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.

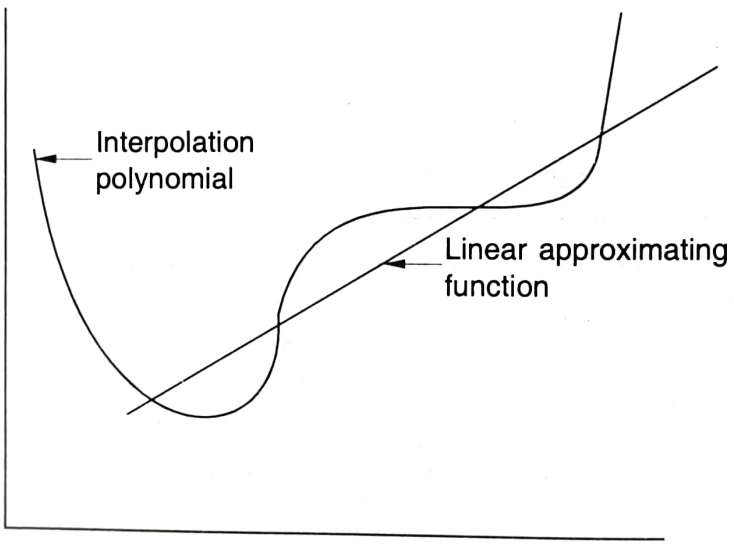

Fig. 9.1 Curve fitting to a set of points

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,

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

9.3 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

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

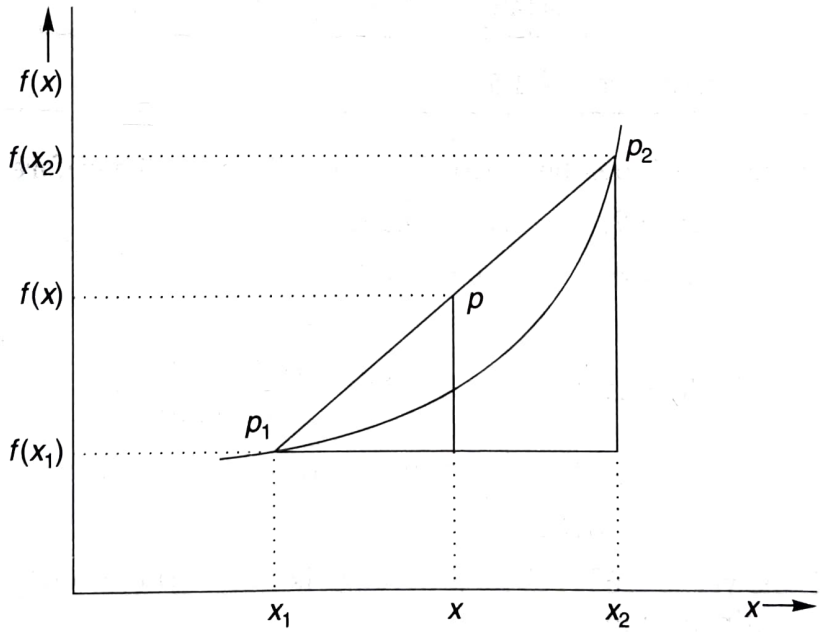

Fig. 9.2 Graphical representation of linear interpolation

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.

$$C_1 = x_1$$

$$a_0 = f(x_1)$$

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

Comparing eq^n (9.5) with eq^n (9.3)

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

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

where

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

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

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

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

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

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

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

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

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

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

$$p(1.5) = 0.9375$$

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.

```
            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 = 2.500000 is 1.5794000

Stop - Program terminated.
```

9.5 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,

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

$$\begin{aligned} p_2(x) &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) \\ &= 0 + 0.3010(x - 1) + (-0.06245)(x - 1)(x - 2) \\ p_2(2.5) &= 0.3010 \times 1.5 - (0.06245)(1.5)(0.5) \\ &= 0.4515 - 0.0468 \\ &= 0.4047 \end{aligned}$$

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

$$\begin{aligned} p_2(x) &= p_1(x) + a_2(x - x_0)\,(x - x_1) \\ &= p_1(x) + \Delta_2 \end{aligned}$$

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:

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

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

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

9.6 DIVIDED DIFFERENCE TABLE

We have seen that the coefficients of Newton divided difference interpolation polynomial are evaluated using the divided differences at the interpolating points. We have also seen that a higher-order divided difference is obtained using the lower-order differences. Finally, the first-order divided differences use the given interpolating points (i.e., x_k and f_k values). For example, consider the second-order divided difference

$$a_2 = f[x_0, x_1, x_2]$$

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

where $f[x_1, x_2]$ and $f[x_0, x_1]$ are first-order divided differences and are given by

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

$$f[x_1, x_2] = \frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{f_2 - f_1}{x_2 - x_1}$$

This shows that, given the interpolating points, we can obtain recursively a higher-order divided difference, starting from the first-order differences. While this can be conveniently implemented in a computer, we can generate a *divided difference table* for manual computing. A divided difference table for five data points is shown in Fig. 9.3. A particular entry in the table is obtained as follows:

$$f[x_1, x_2, x_3, x_4] = \frac{f[x_2, x_3, x_4] - f[x_1, x_2, x_3]}{x_4 - x_1}$$

i	x_i	$f[x_i]$	*First difference*	*Second difference*	*Third difference*	*Fourth difference*
0	x_0	$f[x_0]$				
			$f[x_0, x_1]$			
1	x_1	$f[x_1]$		$f[x_0, x_1, x_2]$		
			$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$	
2	x_2	$f[x_2]$		$f[x_1, x_2, x_3]$		$f[x_0, \ldots, x_4]$
			$f[x_2, x_3]$		$f[x_1, x_2, x_3, x_4]$	
3	x_3	$f[x_3]$		$f[x_2, x_3, x_4]$		
			$f[x_3, x_4]$			
4	x_4	$f[x_4]$				

Fig. 9.3 Divided difference table

Draw the two diagonals from the entry to be calculated through its neighbouring entries to the left. If these lines terminate at $f(x_i)$ and $f(x_j)$,

then divide the difference of the neighbouring entries by the corresponding difference $x_j - x_i$. The result is the desired entry. This is illustrated in Fig. 9.3. for the entry $f[x_1, x_2, x_3, x_4]$.

When the table is completed, the entries at the top of each column represent the divided difference coefficients.

Example 9.7

Given the following set of data points, obtain the table of divided differences. Use the table to estimate the value of $f(1.5)$.

i	0	1	2	3	4
x_i	1	2	3	4	5
$f(x_i)$	0	7	26	63	124

The divided difference table is given below:

i	x_i	$f(x_i)$	*First difference*	*Second difference*	*Third difference*	*Fourth difference*
0	1	0				
			7			
1	2	7		12		
			19		6	
2	3	26		18		0
			37		6	
3	4	63		24		
			61			
4	5	124				

The value of polynomial at $x = 1.5$ is computed as follows:

$$p_0(1.5) = 0$$
$$p_1(1.5) = 0 + 7(1.5 - 1) = 3.5$$
$$p_2(1.5) = 3.5 + 12(1.5 - 1)(1.5 - 2) = 0.5$$
$$p_3(1.5) = 0.5 + 6\,(1.5 - 1)(1.5 - 2)(1.5 - 3) = 2.25$$
$$p_4(1.5) = 2.25 + 0 = 2.25$$

The function value at $x = 1.5$ is 2.25

Note that $p_3(1.5) = p_4(1.5)$. This implies that correct results can be obtained using the third-order interpolation polynomial. It also illustrates that we can compute $f(1.5)$ in stages (recursively) using interpolation polynomials in increasing order. Computation is terminated when two consecutive estimates are approximately equal or their difference is within a specified limit.

It is clear that the computational effort required in adding one more data point to the estimation process is very much reduced due to the recursive nature of computation.

Let us have a close look at the divided difference table of Example 9.7. Notice the constant values under the column "third difference" and zero value under the column "fourth difference". Recall that the first divided difference is given by

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

This is nothing but the finite divided difference approximation of the first derivative of the function. Similarly, $f[x_0, x_1, x_2]$ is the second derivative and so on. Since the third derivative is constant, the function $f(x)$ should be a third-degree polynomial. In fact, the function used in Example 9.7 is

$$f(x) = x^3 - 1$$

and therefore

$$\frac{d'''f}{dx} = 6$$

and the fourth derivative is zero.

Program NEWINT

Program NEWINT constructs the Newton interpolation polynomial for a given set of data points and then computes the interpolation value at a specified value.

```
*-------------------------------------------------------*
      PROGRAM NEWINT
*-------------------------------------------------------*
* Main program                                          *
*   This program constructs the Newton interpolation    *
*   polynomial for a given set of data points and then  *
*   computes interpolation value at a specified value   *
*-------------------------------------------------------*
* Functions invoked                                     *
*    NIL                                                *
*-------------------------------------------------------*
* Subroutines used                                      *
*    NIL                                                *
*-------------------------------------------------------*
* Variables used                                        *
*   N  - Number of data points                          *
*   X  - Array of independent data points               *
*   F  - Array of function values                       *
*   XP - Desired point for interpolation                *
```

```
60    CONTINUE
      SUM = SUM + A(I) * PI
70  CONTINUE

    FP = SUM

*  Write results

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

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

Test Run Results Let us use the same table values that were used for testing the program LAGRAN. Test run results are given below:

```
Input number of data points
3
Input the values of X and F(x), one set on each line
2  1.4142
3  1.7321
4  2.0
Input XP where interpolation is required
2.5

NEWTON INTERPOLATION

Interpolated Function Value
at X = 2.5000000 is 1.5794000

Stop - Program terminated.
```

9.7 INTERPOLATION WITH EQUIDISTANT POINTS

In this section, we consider a particular case where the function values are given at a sequence of equally spaced points. Most of the engineering and scientific tables are available in this form. We often use such tables to estimate the value at a non-tabular point. Let us assume that

$$x_k = x_0 + kh$$

where x_0 is the reference point and h is the step size. The integer k may take either positive or negative values depending on the position of the reference point in the table. We also assume that we are going to use *simple differences* rather than *divided differences*. For this purpose, we define the following:

The *first forward difference* Δf_i is defined as

$$\Delta f_i = f_{i+1} - f_i$$

The *second forward difference* is defined as

$$\Delta^2 f_i = \Delta f_{i+1} - \Delta f_i$$

In general,

$$\boxed{\Delta^j f_i = \Delta^{j-i} f_{i+1} - \Delta^{j-1} f_i} \qquad (9.18)$$

We can now express the *simple forward differences* in terms of the *divided differences*. We know that

$$f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{f_1 - f_0}{h}$$

Therefore,

$$f_1 - f_0 = h\, f[x_0, x_1]$$

Then

$$\Delta f_0 = f_1 - f_0 = h\, f[x_0, x_1]$$

Similarly,

$$\Delta f_1 = h\, f[x_1, x_2]$$

Now,

$$\begin{aligned}
\Delta^2 f_0 &= \Delta f_1 - \Delta f_0 \\
&= h\, f[x_1, x_2] - h\, f[x_0, x_1] \\
&= h\, \{f[x_1, x_2] - f[x_0, x_1]\} \\
&= h\, .\, 2h\, .\, f[x_0, x_1, x_2] \\
&= 2\, h^2\, f[x_0, x_1, x_2]
\end{aligned}$$

In general, by induction,

$$\Delta^j f_i = j!\, h^j\, f[x_i, x_{i+1}, \ldots x_{i+j}]$$

Therefore,

$$f[x_0, x_1, \ldots x_j] = \frac{\Delta^j f_0}{j!\, h^j}$$

Substituting this in the Newton's divided difference interpolation polynomial (Eq. (9.16)) we get,

$$\boxed{p_n(x) = \sum_{j=0}^{n} \frac{\Delta^j f_0}{j!\, h^j} \prod_{k=0}^{j-1} (x - x_k)} \qquad (9.19)$$

Let us set

$$x = x_0 + sh \qquad \text{and} \qquad p_n(s) = p_n(x)$$

We know that

$$x_k = x_0 + kh$$

Thus we get

$$x - x_k = (s - k)h$$

Substituting this in Eq. (9.19), we get

$$p_n(s) = \sum_{j=0}^{n} \frac{\Delta^j f_0}{j!\, h^j} \prod_{k=0}^{j-1} (s-k)h$$

$$= \sum_{j=0}^{n} \frac{\Delta^j f_0}{j!\, h^j} [s(s-1)\ldots(s-j+1)]\, h^j$$

Thus,

$$\boxed{p_n(s) = \sum_{j=0}^{n} \binom{s}{j} \Delta^j f_0} \tag{9.20}$$

where

$$\binom{s}{j} = \frac{s(s-1)\ldots(s-j+1)}{j!}$$

is the binomial coefficient. Equations (9.19) and (9.20) are known as *Gregory-Newton forward difference formula*.

Forward Difference Table

The coefficients $\Delta^j f_i$ can be conveniently obtained from the *forward difference table* shown in Fig. 9.4. According to Eq. (9.18), each entry is merely the difference between the two diagonal entries immediately on its left. That is

$$\Delta^j f_i = \Delta^{j-1} f_{i+1} - \Delta^{j-1} f_i$$

The differences which appear on the top of each column correspond to the differences of equation (9.20).

x	f	Δf	$\Delta^2 f$	$\Delta^3 f$	$\Delta^4 f$	$\Delta^5 f$	$\Delta^6 f$
x_0	f_0						
		Δf_0					
x_1	f_1		$\Delta^2 f_0$				
		Δf_1		$\Delta^3 f_0$			
x_2	f_2		$\Delta^2 f_1$		$\Delta^4 f_0$		
		Δf_2		$\Delta^3 f_1$		$\Delta^5 f_0$	
x_3	f_3		$\Delta^2 f_2$		$\Delta^4 f_1$		
		Δf_3		$\Delta^3 f_2$			
x_4	f_4		$\Delta^2 f_3$				
		Δf_4					
x_5	f_5						

Fig. 9.4 Forward difference table

As pointed out earlier, difference tables can be used not only to estimate the value of the function at a non-tabular point but can also be used to decide on the degree of the interpolating polynomial that is most appropriate to the given data points.

Example 9.8

Estimate the value of sin θ at $\theta = 25°$ using the Newton-Gregory forward difference formula with the help of the following table.

θ	10	20	30	40	50
sin θ	0.1736	0.3420	0.5000	0.6428	0.7660

In order to use the Newton-Gregory forward difference formula, we need the values of $\Delta^i f_0$. These coefficients can be obtained from the difference table given below. The required coefficients are boldfaced.

θ	sin θ	Δf	$\Delta^2 f$	$\Delta^3 f$	$\Delta^4 f$	$\Delta^5 f$
10	**0.1736**					
		0.1684				
20	0.3420		**−0.0104**			
		0.1580		**0.0048**		
30	0.5000		− 0.0152		**−0.0004**	
		0.1428		0.0044		
40	0.6428		− 0.0196			
		0.1232				
50	0.7660					

$$x_0 = \theta_0 = 10$$

$$h = 10$$

Therefore,

$$s = \frac{x - x_0}{h} = \frac{25 - 10}{10} = 1.5$$

Using Eq. (9.20), we have

$$p_1(s) = 0.1736 + (1.5)\,(0.1684) = 0.4262$$

$$p_2(s) = 0.4262 + \frac{(1.5)\,(0.5)\,(-0.0104)}{2} = 0.4223$$

$$p_3(s) = 0.4223 + \frac{(1.5)\,(0.5)\,(-0.5)\,(0.0048)}{6} = 0.4220$$

$$p_4(s) = 0.4220 + \frac{(1.5)\,(0.5)\,(-0.5)\,(-1.5)\,(-0.0004)}{24} = 0.4220$$

Thus,

$$\sin 25 = 0.4220$$

which is accurate to four decimal places.

Backward Difference Table

If the table is too long and if the required point is close to the end of the table, we can use another formula known as *Newton-Gregory backward difference formula*. Here, the reference point is x_n, instead of x_0. Therefore, we have

$$x = x_n + sh$$

$$x_k = x_n - kh$$

$$x - x_k = (s + k)h$$

Then, the Newton-Gregory backward difference formula is given by

$$p_n(s) = f_n + s\,\nabla f_n + \frac{s(s+1)}{2!}\nabla^2 f_n + \ldots$$

$$+ \frac{s(s+1)\ldots(s+n-1)}{n!}\nabla^n f_n \qquad (9.21)$$

For a given table of data, the backward difference table will be identical to the forward difference table. However, the reference point will be below the point for which the estimate is required. This implies that the value of s will be negative for backward interpolation. The coefficients $\nabla j\, f_i$ can be obtained from the backward difference table shown in Fig. 9.5.

x	f	∇f	$\nabla^2 f$	$\nabla^3 f$	$\nabla^4 f$	$\nabla^5 f$	$\nabla^6 f$
x_0	f_0						
		∇f_1					
x_1	f_1		$\nabla^2 f_2$				
		∇f_2		$\nabla^3 f_3$			
x_2	f_2		$\nabla^2 f_3$		$\nabla^4 f_4$		
		∇f_3		$\nabla^3 f_4$		$\nabla^5 f_5$	
x_3	f_3		$\nabla^2 f_4$		$\nabla^4 f_5$		
		∇f_4		$\nabla^3 f_5$			
x_4	f_4		$\nabla^2 f_5$				
		∇f_5					
x_5	f_5						

Fig. 9.5 Backward difference table

Example 9.9

Repeat the estimation of sin 25 in Example 9.8 using Newton's backward difference formula

$$s = \frac{(x - x_n)}{h} = \frac{25 - 50}{10} = -2.5$$

Using Eq. (9.21), we get

$$\begin{aligned} p_4(2.5) &= 0.7660 + (-2.5)(0.1232) \\ &\quad + \frac{(-2.5)(-1.5)(-0.0196)}{2} \\ &\quad + \frac{(-2.5)(-1.5)(-0.5)(0.0044)}{6} \\ &\quad + \frac{(-2.5)(-1.5)(-0.5)(0.5)(-0.0004)}{24} \\ &= 0.4200 \end{aligned}$$

9.8 SPLINE INTERPOLATION

So far we have discussed how an interpolation polynomial of degree n can be constructed and used given a set of values of functions. There are situations in which this approach is likely to face problems and produce incorrect estimates. This is because the interpolation takes a global rather than a local view of data. It has been proved that when n is large compared to the order of the "true" function, the interpolation polynomial of degree n does not provide accurate results at the ends of the range. This is illustrated in Fig. 9.6. Note that the interpolation polynomial contains undesirable maxima and minima between the data points. This only shows that increasing the order of polynomials does not necessarily increase the accuracy.

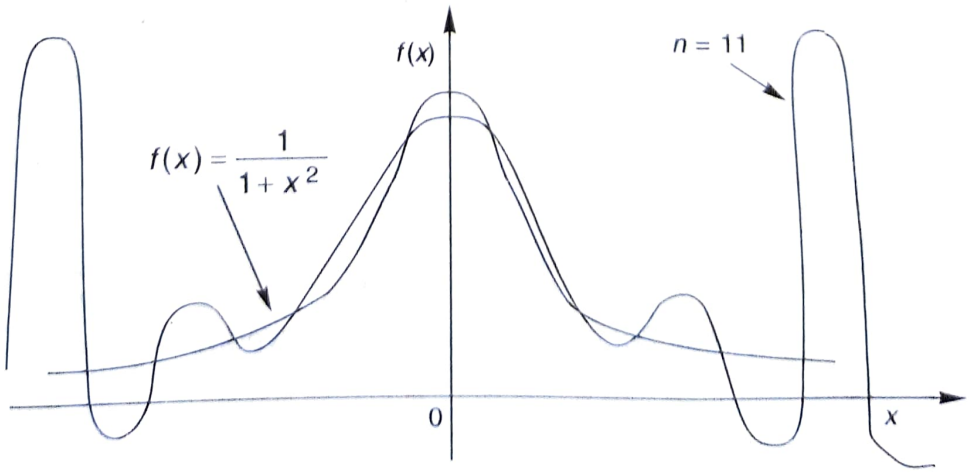

Fig. 9.6 Interpolation polynomial of degree 11 of the function $\frac{1}{1+x^2}$

CHAPTER 10

Curve Fitting: Regression

10.1 INTRODUCTION

In the previous chapter we discussed various methods of curve fitting for data points of well-defined functions. In this chapter, we will discuss methods of curve fitting for experimental data.

In many applications, it often becomes necessary to establish a mathematical relationship between experimental values. This relationship may be used for either testing existing mathematical models or establishing new ones. The mathematical equation can also be used to predict or forecast values of the dependent variable. For example, we would like to know the maintenance cost of an equipment (or a vehicle) as a function of age (or mileage) or the relationship between the literacy level and population growth. The process of establishing such relationships in the form of a mathematical equation is known as *regression analysis* or *curve fitting*.

Suppose the values of y for the different values of x are given. If we want to know the effect of x on y, then we may write a functional relationship

$$y = f(x)$$

The variable y is called the *dependent* variable and x the *independent variable*. The relationship may be either linear or nonlinear as shown in Fig. 10.1. The type of relationship to be used should be decided by the experiment based on the nature of scatteredness of data.

It is a standard practice to prepare a *scatter diagram* as shown in Fig. 10.2 and try to determine the functional relationship needed to fit the points. The line should best fit the plotted points. This means that the

average error introduced by the assumed line should be minimum. The parameters a and b of the various equations shown in Fig. 10.1 should be evaluated such that the equations best represent the data.

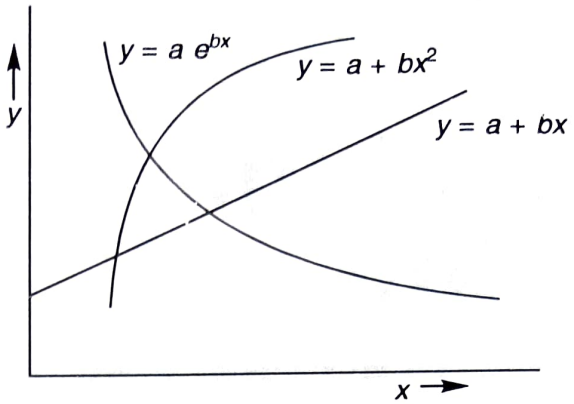

Fig. 10.1 Various relationships between x and y

We shall discuss in this chapter a technique known as *least-squares regression* to fit the data under the following situations:

1. Relationship is linear
2. Relationship is transcendental
3. Relationship is polynomial
4. Relationship involves two or more independent variables

10.2 FITTING LINEAR EQUATIONS

Fitting a straight line is the simplest approach of regression analysis. Let us consider the mathematical equation for a straight line

$$y = a + bx = f(x)$$

to describe the data. We know that a is the intercept of the line and b its slope. Consider a point (x_i, y_i) as shown in Fig. 10.2. The vertical distance of this point from the line $f(x) = a + bx$ is the error q_i. Then,

$$\begin{aligned} q_i &= y_i - f(x_i) \\ &= y_i - a - bx_i \end{aligned} \tag{10.1}$$

There are various approaches that could be tried for fitting a "best" line through the data. They include:

1. Minimise the sum of errors, i.e., minimise

$$\sum q_i = \sum (y_i - a - bx_i) \tag{10.2}$$

2. Minimise the sum of absolute values of errors

$$\sum |q_i| = \sum |(y_i - a - bx_i)| \tag{10.3}$$

3. Minimise the sum of squares of errors

$$\sum q_i^2 = \sum (y_i - a - bx_i)^2 \tag{10.4}$$