768.86	0.05
9.5	840.85	842.19	-1.34	840.76	0.09

Table 3.3 Errors in First and Second Order Interpolation for Examples 3.1 and 3.2

We still have an additional data point that could be used to increase the order of the interpolating polynomial. The formulas for higher order coefficients become unwieldy. Fortunately, there is a framework for determining the coefficients b_i, $i = 0, 1, 2, \ldots, n$ of a general nth order interpolating polynomial as given in Equation (3.1).

The general approach is based on the evaluation of finite divided differences. Given a set of n+1 data points $[x_i, f(x_i)]$, $i = 0, 1, 2, \ldots, n$ the finite divided differences of various orders are defined in Equations (3.13) - (3.13e). Note how the divided differences are obtained recursively from two divided differences of order one less.

0 th order
$$f[x_i] = f(x_i) \tag{3.13}$$

1 st order
$$f[x_i, x_j] = \frac{f[x_i] - f[x_j]}{x_i - x_j}, \qquad i \neq j \tag{3.13a}$$

$$= \frac{f(x_i) - f(x_j)}{x_i - x_j} \tag{3.13b}$$

2 nd order
$$f[x_i, x_j, x_k] = \frac{f[x_i, x_j] - f[x_j, x_k]}{x_i - x_k}, \qquad i \neq j \neq k \tag{3.13c}$$

$$= \frac{\dfrac{f(x_i) - f(x_j)}{x_i - x_j} - \dfrac{f(x_j) - f(x_k)}{x_j - x_k}}{x_i - x_k} \tag{3.13d}$$

*n*th order
$$f[x_n, x_{n-1}, \dots, x_1, x_0] = \frac{f[x_n, x_{n-1}, \dots, x_2, x_1] - f[x_{n-1}, x_{n-2}, \dots, x_1, x_0]}{x_n - x_0} \tag{3.13e}$$

When there are 3 data points ($n = 2$), the divided differences $f[x_0]$, $f[x_1, x_0]$, and $f[x_2, x_1, x_0]$ are identical to the coefficients b_0, b_1, and b_2. [see Equations (3.14) - (3.14h)]

The coefficient b_0 of the interpolating polynomial $f_2(x)$ is numerically equal to the first of the three zero order finite divided differences, i.e. the one that depends on the data point $[x_0, f(x_0)]$. The coefficient b_1 is equal to the first of the two first order finite divided differences, i.e. the one that depends on the data points $[x_0, f(x_0)]$ and $[x_1, f(x_1)]$. Finally, the coefficient b_2 is equal to the first and only second order finite divided difference, i.e. the one that requires all three data points, $[x_0, f(x_0)]$, $[x_1, f(x_1)]$ and $[x_2, f(x_2)]$.

In general, with $n + 1$ data points there are $n + 1$ zero order divided differences, n first order divided differences, $n-1$ second order divided differences, etc. up to one nth order divided difference. The first computed divided difference of order "i" is equal to the coefficient b_i, $i = 0, 1, 2, \dots, n$.

$$f[x_0] = f(x_0) \tag{3.14}$$

$$= b_0 \tag{3.14a}$$

$$f[x_1, x_0] = \frac{f[x_1] - f[x_0]}{x_1 - x_0} \tag{3.14c}$$

$$= \quad \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{3.14d}$$

$$= \quad b_1 \tag{3.14e}$$

$$f[x_2, x_1, x_0] \quad = \quad \frac{f[x_2, x_1] - f[x_1, x_0]}{x_2 - x_0} \tag{3.14f}$$

$$= \quad \frac{\left[\dfrac{f(x_2) - f(x_1)}{x_2 - x_1}\right] - \left[\dfrac{f(x_1) - f(x_0)}{x_1 - x_0}\right]}{x_2 - x_0} \tag{3.14g}$$

$$= \quad b_2 \tag{3.14h}$$

Table 3.4 is a helpful aid in remembering how to compute the finite divided differences. The coefficients b_i, $i = 0, 1, 2, \ldots, n$ of the interpolating polynomial in Equation (3.1) are the first finite divided-difference entries in their respective columns. The polynomial is referred to as the Newton divided-difference interpolating polynomial. The arrows indicate which finite divided differences are used to calculate the higher order ones. Always remember, the denominator of any divided difference is the difference between its first and last argument.

$$\begin{tikzcd}[row sep=tiny, column sep=small]
 & & f_1[\ ] & f_2[\ ] & f_{n-1}[\ ] & f_n[\ ] \\
x_0 & f[x_0] \arrow[dr] & & & & \\
 & & f[x_1, x_0] \arrow[dr] & & & \\
x_1 & f[x_1] \arrow[ur] \arrow[dr] & & f[x_2, x_1, x_0] & & \\
 & & f[x_2, x_1] \arrow[ur] \arrow[dr] & & & \\
x_2 & f[x_2] \arrow[ur] \arrow[dr] & & f[x_3, x_2, x_1] & & \\
 & & f[x_3, x_2] \arrow[ur] \arrow[dr] & & & \\
x_3 & f[x_3] \arrow[ur] \arrow[dr] & & f[x_4, x_3, x_2] & & \\
\cdot & \cdot & f[x_4, x_3] \arrow[ur] & \cdot & & \\
\cdot & \cdot & \cdot & \cdot & \cdot \; f[x_{n-1}, x_{n-2}, \ldots, x_2, x_0] \arrow[dr] & \\
\cdot & \cdot & \cdot & \cdot & \cdot & f[x_n, x_{n-1}, x_{n-2} \ldots, x_1, x_0] \\
\cdot & \cdot & \cdot & \cdot & \cdot \; f[x_n, x_{n-1}, \ldots x_2, x_1] \arrow[ur] & \\
x_{n-2} & f[x_{n-2}] \arrow[dr] & \cdot & \cdot & & \\
 & & f[x_{n-1}, x_{n-2}] \arrow[dr] & \cdot & & \\
x_{n-1} & f[x_{n-1}] \arrow[ur] \arrow[dr] & & f[x_n, x_{n-1}, x_{n-2}] & & \\
 & & f[x_n, x_{n-1}] \arrow[ur] & & & \\
x_n & f[x_n] \arrow[ur] & & & &
\end{tikzcd}$$

Table 3.4 Table of Finite Divided Differences for a Function $f(x)$ Known at Discrete Data Points

Example 3.3

Torque-speed data for an electric motor is given in the first two columns of the table below. Find the equation of the Newton divided-difference interpolating polynomial that passes through each data point and use it to estimate the torque at 1800 rpm.

Speed, ω (rpm x 1000)	Torque, T_i (ft-lb)	$f_1[\,]$	$f_2[\,]$	$f_3[\,]$	$f_4[\,]$
0.5	31 (b_0)				
		-6 (b_1)			
1.0	28		-2 (b_2)		
		-8		-6.667 (b_3)	
1.5	24		-12		6 (b_4)
		-20		5.333	
2.0	14		-4		
		-24			
2.5	2				

Table 3.5 Data Points and Finite Divided Differences for Example 3.6

The notation $f_1[\ ]$, $f_2[\ ]$, $f_3[\ ]$, and $f_4[\ ]$ represent finite divided differences of order 1 through 4, respectively. The finite divided differences and the coefficients b_0, b_1, b_2, b_3, and b_4 of the interpolating polynomial are shown in the table. The result is,

$$T(\omega) = b_0 + b_1(\omega-\omega_0) + b_2(\omega-\omega_0)(\omega-\omega_1) + b_3(\omega-\omega_0)(\omega-\omega_1)(\omega-\omega_2)$$
$$+\ b_4(\omega-\omega_0)(\omega-\omega_1)(\omega-\omega_2)(\omega-\omega_3) \qquad (3.15)$$

$$T(\omega) = 31 - 6\,(\omega-0.5) - 2\,(\omega-0.5)(\omega-1) - 6.667\,(\omega-0.5)(\omega-1)(\omega-1.5)$$
$$+\ 6\,(\omega-0.5)(\omega-1)(\omega-1.5)(\omega-2) \qquad (3.16)$$

$$T(1.8) = 31 - 6\,(1.8-0.5) - 2\,(1.8-0.5)(1.8-1) - 6.667\,(1.8-0.5)(1.8-1)(1.8-1.5)$$
$$+\ 6\,(1.8-0.5)(1.8-1)(1.8-1.5)(1.8-2)$$

$$T(1.8) = 18.7$$

The true function $T = f(\omega)$ is not available to assess the accuracy of this estimate. Conceivably, one could derive a structure or analytical form of the function based on established engineering principles and scientific laws. A number of physical parameters, i.e. constants would likewise have to be known before the function could be used for evaluation of motor performance. Rarely is this ever attempted in situations where all that is required is a reliable estimate of how a particular device, component or system will perform. Conversely, a model of the torque motor based on scientific principles is far more useful to an engineer designing a motor to satisfy specific design criteria.

There are situations where a combination of empirical and scientific modeling are used in conjunction with each other. Imagine a situation where a theoretical model is known, however it may be of such complexity that data points are expensive to obtain. In this situation, a minimum number of data points can be obtained from solution of equations (differential and algebraic) comprising the scientific model. The data points are then used as a basis to generate an empirical model for interpolation.

Little has been said up to this point about the ordering of the data points, primarily because an nth order polynomial passing through $n+1$ data points is unique and the ordering is irrelevant. Changing the order of the data points will affect the representation of the polynomial; however the polynomial itself has not changed. The selection of which $n+1$ data points to draw from a larger sample to obtain an nth order polynomial is a different matter. In this case, some thought should be given as to which $n+1$ points produce the best interpolating polynomial. Example 3.4 illustrates these points.

Example 3.4

Consider the data in Table 3.2. Find the third order Newton divided-difference interpolating polynomial $f_3(i)$ containing the 4 data points using different orderings of the data points. Estimate the function value $f(8.25)$.

The data from Table 3.2 and the divided differences are given in Table 3.6.

k	i_k	$A_k = f(i_k)$	$f_1[\,]$	$f_2[\,]$	$f_3[\,]$
0	7	665.30 (b_0)			
			68.46 (b_1)		
1	8	733.76		1.200 (b_2)	
			70.86		-0.05167 (b_3)
2	9	804.62		1.045	
			72.95		
3	10	877.57			

Table 3.6 Finite Divided Differences and Newton Interpolating Polynomial Coefficients for Data in Example 3.2

The coefficients b_0, b_1, b_2 and b_3 are read directly from the table. The resulting third order Newton divided difference interpolating polynomial is given in Equation (3.18)and the polynomial is then used for the interpolation of $f(8.25)$.

$$f_3(i) = b_0 + b_1(i-i_0) + b_2(i-i_0)(i-i_1) + b_3(i-i_0)(i-i_1)(i-i_2) \tag{3.17}$$

$$f_3(i) = 65.3 + 68.46(i-7) + 1.2(i-7)(i-8) - 0.05167(i-7)(i-8)(i-9) \tag{3.18}$$

$$\begin{aligned} f_3(8.25) &= 65.3 + 68.46(8.25-7) + 1.2(8.25-7)(8.25-8) - 0.05167(8.25-7)(8.25-8)(8.25-9) \\ &= 751.26 \end{aligned}$$

In Table 3.7 the data points are reordered and the process of finding the interpolating polynomial is repeated.

k	i_k	$A_k = f(i_k)$	$f_1[\,]$	$f_2[\,]$	$f_3[\,]$
0	7	665.30 (b_0)			
			70.757 (b_1)		
1	10	877.57		1.148 (b_2)	
			71.905		-0.0515 (b_3)
2	8	733.76		1.045	
			70.860		
3	9	804.62			

Table 3.7 Finite Divided Differences and Newton Interpolating Polynomial Coefficients for Data in Example 3.2 with Reordering of Data Points

The new polynomial based on reordered data points and the interpolated value are

$$f_3(i) = 65.3 + 70.757(i-7) + 1.148(i-7)(i-10) - 0.0515(i-7)(i-10)(i-8) \tag{3.19}$$

$$\begin{aligned} f_3(8.25) &= 65.3 + 70.757(8.25-7) + 1.148(8.25-7)(8.25-10) \\ &\quad - 0.0515(8.25-7)(8.25-10)(8.25-8) \\ &= 751.26 \end{aligned}$$

As expected, the estimated values are the same in both cases. It's a simple matter to show that the polynomial expressions in Equations (3.18) and (3.19) are in fact the same third order polynomial.

Restricting the interpolating polynomial to be 2nd order will produce as many different 2nd order polynomials as there are ways to select groups of three points from a total of four data points. In general, for a specific value of *i*, referred to as the interpolant, each interpolating polynomial $f_2(i)$ will yield a different estimate of $f(i)$. One such estimate of $f(8.25)$ has already been computed using the first three points in Table 3.2. The optimal choice of data points for determining the second order interpolating polynomial should be the set of points closest to the interpolant. This will be proven later.

Generally speaking, when Newton divided-difference polynomials $f_n(x)$ are used for interpolation to estimate values of a function $f(x)$, we should expect a difference between $f_n(x)$ and the true function value $f(x)$. In the case of linear interpolation, the error incurred, $R_1(x)$ is the difference between $f(x)$ and $f_1(x)$ and it varies over the interval $x_0 \leq x \leq x_1$ as shown in Figure 3.2. By definition, $R_1(x)$ satisfies

$$f(x) = f_1(x) + R_1(x) \tag{3.20}$$

Solving for $R_1(x)$,

$$R_1(x) = f(x) - f_1(x) \tag{3.21}$$

We should not expect to implement Equation (3.21) when $f(x)$ is unknown. If there was some way to estimate $f(x)$, we could obtain an approximation for the error term or remainder $R_1(x)$ as its sometimes referred to. From Figure 3.2, observe that $f_2(x)$ could be used for that purpose. That is,

$$R_1(x) \approx f_2(x) - f_1(x) \tag{3.22}$$

The right hand side of Equation (3.22) is nothing more than the second order term of the polynomial $f_2(x)$. This is easily verified by looking at Equations (3.2) and (3.7). Thus,

$$R_1(x) \approx b_2(x - x_0)(x - x_1) \tag{3.23}$$

However, in order to use Equation (3.23), an additional data point $[x_2, f(x_2)]$ is required to obtain b_2. How good is the approximation of $R_1(x)$ from Equation (3.22) or equivalently Equation (3.23)? Comparison of Equations (3.21) and (3.22) reveals that the approximation of the error term $R_1(x)$ depends on how close the function $f(x)$ and the interpolating polynomial $f_2(x)$ are. Figure 3.2 illustrates this point graphically. If the function $f(x)$ happens to be a quadratic polynomial, then $f_2(x)$ and $f(x)$ are identical and the estimate of $R_1(x)$ would be exact for any value of x.

A numerical example will help clarify the process of approximating the errors occurring from the use of Newton divided-difference interpolating polynomials.

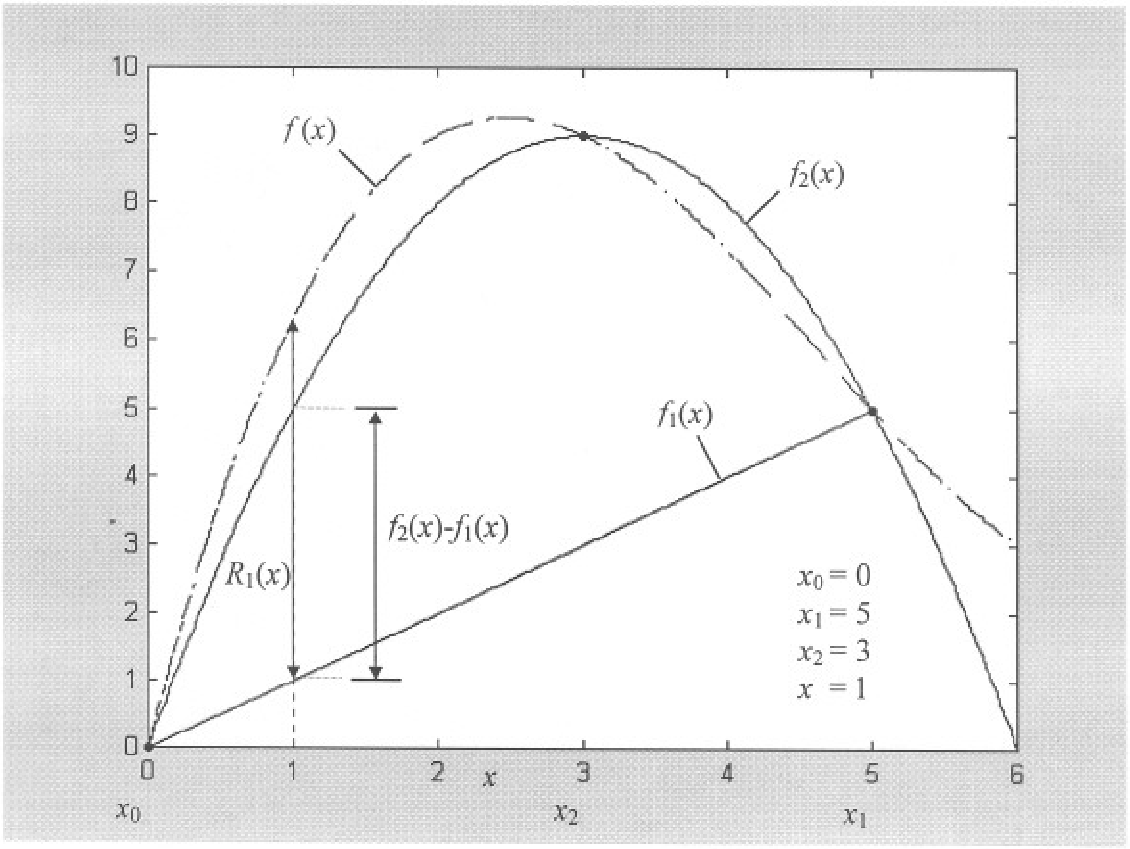


Figure 3.2 Estimating the error $R_1(x)$ in Linear Interpolation

Example 3.5

The solution to the differential equation $\frac{d}{dt}y(t) + y(t) = 1$ when $y(0) = 0$ is $y(t) = 1 - e^{-t}$, $t \geq 0$. Let T be the time it takes for the solution $y(t)$ to reach the value A, where $0 \leq A \leq 1$, i.e. $y(T) = A$. Several points (T, A) on the solution $y(t)$ are tabulated below. A low order polynomial is needed for interpolation of T for a given value of A.

T	A	T	A
0.0000	0.0	0.6931	0.5
0.1054	0.1	0.9163	0.6
0.2231	0.2	1.2040	0.7

0.3567	0.3	1.6094	0.8
0.5108	0.4	2.3026	0.9

Table 3.8 Data Points for Finding Interpolating Polynomial to Estimate T Given A

The underlined data points were selected to determine a third order Newton divided-difference polynomial which can be used for interpolation over the interval. The divided difference table is shown below.

i	A_i	$T_i = f(A_i)$	$f_1[\,]$	$f_2[\,]$	$f_3[\,]$	$f_4[\,]$
0	0.0	0.0000				
			1.1890			
1	0.3	0.3567		1.3275		
			2.1183		4.7745	
2	0.7	1.2040		5.6246		7.6676
			5.4930		8.6083	
3	0.9	2.3026		7.3462		
			4.0238			
4	0.5	0.6931				

Table 3.9 Divided Difference Table for Finding $f_3(A)$ and $f_4(A)$

From the table, the third order interpolating polynomial $f_3(A)$ is

$$f_3(A) = 1.189A + 1.3275A(A-0.3) + 4.7745A(A-0.3)(A-0.7) \qquad (3.24)$$

Suppose we wish to estimate $f(0.45)$, the time required for the solution to reach 0.45, using $f_3(0.45)$. The result is

$$\begin{aligned} f_3(0.45) &= 1.189(0.45) + 1.3275(0.45)(0.45-0.3) + 4.7745(0.45)(0.45-0.3)(0.45-0.7) \\ &= 0.5441 \end{aligned}$$

Choosing the additional data point (0.5, 0.6931) because of its proximity to the interpolant value 0.45 allows us to find the 4th order interpolating polynomial $f_4(A)$. Table 3.9 includes the additional finite divided differences calculated from the new data point. The result is

$$\begin{aligned} f_4(A) = {} & 1.189A + 1.3275A(A-0.3) + 4.7745A(A-0.3)(A-0.7) \\ & + 7.6675A(A-0.3)(A-0.7)(A-0.9) \end{aligned} \qquad (3.25)$$

and the improved estimate of $f(0.45)$ is therefore

$$\begin{aligned} f_4(0.45) &= f_3(0.45) + 7.6675A(A-0.3)(A-0.7)(A-0.9) \\ &= 0.5441 + 7.6675(0.45)(0.45-0.3)(0.45-0.7)(0.45-0.9) \\ &= 0.5441 + 0.0582 \\ &= 0.6023 \end{aligned}$$

Based on reasoning analogous to that used to obtain $R_1(x)$ in Equation (3.22), the error term $R_3(0.45)$ is approximated as

$$\begin{aligned} R_3(0.45) &\approx f_4(0.45) - f_3(0.45) \\ &\approx 0.0582 \end{aligned}$$

It is easily shown that the true function relating T and A is given by

$$T = f(A) = -\ln(1-A) \tag{3.26}$$

Figure 3.3 shows two graphs. Each one contains the complete set of data points from Table 3.8 and the true function $f(A)$. The top graph contains the third order polynomial $f_3(A)$ and the four data points used to find it are shown with an asterisk. The lower graph contains the fourth order polynomial $f_4(A)$. The same four data points used to find $f_3(A)$ and the one additional point used to find $f_4(A)$ are shown as asterisks.

The true error $R_3(0.45)$ is also shown in Figure 3.3. As expected $R_3(A)$ is zero when A is any of the four data point values. The exact value for $R_3(0.45)$ is obtained as the difference between $f(0.45)$ and $f_3(0.45)$. The result is

$$\begin{aligned} R_3(0.45) &= f(0.45) - f_3(0.45) \\ &= -\ln(1-0.45) - 0.5441 \\ &= 0.0537 \end{aligned}$$

which should be compared to the estimated value of 0.0582 previously computed.

Keep in mind that $f_3(A)$, $f_4(A)$, and $R_3(A)$ are all sensitive to the choice of data points selected from Table 3.8. Knowing a priori that $f(0.45)$ is to be estimated would dictate a different choice of data points as the basis for determining an interpolating function to approximate $f(A)$. The 4 closest points to the interpolant value 0.45 are (0.3,0.3567), (0.4,0.5108), (0.5,0.6931) and (0.6,0.9163,).

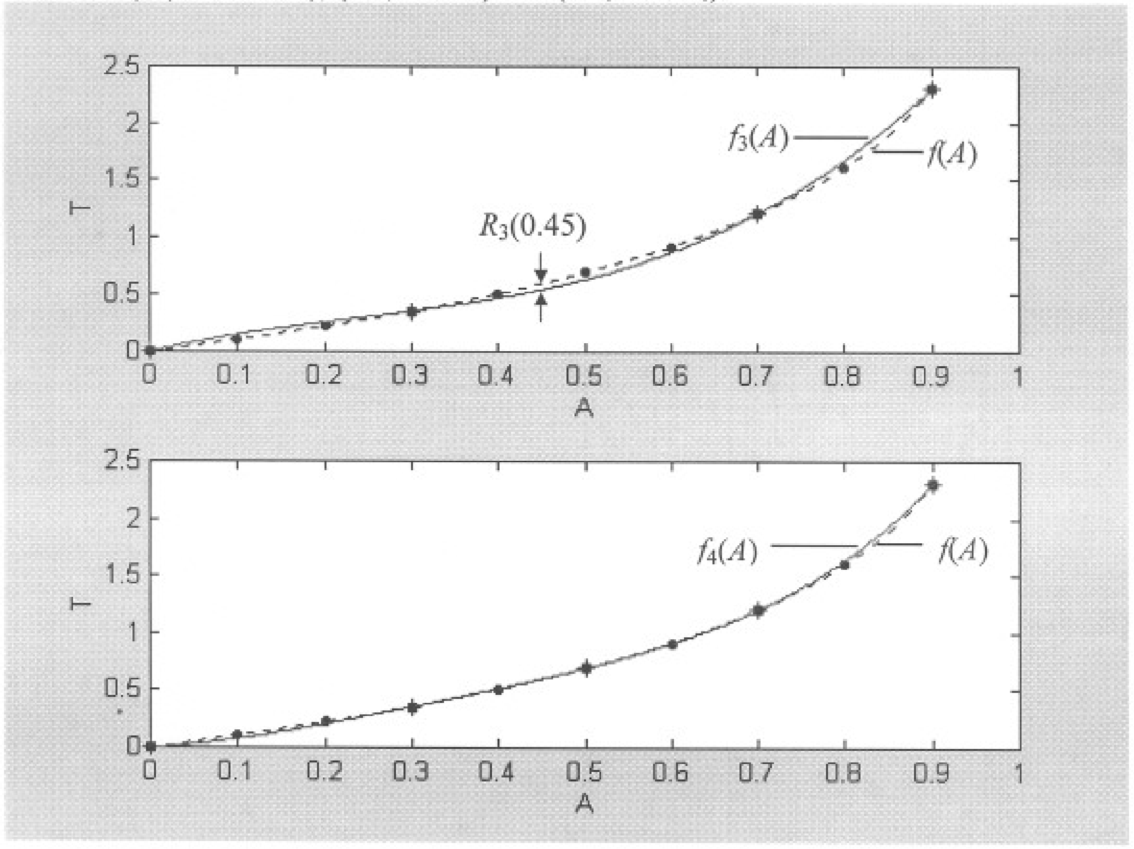


Figure 3.3 Third and Fourth Order Interpolating Polynomials and the True Function for Data in Table 3.8

At this point it is necessary to introduce a new function $f[x, x_1, x_0]$, similar to a second order divided difference, defined as

$$f[x,x_1,x_0] = \frac{f[x,x_1]-f[x_1,x_0]}{x-x_0} \tag{3.27}$$

$$= \frac{\frac{f(x)-f(x_1)}{x-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x-x_0} \tag{3.27a}$$

However, unlike the second order finite divided differences previously encountered, its first argument x is treated as a variable. Of course, once x assumes a numerical value, $f[x, x_1, x_0]$ becomes an ordinary second order divided difference. The importance of $f[x, x_1, x_0]$ is its relationship to $R_1(x)$ which is presented here and left as an exercise for later.

Section 4 Lagrange Interpolating Polynomials

In the previous sections we encountered two different ways of representing the unique nth order (or lower) polynomial required to pass through a given set of $n+1$ points. Yet another way of writing the polynomial, constrained in the same fashion, is presented here. It is referred to as Lagrange's form of the interpolating polynomial.

Once again, we assume the existence of a set of data points (x_i, y_i), $i = 0, 1, \ldots, n$ obtained from a function $f(x)$ so that $y_i = f(x_i)$, $i = 0, 1, \ldots, n$. A suitable function for interpolation $I(x)$ is expressible as

$$I(x) = \sum_{i=0}^{n} L_i(x) \cdot f(x_i) \tag{4.1}$$

$$= L_0(x) \cdot f(x_0) + L_1(x) \cdot f(x_1) + \ldots\ldots + L_n(x) \cdot f(x_n) \tag{4.1a}$$

The functions $L_i(x)$, $i = 0, 1, \ldots, n$ are chosen to satisfy

$$L_i(x) = \begin{cases} 0 & x = x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \\ 1 & x = x_i \end{cases} \tag{4.2}$$

Before we actually define the $L_i(x)$ functions, let's be certain we understand the implications of Equations (4.1) and (4.2). The best way to accomplish this is simply to choose a value for "n" and write out the resulting equations. Suppose we have the four data points $[x_i, f(x_i)]$, $i = 0, 1, 2, 3$. From Equations (4.1) with $n = 3$, the interpolating function $I(x)$ becomes

$$I(x) = \sum_{i=0}^{3} L_i(x) \cdot f(x_i) \tag{4.3}$$

$$= L_0(x) \cdot f(x_0) + L_1(x) \cdot f(x_1) + L_2(x) \cdot f(x_2) + L_3(x) \cdot f(x_3) \tag{4.3a}$$

and it remains to be shown that $I(x)$ is identical to $f(x)$ when x is any one of the four data points. Evaluating $I(x)$ at x_0, x_1, x_2 and x_3,

$$I(x_0) = L_0(x_0) \cdot f(x_0) + L_1(x_0) \cdot f(x_1) + L_2(x_0) \cdot f(x_2) + L_3(x_0) \cdot f(x_3) \tag{4.4}$$

$$I(x_1) = L_0(x_1) \cdot f(x_0) + L_1(x_1) \cdot f(x_1) + L_2(x_1) \cdot f(x_2) + L_3(x_1) \cdot f(x_3) \tag{4.4a}$$

$$I(x_2) = L_0(x_2)\cdot f(x_0) + L_1(x_2)\cdot f(x_1) + L_2(x_2)\cdot f(x_2) + L_3(x_2)\cdot f(x_3) \qquad (4.4b)$$

$$I(x_3) = L_0(x_3)\cdot f(x_0) + L_1(x_3)\cdot f(x_1) + L_2(x_3)\cdot f(x_2) + L_3(x_3)\cdot f(x_3) \qquad (4.4c)$$

According to Equation (4.2), $L_0(x_0) = 1$ and $L_1(x_0) = L_2(x_0) = L_3(x_0) = 0$. Equation (4.4) is simplified as shown below.

$$\begin{aligned} I(x_0) &= L_0(x_0)\cdot f(x_0) + L_1(x_0)\cdot f(x_1) + L_2(x_0)\cdot f(x_2) + L_3(x_0)\cdot f(x_3) \\ &= 1\cdot f(x_0) + 0\cdot f(x_1) + 0\cdot f(x_2) + 0\cdot f(x_3) \\ &= f(x_0) \end{aligned}$$

By the same reasoning, $I(x_1) = f(x_1)$, $I(x_2) = f(x_2)$, and $I(x_3) = f(x_3)$ which means the interpolating function $I(x)$ passes through the given set of data points.

The analytical form of $I(x)$ depends on the functions $L_i(x)$, $i = 0,1, \ldots, n$ that satisfy Equation (4.2). They are called Lagrange coefficient polynomials and are defined as follows:

$$L_i(x) = \prod_{j=0,1,\ldots,i-1,i+1,\ldots,n} \left(\frac{x - x_j}{x_i - x_j}\right) \qquad i = 0, 1, 2, \ldots, n \qquad (4.5)$$

The symbol $\prod$ in Equation (4.5) is a symbolic notation for multiplication in the same way the symbol $\sum$ denotes summation of its arguments. To better understand Equation (4.5), $L_i(x)$ are expressed in a more explicit form. Given below are expressions for $L_1(x)$ through $L_3(x)$ when there are four data points ($n = 3$).

$$i = 0, \quad L_0(x) = \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)} \qquad (4.6)$$

$$i = 1, \quad L_1(x) = \frac{(x - x_0)(x - x_2)(x - x_3)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)} \qquad (4.6a)$$

$$i = 2, \quad L_2(x) = \frac{(x - x_0)(x - x_1)(x - x_3)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)} \qquad (4.6b)$$

$$i = 3, \quad L_3(x) = \frac{(x - x_0)(x - x_1)(x - x_2)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)} \qquad (4.6c)$$

Notice that each Lagrange coefficient polynomial in Equation (4.6) is a third order polynomial as a result of the x^3 term in the numerator. For the general case when there are n+1 data points, the Lagrange coefficient polynomials L_i(x) in Equations (4.1) are nth order polynomials and therefore so is the interpolating function $I(x)$. Henceforth we shall represent the interpolating polynomial $I(x)$ in Equation (4.1) by $f_n(x)$ and refer to it as the Lagrange interpolating polynomial.

The advantage of Lagrange interpolation in comparison with the standard polynomial form (Equation 2.1) or the Newton divided-difference representation (Equation 3.1) is its simplicity. That is, the Lagrange interpolating polynomial can be determined without need of solving a system of simultaneous equations or performing repetitive calculations as in the case of Newton interpolating polynomials where a table of divided differences is required. Owing to the manner in which the Lagrange coefficient polynomials are defined in Equation (4.5), the Lagrange interpolating polynomial is written by inspection of the data points using Equation (4.1).

We demonstrate the procedure in the following example.

Example 4.1

The measured voltage as a function of time across the terminals of a 5 ohm load with three different types of batteries, each with a nominal rating of 1.5 volts, is tabulated below.

t (hours)	v (volts)		
	Rechargeable Ni-Cd 1st Charge	Rechargeable Alkaline 1st Charge	Rechargeable Alkaline 2nd Charge
0	1.40	1.40	1.35
1	1.30	1.17	1.15
2	1.00	1.10	1.05
3	0.40	1.05	1.00
4	0.05	0.90	0.40

Table 4.1 Voltage and Elapsed Time for Rechargeable Batteries

The Lagrange interpolating polynomial for each type of battery is obtained directly from the table. For the Ni-Cd battery, it is

$$f_4(t) = \frac{(t-t_1)(t-t_2)(t-t_3)(t-t_4)}{(t_0-t_1)(t_0-t_2)(t_0-t_3)(t_0-t_4)}\cdot v(t_0) + \frac{(t-t_0)(t-t_2)(t-t_3)(t-t_4)}{(t_1-t_0)(t_1-t_2)(t_1-t_3)(t_1-t_4)}\cdot v(t_1)$$

$$+ \quad \frac{(t-t_0)(t-t_1)(t-t_3)(t-t_4)}{(t_2-t_0)(t_2-t_1)(t_2-t_3)(t_2-t_4)} \cdot v(t_2) + \frac{(t-t_0)(t-t_1)(t-t_2)(t-t_4)}{(t_3-t_0)(t_3-t_1)(t_3-t_2)(t_3-t_4)} \cdot v(t_3)$$

$$+ \quad \frac{(t-t_0)(t-t_1)(t-t_2)(t-t_3)}{(t_4-t_0)(t_4-t_1)(t_4-t_2)(t_4-t_3)} \cdot v(t_4) \qquad (4.7)$$

$$f_4(t) = \frac{(t-1)(t-2)(t-3)(t-4)}{(0-1)(0-2)(0-3)(0-4)} \cdot 1.40 + \frac{t(t-2)(t-3)(t-4)}{(1-0)(1-2)(1-3)(1-4)} \cdot 1.30$$

$$+ \frac{t(t-1)(t-3)(t-4)}{(2-0)(2-1)(2-3)(2-4)} \cdot 1.00 + \frac{t(t-1)(t-2)(t-4)}{(3-0)(3-1)(3-2)(3-4)} \cdot 0.40$$

$$+ \frac{t(t-1)(t-2)(t-3)}{(4-0)(4-1)(4-2)(4-3)} \cdot 0.05 \qquad (4.8)$$

$$f_4(t) = \tfrac{1.4}{24}(t-1)(t-2)(t-3)(t-4) - \tfrac{1.3}{6}t(t-2)(t-3)(t-4) + \tfrac{1.00}{4}t(t-1)(t-3)(t-4)$$

$$- \tfrac{0.40}{6}t(t-1)(t-2)(t-4) + \tfrac{0.05}{24}t(t-1)(t-2)(t-3) \qquad (4.9)$$

The same procedure is used to obtain the fourth order Lagrange interpolating polynomials for the two alkaline batteries. Figure 4.1 shows the data points and the fourth order interpolating polynomial for each battery.

It is generally advisable to begin with lower order polynomials for interpolation and then increase the order in a systematic fashion until the results are acceptable. In other words, a reduced subset of data points is chosen based on the expected range of the interpolant. Suppose we have n+1 data points measured from a function $f(x)$ where the range of x is from x_{min} to x_{max}. Assuming interpolation over the entire interval is required with a single polynomial, we might choose a third or possibly fourth order polynomial (assuming n>4) based on a judicious choice of four or five points that include both extremes. Visual inspection of the polynomial and its relationship to the entire set of data points will determine if a higher order polynomial is required and which additional data point should be included.

When an additional data point is required, the higher order polynomial will generally exhibit more curvature than the previous one throughout the interpolation interval. For example, Figure 4.2 shows polynomials of order n =1, 2, 3, and 4 required to pass through data sets of two, three, four, and five points respectively

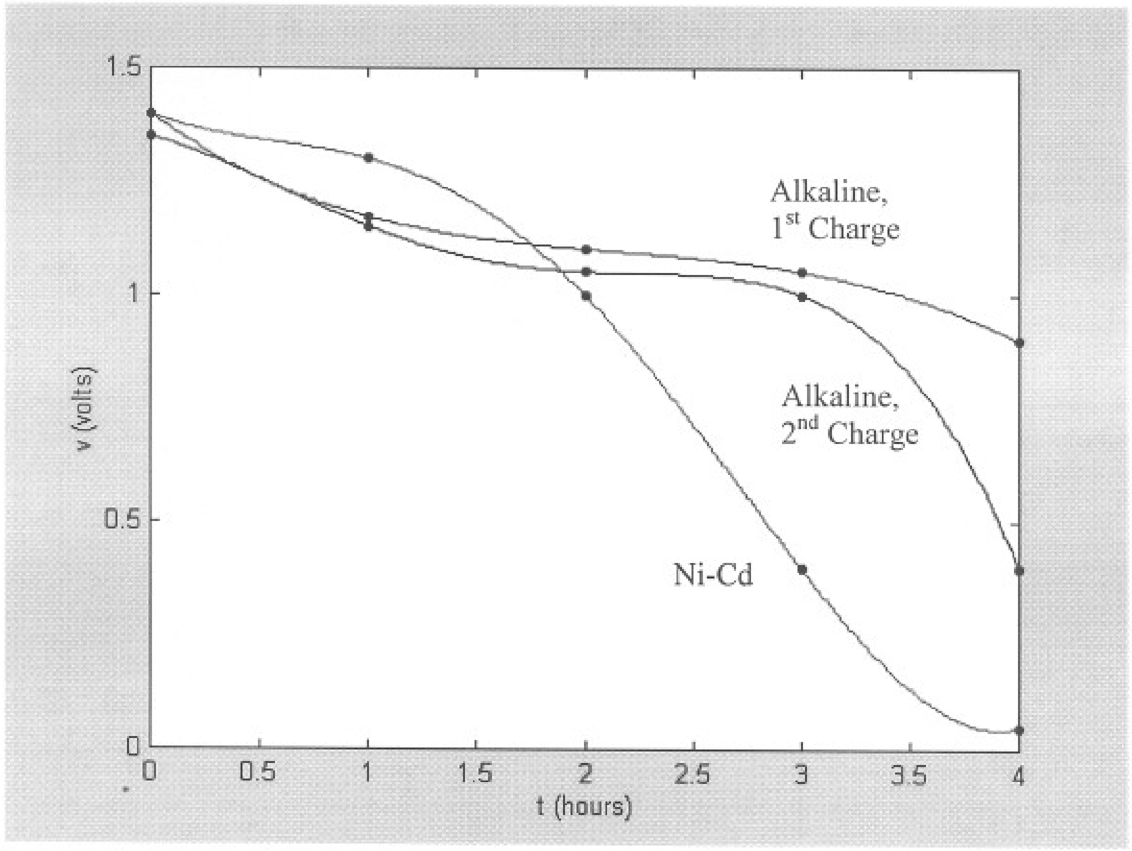


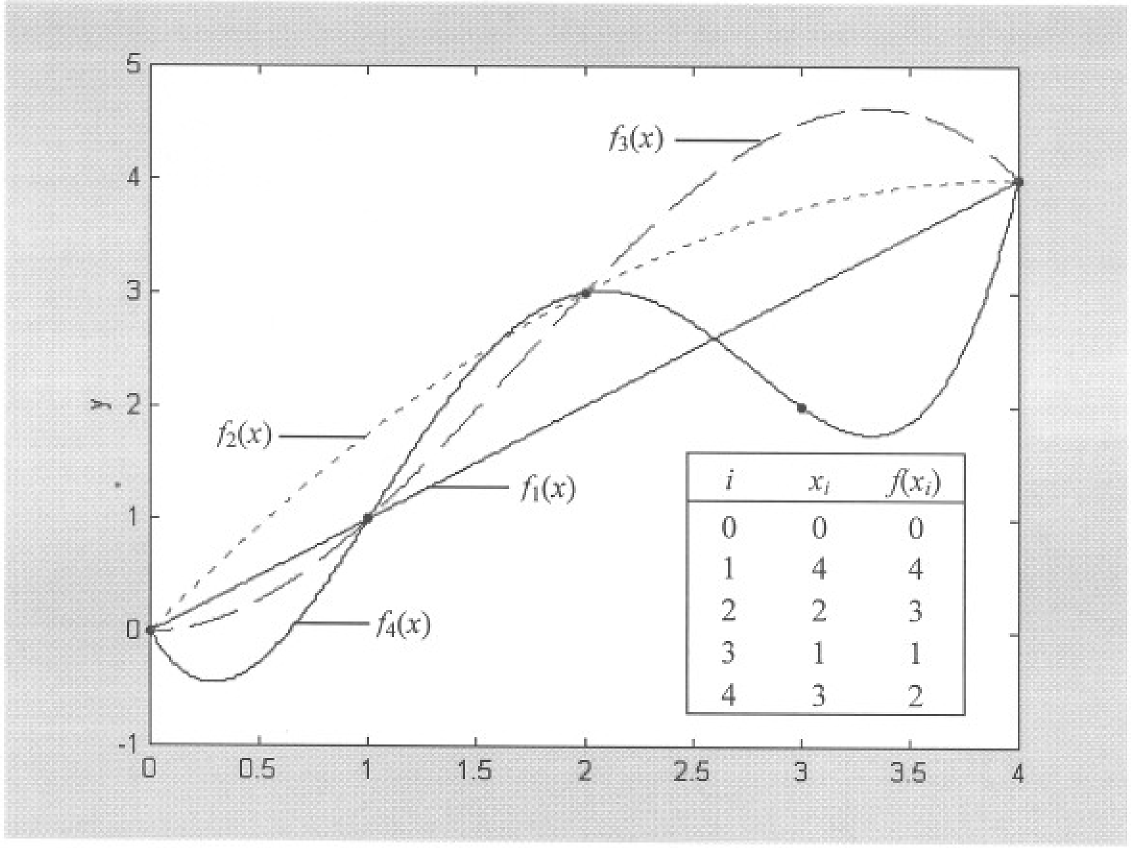
Figure 4.1 Fourth Order Interpolating Polynomials For Battery Data in Table 4.1

It's clear that the quadratic polynomial $f_2(x)$ exhibits more curvature than the linear function $f_1(x)$ which of course has none. This is possible due to the presence of the x^2 term in $f_2(x)$. Similarly $f_3(x)$ curves more than $f_2(x)$ due to the additional point $[x_3, f(x_3)]$ and the fourth order term x^4 in $f_4(x)$ accounts for the added curvature compared to $f_3(x)$ over the interval (0,4).

With an nth order interpolating polynomial in standard form, Equation (2.1), or the Newton divided-difference form, Equation (3.1), there is only one high order term, i.e. a single term with x^n. This is in contrast to the Lagrange form of the interpolating polynomial, Equation (4.1), in which each term of the overall expression is an nth order polynomial. If the order of the interpolating polynomial is to be increased from n to n+1 by including an additional data point, each Lagrange coefficient polynomial in Equation (4.5) increases in order from n to n+1 as well. Consequently, the entire Lagrange interpolating polynomial must be recomputed.

Despite the fact interpolating polynomials in standard form contain a single high order term, an extra data point requires recalculation of all the coefficients a_i, $i = 0,1,2, \ldots, n$ in addition to the new coefficient a_{n+1}. The system of equations to be solved was considered in Section 3.2 and enumerated in matrix form in Equation (2.3). The Vandermonde matrix and the column vectors of coefficients and function values in Equation (2.3) are $(n+1) \times (n+1)$, $n \times 1$, and $n \times 1$, respectively.

The Newton divided-difference interpolating polynomial (of the three forms considered) minimizes the computational effort necessary to accommodate an additional data point. As we pointed out in Section 3.3, the Newton divided-difference interpolating polynomial $f_n(x)$ in Equation (3.1) is formulated in such a way that the coefficients b_i, $i = 1,2,3,\ldots,n$ do not change; only b_{n+1} is calculated using the expanded table of finite divided differences.



i	x_i	$f(x_i)$
0	0	0
1	4	4
2	2	3
3	1	1
4	3	2

Figure 4.2 Increasing Curvature of a Polynomial as a Function of its Order

As with any interpolation method based on the use of approximation functions, there is an error term present which accounts for the difference between the true function value (usually unknown) and the interpolated value. Using an nth order Lagrange interpolating polynomial $f_n(x)$ to estimate values of $f(x)$, the error term $R_n(x)$ satisfies

$$f(x) = f_n(x) + R_n(x) \tag{4.10}$$

which is identical to Equation (3.31) in Section 3 when $f_n(x)$ was an nth order Newton divided-difference interpolating polynomial. Since $f_n(x)$ is unique for a given set of n+1 data points, much of the error analysis presented in Section 3 is applicable to Lagrange interpolating polynomials. Thus, an estimate of the error $R_n(x)$ is still the difference $f_{n+1}(x)$ - $f_n(x)$, where an additional data point is required to evaluate $f_{n+1}(x)$. The real advantage of Newton divided-difference interpolating polynomials is the reduced computational effort (compared with the Lagrange form) to obtain $f_{n+1}(x)$ when $f_n(x)$ is already computed.

Example 4.2

In Example 4.1, estimate the error incurred by the use of a fourth order Lagrange interpolating polynomial for approximating the voltage of a Ni-Cd battery after 1.5 hours. Assume an additional data point is available, namely $t = 2.5$ hours, $v = 0.7$ volts. From Equation (4.9),

$$\begin{aligned} f_4(1.5) &= \tfrac{1.4}{24}(1.5-1)(1.5-2)(1.5-3)(1.5-4) - \tfrac{1.3}{6}1.5(1.5-2)(1.5-3)(1.5-4) \\ &\quad + \tfrac{1.00}{4}1.5(1.5-1)(1.5-3)(1.5-4) - \tfrac{0.40}{6}1.5(1.5-1)(1.5-2)(1.5-4) \\ &\quad + \tfrac{0.05}{24}1.5(1.5-1)(1.5-2)(1.5-3) \\ f_4(1.5) &= 1.1965 \end{aligned}$$

Supplementing Table 4.1 with the new data point and determining the fifth order Lagrange interpolating polynomial yields,

$$\begin{aligned} f_5(t) &= \frac{(t-1)(t-2)(t-3)(t-4)(t-2.5)}{(0-1)(0-2)(0-3)(0-4)(0-2.5)}\cdot 1.40 + \frac{t(t-2)(t-3)(t-4)(t-2.5)}{(1-0)(1-2)(1-3)(1-4)(1-2.5)}\cdot 1.30 \qquad (4.11) \\ &+ \frac{t(t-1)(t-3)(t-4)(t-2.5)}{(2-0)(2-1)(2-3)(2-4)(2-2.5)}\cdot 1.00 + \frac{t(t-1)(t-2)(t-4)(t-2.5)}{(3-0)(3-1)(3-2)(3-4)(3-2.5)}\cdot 0.40 \\ &+ \frac{t(t-1)(t-2)(t-3)(t-2.5)}{(4-0)(4-1)(4-2)(4-3)(4-2.5)}\cdot 0.05 + \frac{t(t-1)(t-2)(t-3)(t-4)}{(2.5-0)(2.5-1)(2.5-2)(2.5-3)(2.5-4)} 0.70 \end{aligned}$$

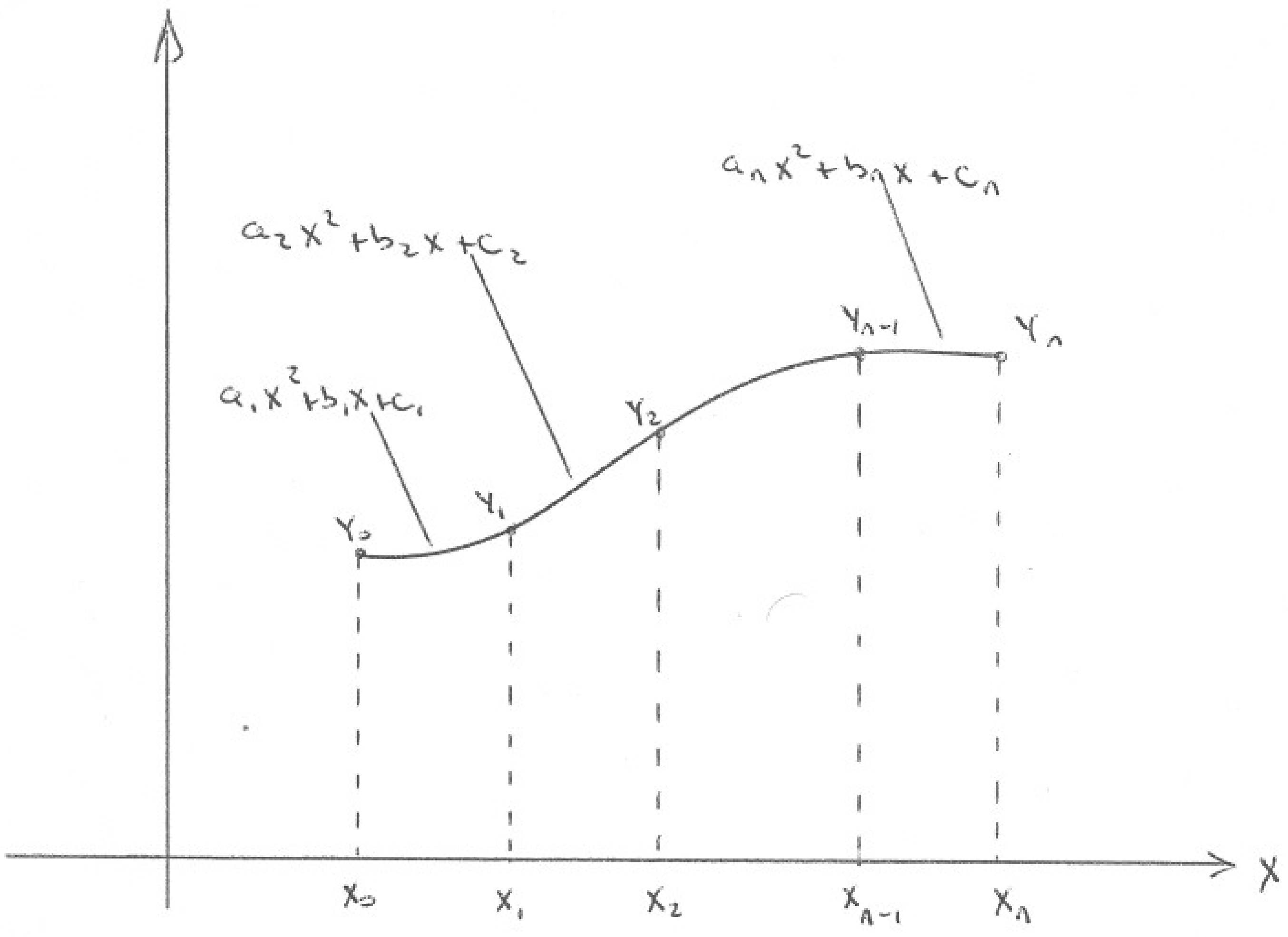
$R_4(1.5)$ is the difference between the true (unknown) function value $f(1.5)$ and the fourth order interpolating polynomial estimate $f_4(1.5)$, i.e.

$$R_4(1.5) = f(1.5) - f_4(1.5)$$

We can estimate $R_4(1.5)$ using

$$\begin{aligned} R_4(1.5) &\approx f_5(1.5) - f_4(1.5) \\ &\approx 1.2148 - 1.1965 \\ &\approx 0.0184 \end{aligned}$$

Spline Interpolation



Given $n+1$ data pts (x_i, y_i), $i = 0, 1, 2, \ldots, n$ its possible to fit low order polynomials thru pairs of consecutive data pts. For quadratic spline

$$f(x) = \begin{cases} a_1 x^2 + b_1 x + c_1 & x_0 \le x \le x_1 \\ a_2 x^2 + b_2 x + c_2 & x_1 \le x \le x_2 \\ \vdots & \\ a_n x^2 + b_n x + c_n & x_{n-1} \le x \le x_n \end{cases}$$

There are a total of $3n$ constants $\{a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n\}$ to be determined. Therefore we need $3n$ equations to solve for the $3n$ unknowns. The following equations apply:

I. Function Values Equal at Interior Pts

$$\left.\begin{aligned} a_1 x_1^2 + b_1 x_1 + c_1 &= y_1 \\ a_2 x_1^2 + b_2 x_1 + c_2 &= y_1 \end{aligned}\right\} \text{Equality at } x = x_1$$

$$\left.\begin{aligned} a_2 x_2^2 + b_2 x_2 + c_2 &= y_2 \\ a_3 x_2^2 + b_3 x_2 + c_3 &= y_2 \end{aligned}\right\} \text{Equality at } x = x_2$$

$$\vdots$$

$$\left.\begin{aligned} a_{n-1} x_{n-1}^2 + b_{n-1} x_{n-1} + c_{n-1} &= y_{n-1} \\ a_n x_{n-1}^2 + b_n x_{n-1} + c_n &= y_{n-1} \end{aligned}\right\} \text{Equality at } x = x_{n-1}$$

This gives $2(n-1) = 2n-2$ equations!

II. First and Last Quadratics Pass Thru End Points

$$a_1 x_0^2 + b_1 x_0 + c_1 = y_0$$

$$a_n x_n^2 + b_n x_n + c_n = y_n$$

This gives 2 equations!

III. First Derivatives Equal at Interior Pts

$2a_1x_1 + b_1 = 2a_2x_1 + b_2$ Equality at $x = x_1$

$2a_2x_2 + b_2 = 2a_3x_2 + b_3$ Equality at $x = x_2$

$\vdots$

$2a_{n-1}x_{n-1} + b_{n-1} = 2a_nx_{n-1} + b_n$ Equality at $x = x_{n-1}$

This gives $n-1$ equations !

Total number of equations $= (2n-2) + 2 + (n-1)$

$= 3n - 1$

The last equation comes from setting the 2nd derivative equal to zero at one of the end points. Choosing x_0 and setting the 2nd derivative to zero gives $a_2 = 0$ for the last equation.

$$f(v) = \begin{cases} b_1 v + c_1 & 20 \le v \le 35 \\ a_2 v^2 + b_2 v + c_2 & 35 \le v \le 50 \\ a_3 v^2 + b_3 v + c_3 & 50 \le v \le 65 \\ a_4 v^2 + b_4 v + c_4 & 65 \le v \le 80 \end{cases}$$

V	D
20	42
35	92
50	123
65	295
80	464

I

At $v_1 = 35 \Rightarrow$ $b_1(35) + c_1 = 92$

$a_2(35)^2 + b_2(35) + c_2 = 92$

At $v_2 = 50 \Rightarrow$ $a_2(50)^2 + b_2(50) + c_2 = 173$

$a_3(50)^2 + b_3(50) + c_3 = 173$

At $v_3 = 65 \Rightarrow$ $a_3(65)^2 + b_3(65) + c_3 = 295$

$a_4(65)^2 + b_4(65) + c_4 = 295$

II

At $v_0 = 20 \Rightarrow$ $b_1(20) + c_1 = 42$

$v_4 = 80 \Rightarrow$ $a_4(80)^2 + b_4(80) + c_4 = 464$

III

At $v_1 = 35 \Rightarrow$ $+b_1 = 2a_2(35) + b_2$

$50 \Rightarrow$ $2a_2(50) + b_2 = 2a_3(50) + b_3$

$65 \Rightarrow$ $2a_3(65) + b_3 = 2a_4(65) + b_4$

There are $3n - 1 = 3(4) - 1 = 11$ equations in the 11 unknowns $a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_1, c_2, c_3, c_4$

$A\,\underline{x} = \underline{b}$

$$A = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\left[\begin{array}{ccccccccccc}
b_1 & c_1 & a_2 & b_2 & c_2 & a_3 & b_3 & c_3 & a_4 & b_4 & c_4 \\
35 & 1 & & & & & & & & & \\
 & & (35)^2 & 35 & 1 & & & & & & \\
 & & (50)^2 & 50 & 1 & & & & & & \\
 & & & & & (50)^2 & 50 & 1 & & & \\
 & & & & & (65)^2 & 65 & 1 & & & \\
 & & & & & & & & (65)^2 & 65 & 1 \\
20 & 1 & & & & & & & & & \\
 & & & & & & & & (80)^2 & 80 & 1 \\
1 & & -70 & -1 & & & & & & & \\
 & & 100 & 1 & & -100 & -1 & & & & \\
 & & & & & 130 & 1 & & -130 & -1 &
\end{array}\right]$$

$$\underline{x} = \begin{bmatrix} b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ a_3 \\ b_3 \\ c_3 \\ a_4 \\ b_4 \\ c_4 \end{bmatrix} \qquad \underline{b} = \begin{bmatrix} 92 \\ 92 \\ 17.3 \\ 17.3 \\ 29.5 \\ 29.5 \\ 42 \\ 46.4 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Quadratic Spline Interpolation

```
A=[35 1 0 0 0 0 0 0 0 0 0;
   0 0 1225 35 1 0 0 0 0 0 0;
   0 0 2500 50 1 0 0 0 0 0 0;
   0 0 0 0 0 2500 50 1 0 0 0;
   0 0 0 0 0 4225 65 1 0 0 0;
   0 0 0 0 0 0 0 0 4225 65 1;
   20 1 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 6400 80 1;
   1 0 -70 -1 0 0 0 0 0 0 0;
   0 0 100 1 0 -100 -1 0 0 0 0;
   0 0 0 0 0 130 1 0 -130 -1 0;]

b=[92;92;173;173;295;295;42;464;0;0;0]

AInverse=inv(A);
x=AInverse*b;

w=2;
for m=3:3:9
   fprintf('a(%1.0f) = ',w)
   a(w)=x(m);
   disp(a(w))
   w=w+1;
end

w=1;
for m=1:3:10
   fprintf('b(%1.0f) = ',w)
   b(w)=x(m);
   disp(b(w))
   w=w+1;
end

w=1;
for m=2:3:11
   fprintf('c(%1.0f) = ',w)
   c(w)=x(m);
   disp(c(w))
   w=w+1;
end

V1=linspace(20,35,300);
FV1=b(1).*V1 +c(1);
plot(V1,FV1,'k')

hold on
grid on

V1p=20;
FV1p=b(1).*V1p +c(1);
plot(V1p, FV1p, 'kp')

V2=linspace(35,50,300);
FV2=a(2).*(V2.^2)+b(2).*V2+c(2);
plot(V2,FV2,'k')

V2p=35;
FV2p=a(2).*(V2p.^2)+b(2).*V2p+c(2);
plot(V2p,FV2p,'kp')

V3=linspace(50,65,300);
FV3=a(3).*(V3.^2)+b(3).*V3+c(3);
plot(V3,FV3,'k')

V3p=50;
FV3p=a(3).*(V3p.^2)+b(3).*V3p+c(3);
plot(V3p,FV3p,'kp')

V4=linspace(65,80,300);
FV4=a(4).*(V4.^2)+b(4).*V4+c(4);
plot(V4,FV4,'k')
```

```
V4p=65;
FV4p=a(4).*(V4p.^2)+b(4).*V4p+c(4);
plot(V4p,FV4p,'kp')

V5p=80;
FV5p=a(4).*(V5p.^2)+b(4).*V5p+c(4);
plot(V5p,FV5p,'kp')

xlabel('Speed (V): mph')
ylabel('Stopping Distance (D): ft')
title('Estimated Stopping Distances')
```

```
» myspline

A =

  Columns 1 through 6

          35           1           0           0           0           0
           0           0        1225          35           1           0
           0           0        2500          50           1           0
           0           0           0           0           0        2500
           0           0           0           0           0        4225
           0           0           0           0           0           0
          20           1           0           0           0           0
           0           0           0           0           0           0
           1           0         -70          -1           0           0
           0           0         100           1           0        -100
           0           0           0           0           0         130

  Columns 7 through 11

           0           0           0           0           0
           0           0           0           0           0
           0           0           0           0           0
          50           1           0           0           0
          65           1           0           0           0
           0           0        4225          65           1
           0           0           0           0           0
           0           0        6400          80           1
           0           0           0           0           0
          -1           0           0           0           0
           1           0        -130          -1           0


b =

    92
    92
   173
   173
   295
   295
    42
   464
     0
     0
     0

a(2) =      0.1378

a(3) =      0.0444

a(4) =      0.1644

b(1) =      3.3333

b(2) =     -6.3111

b(3) =      3.0222

b(4) =    -12.5778

c(1) =    -24.6667

c(2) =    144.1111

c(3) =    -89.2222

c(4) =    417.7778
```

A =

35	1	0	0	0	0	0	0	0	0	0
0	0	1225	35	1	0	0	0	0	0	0
0	0	2500	50	1	0	0	0	0	0	0
0	0	0	0	0	2500	50	1	0	0	0
0	0	0	0	0	4225	65	1	0	0	0
0	0	0	0	0	0	0	0	4225	65	1
20	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	6400	80	1
1	0	-70	-1	0	0	0	0	0	0	0
0	0	100	1	0	-100	1	0	0	0	0
0	0	0	0	0	130	1	0	-130	-1	0

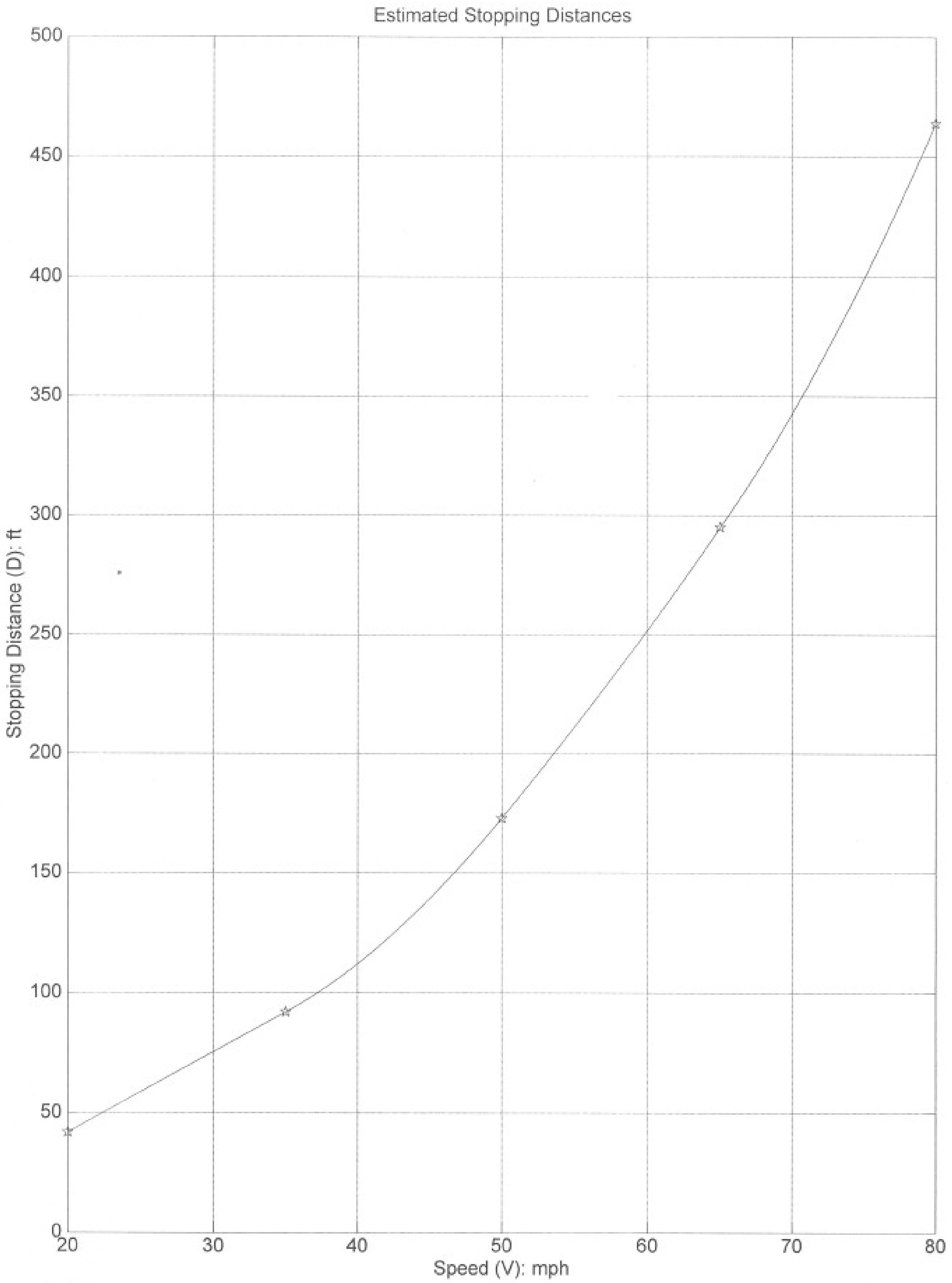
b=

92
92
173
173
295
295
42
464
0
0
0

a_1 =	0.0000
a_2 =	0.1378
a_3 =	0.0444
a_4 =	0.1644
b_1 =	3.3333
b_2 =	-6.3111
b_3 =	3.0222
b_4 =	-12.5778
c_1 =	-24.6667
c_2 =	144.1111
c_3 =	-89.2222
c_4 =	417.7778

<u>f(x) =</u>

3.3333V - 24.6667	20 <= V <= 35
0.1378V2 - 6.3111V + 144.1111	35 <= V <= 50
0.0444V2 + 3.0222V - 89.2222	50 <= V <= 65
0.1644V2 - 12.5788V + 417.7778	65 <= V <= 80



Estimated Stopping Distances
Stopping Distance (D): ft
Speed (V): mph
0
50
100
150
200
250
300
350
400
450
500
20
30
40
50
60
70
80