Summary of last session

-Chapter 4. Solving Systems of Linear Equations

- **4.4 Norms and the Analysis of Errors**

  Vector norms, Matrix Norms, Condition Number

- **4.6 Solution of Equations by Iterative Methods**

  Jacobi method, Gauss-Seidel method,

  General iteration methods, Richardson method.

Chapter 6

Approximating Functions

6.0 Introduction

For many engineering problems, we need to analyze or interpret somedescrete data obtrained from experiments or some observations. We need to find the regularities, or derivatives, or intergations, or roots and so on. Therefore we need to construct some approximate functions from the data. These approximate functions should be simple in form and good at representing the data.

In this chapter, several methods of how to construct such fuctions are discussed. Several different sub-problems will be considered. They differ according to the type of functions being represented, whether it is known at relatively few points or at many (or all) points.

We will discuss three methods mainly. They are **Polynomial Inter-polation (Lagrange and Newton forms (Divided Differences)) and Least-squares Approximation.**

6.1 Polynomial Interpolation

-method of undetermined coefficients

Assuming that function $y = f(x)$ is defined in the interval $[a, b]$ and the values of $y_i = f(x_i)$ at $x_i (i = 0, 1, 2, \cdots, n)$ $(x_0, x_1, \cdots, x_n \in [a, b])$ are given. If there exists a simple function p that makes

$$p(x_i) = y_i \qquad (0 \le i \le n) \qquad (1)$$

then p is called the **interpolation function** of f, and $x_0, x_1, \cdots, x_n$ are the **interpolation nodes**, $[a, b]$ is the **interpolation interval**, (1) is the interpolation conditions and the methods of finding function p is the interpolation methods. When the function p

$$p = p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \qquad (2)$$

is a polynomial of degree at most n where $a_0, \cdots, a_n$ are coefficients of the polynomial, and the method is called **polymonial interpolation**.

6.1 Polynomial Interpolation

-method of undetermined coefficients

The coefficient in (2) satisfy the following linear equation system:

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n = y_1 \\ \qquad\qquad \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n = y_n \end{cases} \qquad (3)$$

To prove the existance and uniqueness of function p, we need to show that solution of (3) exists and unique. The necessary condition for this is

$$V_n(x_0, x_1, \cdots, x_n) = \begin{vmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{vmatrix} = \prod_{i=1}^{n} \prod_{j=0}^{i-1} (x_i - x_j)$$

which is called Vandermonde determinant, is not zero. As the nodes are distinct, $V_n \neq 0$. Therefore the solution exists and is unique.

6.1 Polynomial Interpolation

-method of undetermined coefficients

The error of this interpolation can be analysis as follows:

Assuming that the derivations $f', f'', \ldots, f^{(n+1)}$ exists in the interval, the error $R_n(x)$ (also called remaining item) of the interpolation can be calculated by

$$R_n(x) = f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x) \qquad (4)$$

where $\xi \in [a,b]$ and depends on x, and $\omega_{n+1}(x) = \prod_{i=0}^{n}(x - x_i)$.

Generally, the value of ξ is unknown. However if we can determine upper bound of $f^{(n+1)}$, i.e. $\max_{a<x<b} | f^{(n+1)} | = M_{n+1}$, then the error term is limited by

$$|R_n(x)| \le \frac{M_{n+1}}{(n+1)!} | \omega_{n+1}(x) |$$

6.1 Polynomial Interpolation

-Lagrange interpolation method

The calculation for the coefficients $[a_0,\cdots,a_n]$ by solving the equation system (3) can be very time consuming. We introduce following methods. We have many ways to give the polynomial p for given data. Firstly, let's consider a two-point case, x_0, x_1 $(n=1)$. We have conditions

$$p_1(x_0)=y_o=a_o+a_1x_0$$

$$p_1(x_1)=y_1=a_1+a_1x_1$$

Solving the equation system, we get

$$a_o=\frac{y_1x_o-y_ox_1}{x_o-x_1},\quad a_1=\frac{y_o-y_1}{x_o-x_1}$$

So the polynomial is

$$p_1(x)=\frac{y_1x_o-y_ox_1}{x_o-x_1}+\frac{y_o-y_1}{x_o-x_1}x$$

This is actually using a straight line which pass through the two points (x_0, y_0) and (x_1, y_1) to approximate the function $y=f(x)$.

6.1 Polynomial Interpolation

-Lagrange interpolation method

If we write the interpolation function p_1 in the sum of two linear functions, we have

$$p_1(x) = y_o\left(\frac{x - x_1}{x_o - x_1}\right) + y_1\left(\frac{x - x_o}{x_1 - x_o}\right)$$

let

$$l_0(x) = \frac{x - x_1}{x_o - x_1}, \text{ and } \quad l_1(x) = \frac{x - x_o}{x_1 - x_o}$$

Obviously,

$$l_0(x_o) = 1,\ l_0(x_1) = 0,\ l_1(x_o) = 0,\ l_1(x_1) = 1.$$

This indicates that at the ith-interpolation point $l_i(x)(i = 0,1) = 1$ and $l_i(x) = 0$ elsewhere. Similarly, we consider $n+1$ nodes, and construct a function $l_i(x)(i = 0,1,\cdots,n)$ of which the order is not greater than n to satisfy

$$l_i(x_j) = \begin{cases} 0 & j \neq i \\ 1 & j = i \end{cases} \qquad (5)$$

6.1 Polynomial Interpolation

-Lagrange interpolation method

and using the values of y at the interpolation nodes as the cocefficients for corresponding $l_i(x)$ to generate the polynomial (linear sum of these terms);

$$p_{\mathrm{n}}(x) = \sum_{i=0}^{n} y_i l_i(x)$$

According to (5), $l_i(x)$ should have n of roots, and $l_i(x_i) = 0$ at every node. Therefore it must have the form

$$l_i(x) = \frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j} \quad (i=0,1,\cdots,n)$$

so

$$p_n(x) = \sum_{i=0}^{n} y_i l_i(x) = \sum_{i=0}^{n} y_i \left(\prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}\right) \qquad (6)$$

(6) is the Lagrange polynomial formula. $l_i(x)$ is called the base function of the interpolation. We often use $L_n(\mathrm{x})$ instead of $p_n(x)$.

6.1 Polynomial Interpolation

-Lagrange interpolation method

Example 1 Consider a function $y = 2^x$ and the following five nodes

x	-2	-1	0	1	2
$y=2^x$	0.25	0.5	1	2	4

(i) take $x_0 = 0, x_1 = 1$ as interpolation nodes and generate the polynomial $L_1(x)$; use $L_1(x)$ to calculate $2^{0.3}$ and estimate the error.

(ii) take $x_0 = -1,\ x_1 = 0,\ x_2 = 1$ as nodes to obtain $L_2(x)$; use $L_2(x)$ to calculate $2^{0.3}$ and estimate the error.

6.1 Polynomial Interpolation

-Lagrange interpolation method

Solution for (i): by (6), the polynomial $L_1(x)$ is

$$L_1(x) = \frac{x-1}{0-1} \bullet 1 + \frac{x-0}{1-0} \bullet 2 = x+1$$

Therefore $$2^{0.3} \approx L_1(0.3) = 1.3$$

and by (4), the error is

$$|R_1(x)| \le \frac{M_2}{2} |(x-x_0)(x-x_1)|$$

where $M_2 = \max\limits_{x_2 \le x \le x_3} |f''(x)|$. Because $f''(x) = (\ln 2)^2 2^x$, we have

$$\max_{0 \le x \le 1} |f''(x)| = 2(\ln 2)^2 = 0.9069$$

so that $$|R_1(0.3)| \le \frac{0.9069}{2} |0.3(0.3-1)| = 0.09522$$

6.1 Polynomial Interpolation

-Lagrange interpolation method

Solution for (ii): similarly by (6), the polynomial $L_2(x)$ is

$$L_2(x)=\frac{(x-0)(x-1)}{(-1-0)(-1-1)}\bullet 0.5+\frac{(x+1)(x-1)}{(0+1)(0-1)}\bullet 1+\frac{(x-0)(x+1)}{(1-0)(1+1)}\bullet 2$$

$$=0.25x^2+0.75x+1$$

Therefore $\quad 2^{0.3}\approx L_2(0.3)=1.248$

and by (4), the error is

$$|R_2(x)|\le\frac{M_3}{6}|(x-x_0)(x-x_1)(x-x_2)|$$

where $M_3=\max\limits_{-1\le x\le 1}|f'''(x)|=2(\ln 2)^3=0.6660$, we have

so that

$$|R_2(0.3)|\le\frac{0.6660}{6}|(0.3+1)(0.3-1)(0.3-1)|=0.03030$$

6.2 Polynomial Interpolation

-- Newton interpolation – divided difference

The Lagrange interpolation formula is neat and symmetric in form, and it is easy to get from the base function. It is easy for programming and also convenient for theoretical analysis. But in some situation at which when we need to add some more node points, the formula appears not convenient as we have to change all the base functions $l_i(x)\,(i = 0,1,\cdots,n)$ accordingly. This is obviously not pratical. However, if we change the formula to

$$p_1(x) = y_0 + \frac{(y_1 - y_0)}{(x_1 - x_0)}(x - x_0) \qquad (7)$$

the problem may be solved. We then derive another form of interpolation. Before deriving the formula, we first introduce some concepts called **Divided Difference**, and Difference which will be used in the formula.

6.2 Polynomial Interpolation

- Newton interpolation – divided difference

Let $f(x)$ be a function whose values are known at a set of points $x_o, x_1, \cdots, x_n$. Assuming that these points are distinct. We call

$$f[x_i, x_j] = \frac{f(x_i) - f(x_j)}{x_i - x_j} \quad (i \neq j)$$

the first order divided difference at points x_i, x_j. The higher order divided difference is similar. We defined

$$f[x_i, x_j, x_k] = \frac{f[x_i, x_j] - f[x_j, x_k]}{x_i - x_k} \quad (i \neq j)$$

as second order divided difference at point x_i, x_j, x_k.

Generally, the expression of

$$f[x_0, x_1, \cdots, x_k] = \frac{f[x_0, x_1, \cdots, x_{k-1}] - f[x_0, x_1, \cdots, x_k]}{x_0 - x_k}$$

is the k-order divided difference.

6.2 Polynomial Interpolation

- Newton interpolation – divided difference

The didvided difference has the following natures

(i) The linearity if $f(x)=a\varphi(x)+b\psi(x)$, then for any k, we have

$$f[x_0, x_1, \cdots, x_k]=a\varphi[x_0, x_1, \cdots, x_k]+b\psi[x_0, x_1, \cdots, x_k]$$

(ii) k-order divided difference can be expressed by linear sum of $f(x_0), f(x_1), \cdots, f(x_k)$, i.e.

$$f[x_0, x_1, \cdots, x_k]=\sum_{i=0}^{k}\frac{f(x_i)}{\omega_{k+1}'(x_i)}$$

where $\omega_{k+1}'(x_i)=\prod_{\substack{j=0 \\ j\neq i}}^{k}(x_i-x_j)$.