CHƯƠNG 7 : MỘT SỐ VẤN ĐỀ VỀ ĐA THỨC VÀ HÀM SỐ

§1. MỘT SỐ KHÁI NIỆM CHUNG

1. Khái niệm về phương pháp tính : Phương pháp tính là môn học về những lí luận cơ bản và các phương pháp giải gần đúng,cho ra kết quả bằng số của các bài toán thường gặp trong toán học cũng như trong kĩ thuật.

Chúng ta thấy rằng hầu hết các bài toán trong toán học như giải các phương trình đại số hay siêu việt,các hệ phương trình tuyến tính hay phi tuyến,các phương trình vi phân thường hay đạo hàm riêng,tính các tích phân,... thường khó giải đúng được,nghĩa là khó tìm kết quả dưới dạng các biểu thức.

Một số bài toán có thể giải đúng được nhưng biểu thức kết quả lại cồng kềnh,phức tạp khối lượng tính toán rất lớn.Vì những lí do trên,việc giải gần đúng các bài toán là vô cùng cần thiết.

Các bài toán trong kĩ thuật thường dựa trên số liệu thực nghiệm và các giả thiết gần đúng.Do vậy việc tìm ra kết quả gần đúng với sai số cho phép là hoàn toàn có ý nghĩa thực tế.

Từ lâu người ta đã nghiên cứu phương pháp tính và đạt nhiều kết quả đáng kể. Tuy nhiên để lời giải đạt được độ chính xác cao,khối lượng tính toán thường rất lớn.Với các phương tiện tính toán thô sơ,nhiều phương pháp tính đã được đề xuất không thể thực hiện được vì khối lượng tính toán quá lớn.Khó khăn trên đã làm phương pháp tính không phát triển được.

Ngày nay nhờ máy tính điện tử người ta đã giải rất nhanh các bài toán khổng lồ,phức tạp,đã kiểm nghiệm được các phương pháp tính cũ và đề ra các phương pháp tính mới. Phương pháp tính nhờ đó phát triển rất mạnh mẽ.Nó là cầu nối giữa toán học và thực tiễn.Nó là môn học không thể thiếu đối với các kĩ sư.

Ngoài nhiệmvụ chính của phương pháp tính là tìm các phương pháp giải gần đúng các bài toán,nó còn có nhiệm vụ khác như nghiên cứu tính chất nghiệm,nghiên cứu bài toán cực trị,xấp xỉ hàm v.v. Trong phần này chúng ta sẽ nghiên cứu một loạt bài toán thường gặp trong thực té và đưa ra chương trình giải chúng.

2. Các đặc điểm của phương pháp tính : Đặc điểm về phương pháp coả môn học này là hữu hạn hoá và rời rạc hoá.

Phương pháp tính thường biến cái vô hạn thành cái hữu hạn,cái liên tục thành cái rời rạc và sau cùng lại trở về với cái vô hạn,cái liên tục.Nhưng cần chú ý rằng quá trình trở lại cái vô hạn,cái liên tục phải trả giá đắt vì khối lượng tính toán tăng lên rất nhiều.Cho nên trong thực tế người ta dừng lại khi nghiệm gần đúg sát với nghiệm đúng ở một mức độ nào đó.

Đặc diểm thứ hai của môn học là sự tiến đến kết quả bằng quá trình liên tiếp.Đó là quá trình chia ngày càng nhỏ hơn,càng dày đặc hơn hoặc quá trình tính toán bước sau dựa vào các kết quả của các bước trước.Công việc tính toán lặp đi lặp lại này rất thích hợp với máy điện toán.

Khi nghiên cứu phương pháp tính người ta thường triệt để lợi dụng các kết quả đạt được trong toán học.Cùng một bài toán có thể có nhiều phương pháp tính khác nhau.Một phương pháp tính được coi là tốt nếu nó đạt các yêu cầu sau :

- phương pháp tính được biểu diễn bằng một dãy hữu hạn các bước tính cụ thể.Các bước tính toán cụ thể này của phương pháp tính được gọi là thuật toán. Thuật toán càng đơn giản càng tốt.
- đánh giá được sai số và sai số càng nhỏ càng tốt.
- thuật toán thực hiện được trên máy điện toán và thời gian chạy máy ít nhất

3. Các loại sai số : Trong việc thiếtlập và giải các bài toán thực tế ta thường gặp các loại sai số.

Giả sử ta xét bài toán A nào đó.Nghiên cứu các quy luật liên hệ giữa các đại lượng trong bài toán dẫn đến phương trình có dạng tổng quát :

$$y = Bx$$

Trong đó : x - đại lượng đã biết

y - đại lượng chưa biết

B - quy luật biến đổi từ x sang y

Bài toán thực tế thường rất phức tạp.Để đơn giản và có thể diễn đạt nó bằng toán học,người ta đưa ra một số giả thiết không hoàn toàn chính xác để nhận được phương trình trên.

Vì vậy nếu gọi y_1 là giá trị đúng của y thì khi đó $y \neq y_1$. Giá trị $|y - y_1|$ được gọi là *sai số giả thiết của bài toán*.

Do x là số liệu ban đầu của bài toán,thu được từ đo lường,thí nghiệm nên nó chỉ là giá trị gần đúng.Sai số này được gọi là *sai số của các số liệu ban đầu*.

Để giải gần đúng phương trình trên ta thường thay B bằng C hay x bằng t để phương trình đơn giản hơn và có thể giải được.Bằng cách đó ta tìm được y_2 gần đúng với y.Giá trị $|y_2 - y|$ được gọi là *sai số phương pháp của bài toán*.

Cuối cùng khi thực hiện các phép tính ta thường thu gọn các kết quả trung gian hay kết quả cuối cùng nên đáp số của bài toán là y_3.Giá trị $|y_3 - y|$ là *sai số tính toán*.

Trong phần này chúng ta quan tâm tới sai số phương pháp.

4. Xấp xỉ và hội tụ : Xét bài toán

$$y = Bx$$

Giả sử y là nghiệm đúng của bài toán mà ta chưa biết.Bằng phương pháp nào đó ta lấy y_1 thay cho y và khi đó y_1 gọi là xấp xỉ thứ nhất của nghiệm và viết :

$$y_1 \approx y$$

Cũng bằng phương pháp tương tự,ta xây dựng được một dãy các xấp xỉ $y_1,y_2,y_3,..y_n$.Nếu ta có :

$$\lim_{n\to\infty} y_n = y$$

thì ta nói dãy xấp xỉ hội tụ tới nghiệm y.

§2. TÍNH GIÁ TRỊ CỦA ĐA THỨC THEO SƠ ĐỒ HORNER

1. Sơ đồ Horner : Giả sử chúng ta cần tìm giá trị của một đa thức tổng quát dạng :

$$P(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} +....+ a_n \qquad (1)$$

tại một trị số x nào đó. Trong (1) các hệ số a_i là các số thực đã cho. Chúng ta viết lại (1) theo thuật toán Horner dưới dạng :

$$P(x_o) = (...((a_0x + a_1)x+ a_2x)+...+ a_{n-1})x + a_n \qquad (2)$$

Từ (2) ta nhận thấy :

$$P_0 = a_0$$
$$P_1 = P_0x + a_1$$
$$P_2 = P_1x + a_2$$
$$P_3 = P_2x + a_3$$
.....
$$P(x) = P_n = P_{n-1}x + a_n$$

Tổng quát ta có :

$$P_k = P_{k-1}x + a_k \text{ với } k =1,2...n ; P_0 = a_0$$

Do chúng ta chỉ quan tâm đến trị số của P_n nên trong các công thức truy hồi về sau chúng ta sẽ bỏ qua chỉ số k của P và viết gọn P := Px + ak với k = 0...n.Khi ta tính tới k = n thì P chính là giá trị cần tìm của đa thức khi đã cho x. Chúng ta thử các bước tính như sau :

Ban đầu			$P = 0$
Bước 0	$k = 0$		$P = a_o$
Bước 1		$k = 1$	$P = a_o x + a_1$
Bước 2		$k = 2$	$P = (a_o x + a_1)x + a_2$
.....			
Bước n-1	$k = n - 1$		$P = P(x_o) = (...((a_o x + a_1)x+a_2 x)+...+a_{n-1})x$
Bước n		$k = n$	$P = P(x_o) = (...((a_o x + a_1)x+a_2 x)+...+a_{n-1})x + a_n$

Sau đây là chương trình thực hiên thuật toán trên

Chương trình 7-1

```
#include <conio.h>
#include <stdio.h>
#define m  10

void main(void)
 {
   int  k,n;
   float p,x;
   float a[m];

   clrscr();
   printf("\nCho bac cua da thuc n = ");
   scanf("\%d",&n);
   printf("Vao cac he so a:\n");
   for (k=1;k<=n+1;k++)
           {
        printf("a[%d] = ",k-1);
        scanf("%f",&a[k]);
     };
   printf("Cho gia tri x = ");
   scanf("%f",&x);
   p=0.0;
   for (k=1;k<=n+1;k++)
     p=p*x+a[k];
   printf("Tri so cua da thuc P tai x =%.2f la :%.5f",x,p);
   getch();
 }
```

2. Sơ đồ Horner tổng quát : Giả sử chúng ta có đa thức :

$$P_n(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} +....+ a_n \qquad (1)$$

Khai triển Taylor của đa thức tại $x = x_o$ có dạng :

$$P_n(x) = P_n(x_0) + \frac{P'(x_0)}{1!}(x - x_0) + \frac{P''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{P^{(n)}(x_0)}{2!}(x - x_0)^n \qquad (2)$$

Mặt khác chúng ta có thể biến đổi đa thức về dạng :

$$P_n(x) = (x - x_o)P_{n-1}(x) + P_n(x_o) \qquad (3)$$

Trong đó $P_{n-1}(x)$ là đa thức bậc n-1 và có dạng :

$$P_{n-1}(x) = b_o x^{n-1} + b_{o-1}x^{n-2} + b_2 x^{n-3} + \ldots + b_{n-1} \quad (4)$$

Thuật toán để tìm các hệ số nhận được bằng cách so sánh (1) và (3) :

$b_o = a_o$

$b_i = a_i + b_{i-1}x_o$

$b_n = P_n(x_o)$

So sánh (2) và (3) ta có :

$$(x - x_0)P_{n-1}(x_0) + P_n(x_0) = P_n(x_0) + \frac{P'(x_0)}{1!}(x - x_0) + \frac{P''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{P^{(n)}(x_0)}{2!}(x - x_0)^n$$

hay :

$$(x - x_0)P_{n-1}(x) = \frac{P'(x_0)}{1!}(x - x_0) + \frac{P''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{P^{(n)}(x_0)}{2!}(x - x_0)^n$$

và khi chia hai vế cho $(x - x_0)$ ta nhận được :

$$P_{n-1}(x) = \frac{P'(x_0)}{1!} + \frac{P''(x_0)}{2!}(x - x_0) + \cdots + \frac{P^{(n)}(x_0)}{2!}(x - x_0)^{n-1} \quad (5)$$

So sánh (4) và (5) ta nhận được kết quả :

$$P_{n-1}(x_0) = \frac{P'(x_0)}{1!}$$

Trong đó $P_{n-1}(x)$ lại có thể phân tích giống như $P_n(x)$ dạng (3) để tìm ra $P_{n-1}(x_o)$.Quá trình này được tiếp tục cho đến khi ta tìm hết các hệ số của chuỗi Taylor của $P_n(x)$

Tổng quát thuật toán thể hiện ở bảng sau :

$P_n(x)$	a_o	a_1	a_2	a_3	...	a_{n-1}	a_n
$x = x_o$	0	$b_o x_o$	$b_1 x_o$	$b_2 x_o$		$b_{n-2} x_o$	$b_{n-1} x_o$
$P_{n-1}(x)$	b_o	b_1	b_2	b_3	...	b_{n-1}	$b_n = P_n(x_o)$

Để hiểu rõ hơn chúng ta lấy một ví dụ cụ thể sau : Khai triển đa thức sau tại $x_0 = 2$

$P(x) = x^5 - 2x^4 + x^3 - 5x + 4$

Ta lập bảng tính sau :

	1	-2	1	0	-5	4
2	0	2	0	2	4	2
	1	0	1	2	-1	2 = P(2)/0!
2	0	2	4	10	24	
	1	2	5	12	23 = P'(2)/1!	
2	0	2	8	26		
	1	4	13	38 = P''(2)/2!		
2	0	2	12			
	1	6	25 = P'''(2)/3!			
2	0	2				
	1	8 = P''''(2)/4!				

2 0

1 = P''''(2)/4!

Như vậy :

$P_n(x) = (x-2)^5 + 8(x-2)^4 + 25(x-2)^3 + 38(x-2)^2 + 23(x-2) + 2$

Chương trình sau dùng để xác định các hệ số của chuỗi Taylor của đa thức P(x) tại $x_0 = 2$.

Chương trình 7-2

```
#include <conio.h>
#include <stdio.h>
#define  m  10

void main(void)
  {
    float a[m],b[m],c[m];
    int n,i,j,k;
    float x;

    clrscr();
    printf("Cho bac cua da thuc n = ");
    scanf("%d",&n);
    printf("Cho gia tri x = ");
    scanf("%f",&x);
    printf("Vao cac he so a\n");
    for (k=n;k>=0;k--)
      {
          printf("a[%d] = ",n-k);
          scanf("%f",&a[k]);
      }
    printf("\n");
    b[n] = a[n];
    c[n] = a[n];
    for (k=0;k<=n-1;k++)
      {
           for (i=n-1;i>=k;i--)
             b[i] = b[i+1]*x + a[i];
           c[k] = b[k];
           for (j=n;j>=k+1;j--)
             a[j] = b[j];
      }
    printf("\nSo do Horner tong quat");
    printf("\nKhai trien tai x = %.4f\n",x);
    for (k=n;k>=0;k--)
          printf("%10.4f\t",c[k]);
    getch();
  }
```

§3. CÁC PHÉP TÍNH TRÊN ĐA THỨC

1. Phép cộng hai đa thức : Giả sử chúng ta có hai đa thức A(x) bậc n và B(x) bậc m với n>m. Khi cộng hai đa thức này,chúng ta cộng lần lượt các hệ số cùng bậc của chúng với nhau.Ta có chương trình sau :

Chương trình 7-3

```
#include <conio.h>
#include <stdio.h>
#define t 10

void main(void)
  {
    int k,n,m;
    float a[t],b[t],c[t];

    clrscr();
    printf("Cho bac cua da thuc A n = ");
    scanf("%d",&n);
    printf("Vao cac he so a\n");
    for (k=1;k<=n+1;k++)
      {
                  printf("a[%d] = ",k-1);
                  scanf("%f",&a[k]);
      }
    printf("Cho bac cua da thuc B m = ");
    scanf("%d",&m);
         printf("Vao cac he so b\n");
    for (k=1;k<=m+1;k++)
      {
                 printf("b[%d] = ",k-1);
                 scanf("%f",&b[k]);
      }
    printf("\n");
    for (k=1;k<=n+1;k++)
    if (k<=n-m)
      c[k] = a[k];
    else
      c[k] = a[k] + b[k-n+m];
    printf("Cac he so cua da thuc tong C la :\n");
    for (k=1;k<=n+1;k++)
           printf("%.4f\t",c[k]);
         getch();
  }
```

2. Phép nhân hai đa thức : Để thấy rõ thuật toán xác định các hệ số của đa thức C(x) là kết quả của phép nhân hai đa thức A(x) và B(x) ta cho một ví dụ cụ thể :

$A(x) = a_o x^5 + a_1 x^4 + a_2 x^3 + a_3 x^2 + a_4 x + a_5$

$B(x) = b_o x^3 + b_1 x^2 + b_2 x + b_3$

$C(x) = A(x).B(x)$

$= a_o b_o x^8 + (a_o b_1 + a_1 b_o)x^7 + (a_o b_2 + a_1 b_1 + a_2 b_o)x^6 + (a_o b_3 + a_1 b_2 + a_2 b_1 + a_3 b_o)x^5 + (a_1 b_3 + a_2 b_2 + a_3 b_1 + a_4 b_o)x^4 + (a_2 b_3 + a_3 b_2 + a_4 b_1 + a_5 b_o)x^3 + (a_3 b_3 + a_4 b_2 + a_5 b_1)x^2 + a_5 b_2 x + a_5 b_3$

Các hệ số của đa thức kết quả là :

$C_o = a_o b_o$

$C_1 = a_o b_1 + a_1 b_o$

$C_2 = a_o b_2 + a_1 b_1 + a_2 b_o$

$C_3 = a_o b_3 + a_1 b_2 + a_2 b_1 + a_3 b_o$

$C_4 = a_1 b_3 + a_2 b_2 + a_3 b_1 + a_4 b_o$

$C_5 = a_2 b_3 + a_3 b_2 + a_4 b_1 + a_5 b_o$

$C_6 = a_3 b_3 + a_4 b_2 + a_5 b_1$

$C_7 = a_5 b_2$

$C_8 = a_5 b_3$

Ta nhận thấy là hệ số C_k của C(x) là tổng các tích các hệ số của đơn thức bậc i của A(x) và bậc (k-i) của B(x). Chỉ số i = 0 khi k <= m+1 và i = k+m khi k > m+1.Chỉ số j = k khi k <= n+1 và j = n +1 khi k > n + 1. Chương trình tính tích hai đa thức :

Chương trình 7-4

```
#include <conio.h>
#include <stdio.h>
#define t  10

void main()
  {
        int k,n,m,l,i,j,p;
        float a[t],b[t],c[2*t];

        clrscr();
        printf("Cho bac cua da thuc A n = ");
        scanf("%d",&n);
        printf("Vao cac he so a\n");
        for (k=1;k<=n+1;k++)
          {
                printf("a[%d] = ",k-1);
                scanf("%f",&a[k]);
          }
        printf("Cho bac cua da thuc B m = ");
        scanf("%d",&m);
        printf("Vao cac he so b\n");
        for (k=1;k<=m+1;k++)
          {
                printf("b[%d] = ",k-1);
                scanf("%f",&b[k]);
          }
        printf("\n");
        l=n+m;
```

```
        for (k=1;k<=l+1;k++)
          {
                if (k<=(n+1))
                  j=k;
                else
                  j=n+1;
                if (k<=(m+1))
                  p=1;
                else
                  p= k-m;
                c[k]=0;
                for (i=p;i<=j;i++)
                  c[k] = c[k] + a[i]*b[k-i+1];
          }
        printf("Cac he so cua da thuc tich C voi bac %d la :\n",l);
        for (k=1;k<=l+1;k++)
          printf("%.4f\t",c[k]);
        getch();
  }
```

3. Chia hai đa thức : Giả sử ta có hai đa thức là $A_n(x)$ và $B_m(x)$ với n ≥ m.Thương hai đa thức này là :

$$\frac{A_n(x)}{B_m(x)} = Q_{n-m}(x) + \frac{R_{m-1}(x)}{B_m(x)}$$

Chương trình sau thực hiện việc chia 2 đa thức :

Chương trình 7-5

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define t 10
void main()
  {
        int k,n,m,l,i,j,jp;
        float a[t],b[t],q[t],r[t],epsi;

        clrscr();
        printf("Cho bac cua da thuc A n = ");
        scanf("%d",&n);
        printf("Vao cac he so a\n");
        for (k=1;k<=n+1;k++)
          {
                printf("a[%d] = ",k-1);
                scanf("%f",&a[k]);
          }
        printf("\n");
        printf("Cho bac cua da thuc B m = ");
        scanf("%d",&m);
```

```
        printf("Vao cac he so b\n");
        for (k=1;k<=m+1;k++)
          {
                printf("b[%d] = ",k-1);
                scanf("%f",&b[k]);
          }
        printf("\n");
        printf("Cho gia tri sai so epsilon epsi = ");
        scanf("%f",&epsi);
        if ((m+1)>1)
          {
                l=n-m+1;
                for (i=0;i<=t;i++)
                  r[i]=a[i];
                j=n;
                for (k=1;k<=l;k++)
                  {
                        q[k]=r[1]/b[1];
                        for (i=1;i<=j;i++)
                          if ((i<m+1))
                                r[i]=r[i+1]-q[k]*b[i+1];
                          else
                                r[i]=r[i+1];
                          j=j-1;
                  }
                while ((abs(r[i])<epsi)&&(j>0))
                  {
                        for (i=1;i<=j;i++)
                          r[i]=r[i+1];
                        j=j-1;
                  }
                if (abs(r[1])<epsi)
                  r[1]=0.0;
                jp=j+1;
        }
else
        {
                l=n+1;
                for (k=1;k<=l;k++)
                  q[k]=a[k]/b[1];
                jp=1;
                r[1]=0.0;
          }
        printf("\n");
        printf("Cac he so cua thuong Q(x) bac %d la : ",l);
        for (k=1;k<=l;k++)
          printf("%.3f\t",q[k]);
        printf("\n");
        printf("Cac he so cua phan du R(x) bac %d la : ",jp-1);
        for (k=1;k<=jp;k++)
```

```
        printf("%.3f",r[k]);
      getch();
}
```