CHAPTER 9

Curve Fitting: Interpolation

9.1 INTRODUCTION

Scientists and engineers are often faced with the task of estimating the value of dependent variable y for an intermediate value of the independent variable x, given a table of discrete data points (x_i, y_i), $i = 0,1,...n$. This task can be accomplished by constructing a function $y(x)$ that will pass through the given set of points and then evaluating $y(x)$ for the specified value of x. The process of construction of $y(x)$ to fit a table of data points is called *curve fitting*. A table of data may belong to one of the following two categories:

1. *Table of values of well-defined functions:* Examples of such tables are logarithmic tables, trigonometric tables, interest tables, steam tables, etc.
2. *Data tabulated from measurements made during an experiment:* In such experiments, values of the dependent variable are recorded at various values of the independent variable. There are numerous examples of such experiments—the relationship between stress and strain on a metal strip, relationship between voltage applied and speed of a fan, relationship between time and temperature raise in heating a given volume of water, relationship between drag force and velocity of a falling body, etc., can be tabulated by suitable experiments.

In category 1, the table values are accurate because they are obtained from well-behaved functions. This is not the case in category 2 where the relationship between the variables is not well-defined. Accordingly, we have two approaches for fitting a curve to a given set of data points.

In the first case, the function is constructed such that it passes through all the data points. This method of constructing a function and estimating values at non-tabular points is called *interpolation*. The functions are known as *interpolation polynomials*.

In the second case, the values are not accurate and, therefore, it will be meaningless to try to pass the curve through every point. The best strategy would be to construct a single curve that would represent the general trend of the data, without necessarily passing through the individual points. Such functions are called *approximating functions*. One popular approach for finding an approximate function to fit a given set of experimental data is called *least-squares regression*. The approximating functions are known as *least-squares polynomials*.

Figure 9.1 shows an approximate linear function and an interpolation polynomial for a set of data. Note that although the interpolation polynomial passes through all the points, the curve oscillates widely at the end and beyond the range of data. The linear approximating curve which does not pass through any of the points appears to represent the trend of data adequately. The straight line gives a much better idea of likely values beyond the table points.

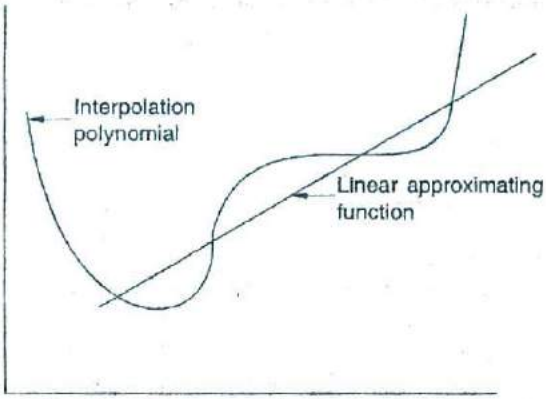

Fig. 9.1 Curve fitting to a set of points

In this chapter, we discuss various methods of interpolation. They include:

1. Lagrange interpolation
2. Newton's interpolation
3. Newton-Gregory forward interpolation
4. Spline interpolation

Before we discuss these methods, we introduce various forms of polynomials that are used in deriving interpolation functions. Least-squares regression techniques are discussed in the next chapter.

9.2 POLYNOMIAL FORMS

The most common form of an nth order polynomial is

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \tag{9.1}$$

This form, known as the *power form*, is very convenient for differentiating and integrating the polynomial function and, therefore, are most widely used in mathematical analysis. However, there are situations where this form has been found inadequate, as illustrated by Example 9.1.

Example 9.1

Consider the power form of $p(x)$ for $n = 1$,

$$p(x) = a_0 + a_1 x$$

Given that

$$p(100) = +3/7$$
$$p(101) = -4/7$$

obtain the linear polynomial $p(x)$ using four-digit floating point arithmetic. Verify the polynomial by substituting back the values $x = 100$ and $x = 101$.

$$p(100) = a_0 + 100\, a_1 = +0.4286$$
$$p(101) = a_0 + 101\, a_1 = -0.5714$$

Then, we get

$$a_1 = -1$$
$$a_0 = 100.4 \text{ (only four significant digits)}$$

Therefore,

$$p(x) = 100.4 - x$$

using this polynomial, we obtain

$$p(100) = 0.4$$
$$p(101) = -0.6$$

Compare these results with the original values of $p(100)$ and $p(101)$. We have lost three decimal digits.

Example 9.1 shows that the polynomials obtained using the power form may not always produce accurate results. In order to overcome such problems, we have alternative forms of representing a polynomial. One of them is the *shifted power form* as shown below:

$$p(x) = a_0 + a_1 (x - C) + a_2 (x - C)^2 + \ldots + a_n (x - C)^n \tag{9.2}$$

where C is a point somewhere in the interval of interest. This form of representation significantly improves the accuracy of the polynomial evaluation. This is illustrated by Example 9.2.

Example 9.2

Repeat Example 9.1 using the shifted power form and four-digit arithmetic.

Shifted power form of first order $p(x)$ is

$$p(x) = a_0 + a_1 (x - C)$$

Let us choose the centre C as 100. Then

$$p(x) = a_0 + a_1 (x - 100)$$

This gives,

$$p(100) = a_0 = 3/7 = 0.4286$$

$$p(101) = 0.4286 + a_1 (101 - 100) = -0.5714$$

$$a_1 = -1$$

Thus the linear polynomial becomes

$$p(x) = 0.4286 - (x - 100)$$

Using this polynomial, we obtain

$$p(100) = 0.4286$$

$$p(101) = -0.5714$$

Note the improvement in the results.

Note that Eq. (9.2) is the *Taylor expansion* of $p(x)$ around the point C, when the coefficients a_i are replaced by appropriate function derivatives. It can be easily verified that

$$a_i = \frac{p^{(i)}(C)}{i!} \quad i = 0, 1, 2, \ldots n$$

where $p^{(i)}(C)$ is the ith derivative of $p(x)$ at C.

There is a third form of $p(x)$ known as *Newton form*. This is a generalised shifted power form as shown below:

$$p(x) = a_0 + a_1 (x - C_1) + a_2 (x - C_1)(x - C_2) + a_3 (x - C_1)(x - C_2)(x - C_3) + \ldots + a_n (x - C_1)(x - C_2) \ldots (x - C_n) \tag{9.3}$$

Note that Eq. (9.3) reduces to shifted power form when $C_1 = C_2 = C_3 = \ldots = C_n$ and to simple power form when $C_i = 0$ for all i. The Newton form plays an important role in the derivation of an interpolating polynomial as seen in Section 9.5.

Polynomials can also be expressed in the form

$$p_2(x) = b_0(x - x_1)(x - x_2) + b_1(x - x_0)(x - x_2) + b_2(x - x_0)(x - x_1)$$

In general form,

$$P_n(x) = \sum_{i=0}^{n} b_i \prod_{j=0, j \neq i}^{n} (x - x_j) \tag{9.4}$$

LINEAR INTERPOLATION

The simplest form of interpolation is to approximate two data points by a straight line. Suppose we are given two points $(x_1, f(x_1))$ and $(x_2, f(x_2))$. These two points can be connected linearly as shown in Fig. 9.2. Using the concept of similar triangles, we can show that

$$\frac{f(x) - f(x_1)}{x - x_1} = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

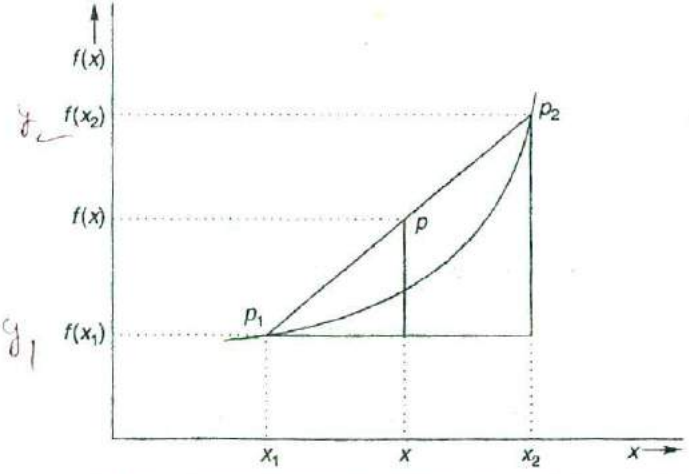

Fig. 9.2 Graphical representation of linear interpolation

Solving for $f(x)$, we get

$$f(x) = f(x_1) + (x - x_1)\frac{f(x_2) - f(x_1)}{x_2 - x_1} \tag{9.5}$$

Equation (9.5) is known as *linear interpolation formula*. Note that the term

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

represents the slope of the line. Further, note the similarity of equation (9.5) with the *Newton form* of polynomial of first-order.

$$C_1 = x_1$$

$$a_0 = f(x_1)$$

$$a_1 = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

The coefficient a_1 represents the first derivative of the function.

Example 9.3

The table below gives square roots for integers.

x	1	2	3	4	5
$f(x)$	1	1.4142	1.7321	2	2.2361

Determine the square root of 2.5

The given value of 2.5 lies between the points 2 and 3. Therefore,

$$x_1 = 2, \quad f(x_1) = 1.4142$$

$$x_2 = 3, \quad f(x_2) = 1.7321$$

Then

$$f(2.5) = 1.4142 + (2.5 - 2.0)\frac{1.7321 - 1.4142}{3.0 - 2.0}$$

$$= 1.4142 + (0.5)(0.3179)$$

$$= 1.5732$$

The correct answer is 1.5811. The difference is due to the use of a linear model to a nonlinear one.

Now, let us repeat the procedure assuming $x_1 = 2$ and $x_2 = 4$.

$$f(x_1) = 1.4142$$

$$f(x_2) = 2.0$$

Then,

$$f(2.5) = 1.4142 + (2.5 - 2.0)\frac{2.0 - 1.4142}{4.0 - 2.0}$$

$$= 1.4142 + (0.5)(0.2929)$$

$$= 1.5607$$

Notice that the error has increased from 0.0079 to 0.0204. In general, the smaller the interval between the interpolating data points, the better will be the approximation.

The results could be improved considerably by using higher-order interpolation polynomials. We shall demonstrate this in the next section.

9.4 LAGRANGE INTERPOLATION POLYNOMIAL

In this section, we derive a formula for the polynomial of degree n which takes specified values at a given set of $n + 1$ points.

Let $x_0, x_1, \ldots x_n$ denote n distinct real numbers and let $f_0, f_1, \ldots, f_n$ be arbitrary real numbers. The points $(x_0, f_0), (x_1, f_1), \ldots (x_n, f_n)$ can be imagined to be data values connected by a curve. Any function $p(x)$ satisfying the conditions

$$p(x_k) = f_k \qquad \text{for} \qquad k = 0, 1, \ldots n$$

is called an *interpolation function*. An interpolation function is, therefore, a curve that passes through the data points as pointed out in Section 9.1.

Let us consider a second-order polynomial of the form

$$\begin{aligned} p_2(x) &= b_1(x - x_0)(x - x_1) \\ &\quad + b_2(x - x_1)(x - x_2) \\ &\quad + b_3(x - x_2)(x - x_0) \end{aligned} \tag{9.6}$$

If (x_0, f_0), (x_1, f_1) and (x_2, f_2) are the three interpolating points, then we have

$$p_2(x_0) = f_0 = b_2(x_0 - x_1)(x_0 - x_2)$$

$$p_2(x_1) = f_1 = b_3(x_1 - x_2)(x_1 - x_0)$$

$$p_2(x_2) = f_2 = b_1(x_2 - x_0)(x_2 - x_1)$$

Substituting for b_1, b_2 and b_3 in Eq. (9.6), we get

$$\begin{aligned} p_2(x) &= f_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} \\ &\quad + f_1 \frac{(x - x_2)(x - x_0)}{(x_1 - x_2)(x_1 - x_0)} \\ &\quad + f_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} \end{aligned} \tag{9.7}$$

Equation (9.7) may be represented as

$$p_2(x) = f_0\, l_0(x) + f_1\, l_1(x) + f_2\, l_2(x)$$

$$= \sum_{i=0}^{2} f_i l_i(x)$$

where

$$l_i(x) = \prod_{j=0, j\neq i}^{2} \frac{(x - x_j)}{(x_i - x_j)}$$

In general, for $n+1$ points we have nth degree polynomial as

$$p_n(x) = \sum_{i=0}^{n} f_i l_i(x) \tag{9.8}$$

where

$$l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{(x - x_j)}{(x_i - x_j)} \tag{9.9}$$

Equation (9.8) is called the *Lagrange interpolation polynomial*. The polynomials $l_i(x)$ are known as *Lagrange basis polynomials*. Observe that

$$l_i(x_j) = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases}$$

Now, consider the case $n = 1$

$$l_0(x) = \frac{x - x_1}{x_0 - x_1}$$

$$l_1(x) = \frac{x - x_0}{x_1 - x_0}$$

Therefore,

$$p_1(x) = f_0 \frac{x - x_1}{x_0 - x_1} + f_1 \frac{x - x_0}{x_1 - x_0}$$

$$= \frac{f_0(x - x_1) - f_1(x - x_0)}{x_0 - x_1}$$

$$= f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_0)$$

This is the *linear interpolation formula*.

Example 9.4

Consider the problem in Example 9.3. Find the square root of 2.5 using the second order Lagrange interpolation polynomial.

Let us consider the following three points:

$$x_0 = 2, \quad x_1 = 3, \quad \text{and} \quad x_2 = 4$$

Then

$$f_0 = 1.4142, \quad f_1 = 1.7321, \quad \text{and} \quad f_2 = 2$$

For $x = 2.5$, we have

$$l_0(2.5) = \frac{(2.5 - 3.0)(2.5 - 4.0)}{(2.0 - 3.0)(2.0 - 4.0)} = 0.3750$$

$$l_1(2.5) = \frac{(2.5 - 2.0)(2.5 - 4.0)}{(3.0 - 4.0)(3.0 - 2.0)} = 0.7500$$

$$l_2(2.5) = \frac{(2.5 - 2.0)(2.5 - 3.0)}{(4.0 - 2.0)(4.0 - 3.0)} = -0.125$$

$$p_2(2.5) = (1.4142)(0.3750) + (1.7321)(0.7500) + (2.0)(-0.125)$$
$$= 0.5303 + 1.2991 - 0.250 = 1.5794$$

The error is 0.0017 which is much less than the error obtained in Example 9.3

Example 9.5

Find the Lagrange interpolation polynomial to fit the following data.

i	0	1	2	3
x_i	0	1	2	3
$e^{x_i} - 1$	0	1.7183	6.3891	19.0855

Use the polynomial to estimate the value of $e^{1.5}$.

Lagrange basis polynomials are

$$l_0(x) = \frac{(x - 1)(x - 2)(x - 3)}{(0 - 1)(0 - 2)(0 - 3)}$$

$$= \frac{x^3 - 6x^2 + 11x - 6}{-3 \times 2 \times 1}$$

$$l_1(x) = \frac{(x - 0)(x - 2)(x - 3)}{(1 - 0)(1 - 2)(1 - 3)}$$

$$= \frac{x^3 - 5x^2 + 6x}{2}$$

$$l_2(x) = \frac{(x-0)(x-1)(x-3)}{(2-0)(2-1)(2-3)}$$

$$= \frac{x^3 - 4x^2 + 3x}{-2}$$

$$l_3(x) = \frac{(x-0)(x-2)(x-3)}{(3-0)(3-1)(3-2)}$$

$$= \frac{x^3 - 3x^2 + 2x}{6}$$

$$p(1.5) = 0.9375$$

The interpolation polynomial is

$$p(x) = f_0\, l_0(x) + f_1\, l_1(x) + f_2\, l_2(x) + f_3\, l_3(x)$$

$$= 0 + \frac{1.7183\,(x^3 - 5x^2 + 6x)}{2}$$

$$+ \frac{6.3891\,(x^3 - 4x^2 + 3x)}{-2}$$

$$+ \frac{19.0856\,(x^3 - 3x^2 + 2x)}{6}$$

$$= \frac{5.0732\,x^3 - 6.3621\,x^2 + 11.5987x}{6}$$

$$= 0.8455x^3 - 1.0604x^2 + 1.9331x$$

$$p(1.5) = 3.3677$$

$$e^{1.5} = p(1.5) + 1 = 4.3677$$

Points to be noted about Lagrange polynomial:

1. It requires $2(n+1)$ multiplications/divisions and $2n+1$ additions and subtractions
2. If we want to add one more data point, we have to compute the polynomial from the beginning. It does not use the polynomial already computed. That is, $p_{k+1}(x)$ does not use $p_k(x)$ which is already available

Program LAGRAN

Program LAGRAN computes the interpolation value at a specified point, given a set of data points, using the Lagrange interpolation polynomial representation.

```
*-------------------------------------------------------*
    PROGRAM LAGRAN
*-------------------------------------------------------*
*Main program                                           *
*   This program computes the interpolation value at a  *
*   specified point, given a set of data points, using  *
*   the Lagrange interpolation representation           *
*-------------------------------------------------------*
*Functions invoked                                      *
*   NIL                                                 *
*-------------------------------------------------------*
*Subroutines used                                       *
*   NIL                                                 *
*-------------------------------------------------------*
*Variables used                                         *
*   XN  - Number of data sets                           *
*   X(I)- Data points                                   *
*   F(I)- Function values at data points                *
*   XP  - Point at which interpolation is required      *
*   FP  - Interpolated value at XP                      *
*   LF  - Lagrangian factor.                            *
*-------------------------------------------------------*
*Constants used                                         *
*   MAX - Maximum number of data points permitted       *
*-------------------------------------------------------*
      INTEGER N,MAX
      REAL X,F,FP,LF,SUM
      PARAMETER(MAX = 10)
      DIMENSION X(MAX),F(MAX)

      WRITE(*,*)     'Input number of data points(N)'
      READ(*,*) N
      WRITE(*,*)     'Input data points X(I) and Function',
     +       'values F(I)'
      WRITE(*,*) 'one set in each line'
      DO 10 I = 1,N
        READ(*,*) X(I), F(I)
10    CONTINUE

      WRITE(*,*)     'Input X value at which'
      WRITE (*,*)    'interpolation is required'
      READ(*,*) XP

      SUM = 0.0
      DO 30 I = 1,N
        LF = 1.0
        DO 20 J = 1,N
```

```
          IF (I.NE.J) THEN
            LF = LF * (XP - X(J)) / (X(I) - X(J))
          ENDIF
20      CONTINUE
        SUM = SUM + LF * F(I)
30    CONTINUE
      FP = SUM

      WRITE(*,*)
      WRITE(*,*) 'LAGRANGIAN INTERPOLATION'
      WRITE(*,*)
      WRITE(*,*) 'Interpolated Function Value'
      WRITE(*,*) 'at X = ', XP, ' is', FP
      WRITE(*,*)

      STOP
      END
* ---------------  End of main LAGRAN ------------------ *
```

Test Run Results The program was used to compute the function value at $x = 2.5$ for the following table of data points:

x	2	3	4
f	1.4142	1.7321	2.0

The results are shown below:

```
Input number of data points(N)
3
Input data points X(I) and Function values F(I)
one set in each line
2 1.4142
3 1.7321
4 2.0
Input X value at which
interpolation is required
2.5

LAGRANGIAN INTERPOLATION

Interpolated Function Value
at X = 2.5000000 is 1.5794000

Stop - Program terminated.
```

NEWTON INTERPOLATION POLYNOMIAL

We have seen that, in Lagrange interpolation, we cannot use the work that has already been done if we want to incorporate another data point

in order to improve the accuracy of estimation. It is therefore necessary to look for some other form of representation to overcome this drawback.

Let us now consider another form of polynomial known as *Newton form* which was discussed in Section 9.2. The Newton form of polynomial is

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots + a_n(x - x_0)(x - x_1) \ldots (x - x_{n-1}) \tag{9.10}$$

where the interpolation points $x_0, x_1, \ldots x_{n-1}$ act as centres.

To construct the interpolation polynomial, we need to determine the coefficients $a_0, a_1, \ldots a_n$. Let us assume that $(x_0, f_0), (x_1, f_1), \ldots (x_{n-1}, f_{n-1})$ are the interpolating points. That is,

$$p_n(x_k) = f_k \qquad k = 0, 1, \ldots n - 1$$

Now, at $x = x_0$, we have (using Eq. (9.10))

$$p_n(x_0) = \boxed{a_0 = f_0} \tag{9.11}$$

Similarly, at $x = x_1$,

$$p_n(x_1) = a_0 + a_1(x_1 - x_0) = f_1$$

Substituting for a_0 from Eq. (9.11), we get

$$\boxed{a_1 = \frac{f_1 - f_0}{x_1 - x_0}} \tag{9.12}$$

At $x = x_2$,

$$p_n(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = f_2$$

Substituting for a_0 and a_1 from Eqs. (9.11) and (9.12) and rearranging the terms, we get

$$\boxed{a_2 = \frac{[(f_2 - f_1)/(x_2 - x_1)] - [(f_1 - f_0)/(x_1 - x_0)]}{x_2 - x_0}} \tag{9.13}$$

Let us define a notation

$$f[x_k] = f_k$$

$$f[x_k, x_{k+1}] = \frac{f[x_{k+1}] - f[x_k]}{x_{k+1} - x_k}$$

$$f[x_k, x_{k+1}, x_{k+2}] = \frac{f[x_{k+1}, x_{k+2}] - f[x_k, x_{k+1}]}{x_{k+2} - x_k}$$

$$f[x_k, x_{k+1}, \ldots x_i, x_{i+1}] = \frac{f[x_{k+1} \ldots x_{i+1}] - f[x_k \ldots x_i]}{x_{i+1} - x_k} \tag{9.14}$$

These quantities are called *divided differences*. Now we can express the coefficients a_i in terms of these divided differences.

$$a_0 = f_0 = f[x_0]$$

$$a_1 = \frac{f_1 - f_0}{x_1 - x_0} = f[x_0, x_1]$$

$$a_2 = \frac{\dfrac{f_2 - f_1}{x_2 - x_1} - \dfrac{f_1 - f_0}{x_1 - x_0}}{x_2 - x_0}$$

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

$$= f[x_0, x_1, x_2]$$

Thus,

$$a_n = f[x_0, x_1, \ldots x_n] \tag{9.15}$$

Note that a_1 represents the *first divided difference* and a_2 the *second divided difference* and so on.

Substituting for a_i coefficients in equation (9.10), we get

$$p_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \ldots + f[x_0, x_1, \ldots x_n](x - x_0)(x - x_1) \ldots (x - x_{n-1})$$

This can be written more compactly as

$$p_n(x) = \sum_{i=1}^{n} f[x_0, \ldots x_i] \prod_{j=0}^{i-1} (x - x_j) \tag{9.16}$$

Equation (9.16) is called *Newton's divided difference interpolation polynomial.*

Example 9.6

Given below is a table of data for log x. Estimate log 2.5 using second order Newton interpolation polynomial.

i	0	1	2	3
x_i	1	2	3	4
log x_i	0	0.3010	0.4771	0.6021

Second order polynomials require only three data points. We use the first three points

$$a_0 = f[x_0] = 0$$

$$a_1 = f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{0.3010}{2 - 1} = 0.3010$$

$$a_2 = f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

$$f[x_1, x_2] = \frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{0.4771 - 0.3010}{3 - 2} = 0.1761$$

Therefore,

$$a_2 = \frac{0.1761 - 0.3010}{3 - 1} = -0.06245$$

$$p_2(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$$

$$= 0 + 0.3010(x - 1) + (-0.06245)(x - 1)(x - 2)$$

$$p_2(2.5) = 0.3010 \times 1.5 - (0.06245)(1.5)(0.5)$$

$$= 0.4515 - 0.0468$$

$$= 0.4047$$

Note that, in Example 9.6, had we used a linear polynomial, we would have obtained the result as follows:

$$p_1(x) = a_0 + a_1(x - x_0)$$

$$p_1(2.5) = 0 + 0.3010\,(1.5) = 0.4515$$

This shows that $p_2\,(2.5)$ is obtained by simply adding a correction factor due to third data point. That is

$$p_2(x) = p_1(x) + a_2(x - x_0)\,(x - x_1)$$

$$= p_1(x) + \Delta_2$$

If we want to improve the results further, we can apply further correction by adding another data point. That is

$$p_3(x) = p_2(x) + \Delta_3$$

where

$$\Delta_3 = a_3(x - x_0)(x - x_1)(x - x_2)$$

This shows that the Newton interpolation formula provides a very convenient form for interpolation at an increasing number of interpolation points. Newton formula can be expressed recursively as follows:

$$p_{k+1}(x) = p_k(x) + f[x_0, \ldots x_{k+1}]\,\phi_k(x)\,(x - x_k) \qquad (9.17)$$

where $p_k(x) = f[x_0, \ldots x_i]\,\phi_i(x) = \sum_{i=0}^{k} a_i\,\phi_i(x)$

and $\phi_i(x) = (x - x_0)(x - x_1)\ldots(x - x_{i-1})$