CHƯƠNG 8 : GIẢI GẦN ĐÚNG PHƯƠNG TRÌNH ĐẠI SỐ VÀ SIÊU VIỆT

§1.KHÁI NIỆM CHUNG

Nếu phương trình đại số hay siêu việt khá phức tạp thì ít khi tìm được nghiệm đúng.Bởi vậy việc tìm nghiệm gần đúng và ước lượng sai số là rất cần thiết.

Ta xét phương trình :

$$f(x) = 0 \qquad (1)$$

với f(x) là hàm cho trước của biến x.Chúng ta cần tìm giá trị gần đúng của nghiệm của phương trình này.

Quá trình giải thường chia làm hai bước: bước sơ bộ và bước kiện toàn nghiệm.

Bước giải sơ bộ có 3 nhiệm vụ:vây nghiệm, tách nghiệm và thu hẹp khoảng chứa nghiệm.

Vây nghiệm là tìm xem các nghiệm của phương trình có thể nằm trên những đoạn nào của trục x.Tách nghiệm là tìm các khoảng chứa nghiệm soa cho trong mỗi khoảng chỉ có đúng một nghiệm.Thu hẹp khoảng chứa nghiệm là làm cho khoảng chứa nghiệm càng nhỏ càng tốt.Sau bước sơ bộ ta có khoảng chứa nghiệm đủ nhỏ.

Bước kiện toàn nghiệm tìm các nghiệm gần đúng theo yêu cầu đặt ra.

Có rất nhiều phương pháp xác định nghiệm của (1).Sau đây chúng ta xét từng phương pháp.

§2.PHƯƠNG PHÁP LẶP ĐƠN

Giả sử phương trình (1) được đưa về dạng tương đương :

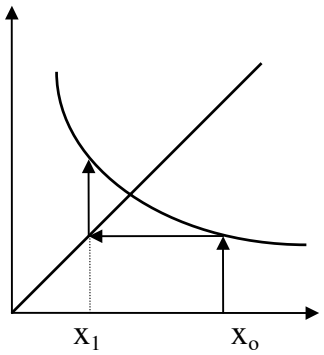
$$x = g(x) \qquad (2)$$

từ giá trị x_o nào đó gọi là giá trị lặp đầu tiên ta lập dãy xấp xỉ bằng công thức :

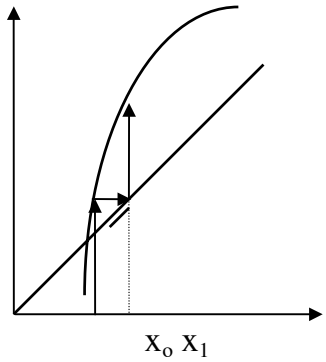
$$x_n = g(x_{n-1}) \qquad (3)$$

với n = 1,2,....

Hàm g(x) được gọi là hàm lặp.Nếu dãy $x_n \to \alpha$ khi $n \to \infty$ thì ta nói phép lặp (3) hội tụ.



Hình a



Hình b

Ta có định lí:Xét phương pháp lặp (3),giả sử :

- [a,b] là khoảng phân li nghiệm α của phương trình (1) tức là của (2)
- mọi x_n tính theo (3) đều thuộc [a,b]
- g(x) có đạo hàm thoả mãn :

$$|g'(x)| \le q < 1 \ , a < x < b \qquad (4)$$

trong đó q là một hằng số thì phương pháp lặp (3) hội tụ
Ta có thể minh hoạ phép lặp trên bằng hình vẽ a và b.

Cách đưa phương trình f(x) = 0 về dạng x = g(x) được thực hiện như sau:ta thấy f(x) = 0 có thể biến đổi thành x = x + λf(x) với λ ≠ 0.Sau đó đặt x + λf(x) = g(x) sao cho điều kiện (4) được thoả mãn.

Ví dụ:xét phương trình

$$x^3 + x - 1000 = 0$$

Sau bước giải sơ bộ ta có nghiệm $x_1 \in (9,10)$
Nếu đưa phương trình về dạng:

$$x = 1000 - x^3 = g(x)$$

thì dễ thấy $| g'(x) | > 1$ trong khoảng (9,10) nên không thoả mãn điều kiện (4)
Chúng ta đưa phương trình về dạng

$$x = \sqrt[3]{1000 - x}$$

thì ta thấy điều kiện (4) được thoả mãn.Xây dựng dãy xấp xỉ

$$x_{n+1} = \sqrt[3]{1000 - x_n}$$

với x_o chọn bất kì trong (9,10)

Trên cơ sở phương pháp này chúng ta có các chương trình tính toán sau:
Chương trình giải phương trình exp((1/3)*ln(1000-x)) với số lần lặp cho trước

Chương trình 8-1

```
//lap don
#include <conio.h>
#include <stdio.h>
#include <math.h>

void main()
 {
        int i,n;
        float x,x0;
        float f(float);

        clrscr();
        printf("Cho so lan lap n = ");
        scanf("%d",&n);
        printf("Cho gia tri ban dau cua nghiem x0 = ");
        scanf("%f",&x0);
        x=x0;
        for (i=1;i<=n;i++)
          x=f(x);
        printf("Nghiem cua phuong trinh la :%.4f",x);
        getch();
 }

float f(float x)
 {
        float a=exp((1./3.)*log(1000-x));
        return(a);
```

```
}
```

và chương trình giải bài toán bằng phương pháp lặp với sai số cho trước

Chương trình 8-2

```
//lap don
#include <conio.h>
#include <stdio.h>
#include <math.h>
void main()
 {
        int i;
        float epsi,x,x0,y;
        float f(float);

        clrscr();
        printf("Cho sai so epsilon = ");
        scanf("%f",&epsi);
        printf("Cho gia tri ban dau cua nghiem x0 = ");
        scanf("%f",&x0);
        x=x0;
        y=f(x);
        if (abs(y-x)>epsi)
          {
                x=y;
                y=f(x);
          }
        printf("Nghiem cua phuong trinh la %.6f",y);
        getch();
 }

float f(float x)
 {
        float a=exp((1./3.)*log(1000-x));
        return(a);
 }
```

Cho giá trị đầu x_o = 1.Kết quả tính toán x = 9.966555

§3.PHƯƠNG PHÁP CHIA ĐÔI CUNG

Giả sử cho phương trình f(x) = 0 với f(x) liên tục trên đoạn [a,b] và f(a).f(b) < 0.Chia đoạn [a,b] thành 2 phần bởi chính điểm chia (a + b)/2.

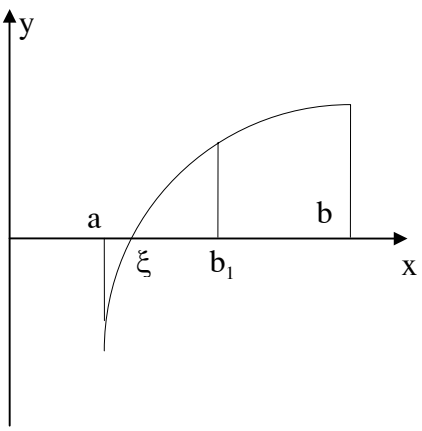
1.Nếu f((a+b)/2) = 0 thì ξ = (a+b)/2

2.Nếu f((a+b)/2) ≠ 0 thì chọn [a,(a + b)/2] hay [(a + b)/2,b] mà giá trị hàm hai đầu trái dấu và kí hiệu là $[a_1,b_1]$.Đối với $[a_1,b_1]$ ta lại tiến hành như [a,b]

Cách làm trên được mô tả trong chương trình sau dùng để tìm nghiệm của phương trình :

$$x^4 + 2x^3 - x - 1 = 0$$

trên đoạn [0,1]



Chương trình 8-3

```
//chia doi cung
#include <conio.h>
#include <stdio.h>
#include <math.h>
#define  epsi  0.00001
void main()
 {
       float x0,x1,x2;
       float y0,y1,y2;
  float f(float);
       int maxlap,demlap;

  clrscr();
       printf("Tim nghiem cua phuong trinh phi tuyen");
       printf("\nbang cach chia doi cung\n");
       printf("Cho cac gia tri x0,x1,maxlap\n");
       printf("Cho gia tri  x0 = ");
       scanf("%f",&x0);
       printf("Cho gia tri  x1 =  ");
       scanf("%f",&x1);
       printf("Cho so lan lap maxlap =  ");
       scanf("%d",&maxlap);
       y0=f(x0);
       y1=f(x1);
       if ((y0*y1)>0)
         {
              printf("Nghiem khong nam trong doan x0 - x1\n");
              printf(" x0 = %.2f\n",x0);
              printf(" x1 = %.2f\n",x1);
              printf(" f(x0) = %.2f\n",y0);
              printf(" f(x1) = %.2f\n",y1);
         }
       demlap=0;
       do
         {
```

```
            x2=(x0+x1)/2;
            y2=f(x2);
            y0=f(x0);
            if (y0*y2>0)
              x0=x2;
            else
              x1=x2;
            demlap=demlap+1;
    }
        while(((abs((y2-y0))>epsi)||(demlap<maxlap)));
        if (demlap>maxlap)
          {
            printf("Phep lap khong hoi tu sau %d lan lap ",maxlap);
            printf(" x0 = %.2f\n",x0);
            printf(" x1 = %.2f\n",x1);
            printf(" f(x2) = %.2f\n",y2);
          }
        else
          {
            printf("Phep lap hoi tu sau %d lan lap\n",demlap);
            printf("Nghiem x = %.2f",x2);
          }
        getch();
  }

float f(float x)
  {
        float a=x*x*x*x+2*x*x*x-x-1 ;
        return(a);
  }
```

Kết quả tính cho nghiệm:x = 0.87

§4.PHƯƠNG PHÁP DÂY CUNG

Giả sử f(x) liên tục trên trên đoạn [a,b] và f(a).f(b) < 0.Cần tìm nghiệm của f(x) = 0.Để xác định ta xem f(a) < 0 và f(b) > 0.Khi đó thay vì chia đôi đoạn [a,b] ta chia [a,b] theo tỉ lệ -f(a)/f(b).Điều đó cho ta nghiệm gần đúng :

$$x_1 = a + h_1$$

Trong đó

$$h_1 = \frac{-f(a)}{-f(a)+f(b)}(b-a)$$

Tiếp theo dùng cách đó với đoạn [a,x_1] hay [x_1,b] mà hai đầu hàm nhận giá trị trái dấu ta được nghiệm gần đúng x_2 v.v.

Về mặ hình học,phương pháp này có nghĩa là kẻ dây cung của đường cong f(x) qua hai điểm A[a,f(a)] và B[b,f(b)].Thật vậy phương trình dây cung AB có dạng :

$$\frac{x-a}{b-a}=\frac{y-f(a)}{f(b)-f(a)}$$

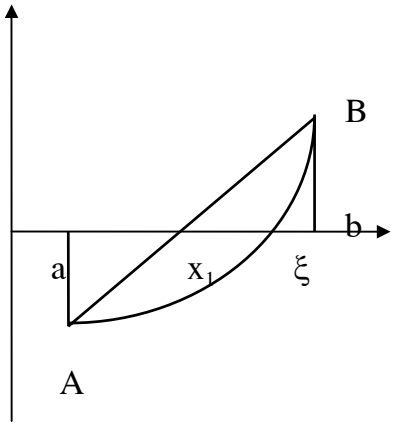
Cho $x = x_1$ $y = 0$ ta có

$$x_1 = a - \frac{f(a)}{f(b)-f(a)}(b-a)$$

Trên cơ sở của phương pháp ta có chương trình tính nghiệm của phương trình

$$x^4 + 2x^3 - x - 1 = 0$$

trên đoạn [0,1]



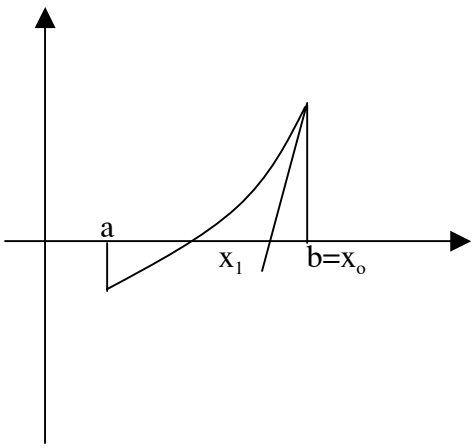
Chương trình 8-4

```
//phuong phap day cung
#include <conio.h>
#include <stdio.h>
#include <math.h>
#define epsi  0.00001

void main()
 {
        float a,b,fa,fb,dx,x;
        float f(float);

        clrscr();
        printf("Tim nghiem cua phuong trinh phi tuyen\n");
        printf("bang phuong phap day cung\n");
        printf("Cho cac gia tri a,b\n");
        printf("Cho gia tri cua a = ");
        scanf("%f",&a);
        printf("Cho gia tri cua b = ");
        scanf("%f",&b);
        fa=f(a);
        fb=f(b);
        dx=fa*(b-a)/(fa-fb);
        while (fabs(dx)>epsi)
         {
                x=a+dx;
                fa=f(x);
                if((fa*fb)<=0)
                  a=x;
                else
                  b=x;
                fa=f(a);
                fb=f(b);
                dx=fa*(b-a)/(fa-fb);
         }
```

```
        printf("Nghiem x = %.3f",x);
        getch();
 }

float f(float x)
 {
        float e=x*x*x*x+2*x*x*x-x-1;
        return(e);
 }
```

Kết quả tính cho nghiệm:x = 0.876

§5.PHƯƠNG PHÁP LẶP NEWTON



Phương pháp lặp Newton (còn gọi là phương pháp tiếp tuyến) được dùng nhiều vì nó hội tụ nhanh.Giả sử f(x) có nghiệm là ξ đã được tách trên đoạn [a,b] đồng thời f'(x) và f''(x) liên tục và giữ nguyên dấu trên đoạn [a,b].Khi đã tìm được xấp xỉ nào đó $x_n \in [a,b]$ ta có thể kiện toàn nó theo phương pháp Newton.Từ mút B ta vẽ tiếp tuyến với đường cong.Phương trình đường tiếp tuyến là

$$y - f(x_0) = f'(b)(x - x_0)$$

Tiếp tuyến này cắt trục hoành tại điểm có y=0,nghĩa là :

$$-f(x_0) = f'(b)(x_1 - x_0)$$

hay :

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Từ x_1 ta lại tiếp tục vẽ tiếp tuyến với đường cong thì giao điểm x_i sẽ tiến tới nghiệm của phương trình.

Việc chọn điểm ban đầu x_o rất quan trọng.Trên hình vẽ trên ta thấy nếu chọn điểm ban đầu x_o = a thì tiếp tuyến sẽ cắt trục tại một điểm nằm ngoài đoạn [a,b].Chọn x_o = b sẽ thuận lợi cho việc tính toán.Chúng ta có định lí :

Nếu f(a).f(b) < 0 ; f(x) và f''(x) khác không và giữ nguyên dấu xác định khi x ∈ [a,b] thì xuất phát từ $x_o \in [a,b]$ thoả mãn điều kiện $f(x_o).f''(x_o) > 0$ có thể tính theo phương pháp Newton nghiệm ξ duy nhất với độ chính xác tuỳ ý.

Khi dùng phương pháp Newton cần lấy x_o là đầu mút của đoạn [a,b] để tại đó $f(x_o).f''(x_o) > 0$.Áp dụng lí thuyết trên chúng ta xây dựng chương trình tính sau:

Chương trình 8-5

```
//phuong phap Newton
#include <conio.h>
#include <stdio.h>
```

```
#include <math.h>
#include <stdlib.h>
#define n 50
#define epsi 1e-5

void main()
  {
        float t,x0;
        float x[n];
        int i;
        float f(float);
        float daoham(float);

        clrscr();
        printf("Tim nghiem cua phuong trinh phi tuyen\n");
        printf("bang phuong phap lap Newton\n");
        printf("Cho gia tri cua x0 = ");
        scanf("%f",&x0);
        i=1;
        x[i]=x0;
        do
          {
                x[i+1] = x[i]-f(x[i])/daoham(x[i]);
                t = fabs(x[i+1]-x[i]);
                x[i]=x[i+1];
                i=i+1;
                if (i>100)
                  {
                        printf("Bai toan khong hoi tu\n");
                        getch();
                        exit(1);
                        }
                else
        ;
          }
        while (t>=epsi);
        printf("Nghiem x = %.5f",x[i]);
        getch();
  }

float f(float x)
  {
        float a=x*x-x-2;
        return(a);
  }

float daoham(float x)
  {
         float d=2*x-1;
         return(d);
```

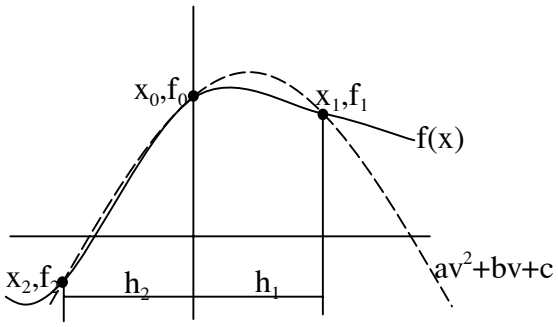
}

Chương trình này được dùng xác định nghiệm của hàm đã được định nghĩa trong function.Trong trường hợp này phương trình đó là:x^2 - x -1 = 0.Kết quả tính với giá trị đầu x_o = 0 cho nghiệm x = 2.

§6.PHƯƠNG PHÁP MULLER

Trong phương pháp dây cung khi tìm nghiệm trong đoạn [a,b] ta xấp xỉ hàm bằng một đường thẳng.Tuy nhiên để giảm lượng tính toán và để nghiệm hội tụ nhanh hơn ta có thể dùng phương pháp Muller.Nội dung của phương pháp này là thay hàm trong đoạn [a,b] bằng một đường cong bậc 2 mà ta hoàn toàn có thể tìm nghiêm chín xác của nó.Gọi các điểm đó có hoành độ lần lượt là a = x_2,b = x_1 và ta chọn thêm một điểm x_0 nằm trong đoạn [x_2,x_1].Gọi

$h_1 = x_1 - x_0$
$h_2 = x_0 - x_2$
$v = x - x_0$
$f(x_0) = f_0$
$f(x_1) = f_1$
$f(x_2) = f_2$
$\gamma = \frac{h_2}{h_1}$



Qua 3 điểm này ta có một đường parabol :

$$y = av^2 + bv + c$$

Ta tìm các hệ số a,b,c từ các giá trị đã biết v:

$v = 0 (x = x_0)$ $\quad a(0)^2 + b(0) + c = f_0$
$v = h_1 (x = x_1)$ $\quad ah_1^2 + bh_1 + c = f_1$
$v = h_2 (x = x_2)$ $\quad ah_2^2 + bh_2 + c = f_2$

Từ đó ta có :

$$a = \frac{\gamma f_1 - f_0(1+\gamma) + f_2}{\gamma h_1^2 (1+\gamma)}$$

$$b = \frac{f_1 - f_0 - ah_1^2}{h_1}$$

$$c = f_0$$

Sau đó ta tìm nghiệm của phương trình $av^2 + bv + c = 0$ và có :

$$n_{1,2} = x_0 - \frac{2c}{b \pm \sqrt{b^2 - 4ac}}$$

Tiếp đó ta chọn nghiệm gần x_0 nhất làm một trong 3 điểm để tính xấp xỉ mới.Các điểm này được chọn gần nhau nhất.
Tiếp tục quá trình tính đến khi đạt độ chính xác yêu cầu thì dừng lại.

Ví dụ:Tìm nghiệm của hàm f(x) = sin(x) - x/2 trong đoạn [1.8,2.2].Ta chọn x_0 = 2

Ta có : $x_0 = 2$ $\quad f(x_0) = -0.0907$ $\quad h_1 = 0.2$
$x_1 = 2.2$ $\quad f(x_1) = -0.2915$ $\quad h_2 = 0.2$
$x_2 = 1.8$ $\quad f(x_2) = 0.07385$ $\quad \gamma = 1$

Vậy thì :

$$a = \frac{1\times(-0.2915)-(-0.0907)\times(1+1)+0.07385}{1\times 0.2^2\times(1+1)} = -0.45312$$

$$b = \frac{-0.2915-(-0.097)-(-0.45312)\times 0.2^2}{0.2} = -0.91338$$

$$c = -0.0907$$

Ta có nghiệm gần x_0 nhất là :

$$n_1 = 2.0 - \frac{2\times(-0.0907)}{-0.91338-\sqrt{(-0.91338)^2-4\times(-0.45312)\times(-0.0907)}} = 1.89526$$

Với lần lặp thứ hai ta có :

$x_0 = 1.89526$ $f(x_0) = 1.9184\times10^{-4}$ $h_1 = 0.10474$

$x_1 = 2.0$ $f(x_1) = -0.0907$ $h_2 = 0.09526$

$x_2 = 1.8$ $f(x_2) = 0.07385$ $\gamma = 0.9095$

Vậy thì :

$$a = \frac{0.9095\times(-0.0907)-(1.9184\times10^{-4})\times 1.9095+0.07385}{0.9095\times 0.10474^2\times 1.9095} = -0.4728$$

$$b = \frac{-0.0907-1.9184\times10^{-4}-(-0.4728)\times 0.10474^2}{0.10474} = -0.81826$$

$$c = 1.9184\times10^{-4}$$

Ta có nghiệm gần x_0 nhất là :

$$n_1 = 1.89526 - \frac{2\times 1.9184\times10^{-4}}{-0.81826-\sqrt{(0.81826)^2-4\times(-0.4728)\times 1.9184\times10^{-4}}} = 1.89594$$

Ta có thể lấy $n_1 = 1.895494$ làm nghiệm của bài toán

Chương trình giải bài toán bằng phương pháp Muller như sau:

Chương trình 8-6

```
//phuong phap Muller
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void main()
 {
        float x0,x1,x2,h1,h2,eps;
        float a,b,c,gamma,n1,n2,xr;
        int dem;
        float f(float);

        clrscr();
        printf("PHUONG PHAP MULLER\n");
        printf("\n");
        printf("Cho khoang can tim nghiem [a,b]\n");
        printf("Cho gia tri duoi a = ");
        scanf("%f",&x2);
```

```
        printf("Cho gia tri tren b = ");
        scanf("%f",&x1);
        if (f(x1)*f(x2)>0)
          {
                printf("\n");
                printf("Nghiem khong nam trong doan nay\n");
                getch();
                exit(1);
          }
        eps=1e-5;
        x0=(x1+x2)/2;
        dem=0;
        do
          {
                dem=dem+1;
                h1=x1-x0;
                h2=x0-x2;
                gamma=h2/h1;
                a=(gamma*f(x1)-
f(x0)*(1+gamma)+f(x2))/(gamma*(h1*h1)*(1+gamma));
                b=(f(x1)-f(x0)-a*(h1*h1))/h1;
                c=f(x0);
                if ((a==0)&&(b!=0))
                  {
                        n1=-c/b;
                        n2=n1;
                  }
                if ((a!=0)&&(b==0))
                  {
                        n1=(-sqrt(-c/a));
                        n2=(sqrt(-c/a));
                  }
                if ((a!=0)&&(b!=0))
                  {
                        n1=x0-2*c/(b+(sqrt(b*b-4*a*c)));
                        n2=x0-2*c/(b-(sqrt(b*b-4*a*c)));
                  }
                if (fabs(n1-x0)>fabs(n2-x0))
                  xr=n2;
                else
                  xr=n1;
                if (xr>x0)
                  {
                        x2=x0;
                        x0=xr;
                  }
                else
                  {
                        x1=x0;
                        x0=xr;
```

```
                }
            }
        while (fabs(f(xr))>=eps);
        printf("\n");
        printf("Phuong trinh co nghiem x = %.5f sau %d lan lap",xr,dem);
        getch();
}

float f(float x)
 {
        float a=sin(x)-x/2;
        return(a);
 }
```

§7.PHƯƠNG PHÁP LẶP BERNOULLI

Có nhiều phương pháp để tìm nghiệm của một đa thức.Ta xét phương trình :

$$a_o x^n + a_1 x^{n-1} + \cdots + a_n = 0$$

Nghiệm của phương trình trên thoả mãn định lí:*Nếu* $max\{|a_1|,|a_2|,...,|a_n|\} = A$ *thì các nghiệm của phương trình thoả mãn điều kiện* $|x| < 1 + A/|a_0|$

Phương pháp Bernoulli cho phép tính toán nghiệm lớn nhất α của một đa thức $P_n(x)$ có n nghiệm thực phân biệt.Sau khi tìm được nghiệm lớn nhất α ta chia đa thức $P_n(x)$ cho $(x - \alpha)$ và nhận được đa thức mới $Q_{n-1}(x)$.Tiếp tục dùng phương pháp Bernoulli để tìm nghiệm lớn nhất của $Q_{n-1}(x)$.Sau đó lại tiếp tục các bước trên cho đến khi tìm hết các nghiệm của $P_n(x)$.

Chúng ta khảo sát phương trình phương trình sai phân φ có dạng như sau :

$$\varphi = a_o y_{k+n} + a_1 y_{k+n-1} + ... + a_n y_k = 0 \qquad (1)$$

Đây là một phương trình sai phân tuyên tính hệ số hằng.Khi cho trước các giá trị đầu $y_o, y_1, ..y_{n-1}$ ta tìm được các giá trị $y_n, y_{n+1}, ..$Chúng được gọi là nghiệm của phương trình sai phân tuyến tính (1).

Đa thức

$$P_n(x) = a_0 x^n + a_1 x^{n-1} + ..+ a_{n-1} x + a_n \qquad (2)$$

với cùng một hệ số a_i như (1) được gọi là đa thức đặc tính của phương trình sai phân tuyến tính (1).Nếu (2) có n nghiệm phân biệt $x_1, x_2, .., x_n$ thì (1) có các nghiệm riêng là

$$y_i = x_i^k$$

Nếu y_i là các nghiệm của phương trình sai phân là tuyến tính (1),thì

$$y_k = c_1 x_1^k + c_2 x_2^k + ... + c_n x_n^k \qquad (3)$$

với các hệ số c_i bất kì cũng là nghiệm của phương trình sai phân tuyến tính hệ số hằng (1).

Nếu các nghiệm là sao cho :

$$|x_1| \geq |x_2| \geq ... |x_n|$$

Vậy thì

$$y_k = c_1 x_1^k [1 + \frac{c_1}{c_2}(\frac{x_2}{x_1})^k + \cdots]$$

và

$$y_{k+1} = c_1 x_1^{k+1} [1 + \frac{c_1}{c_2}(\frac{x_2}{x_1})^{k+1} + \cdots]$$

do đó : $$\frac{y_{k+1}}{y_k}=x_1\frac{[1+\frac{c_2}{c_1}(\frac{x_2}{x_1})^{k+1}+...]}{[1+\frac{c_2}{c_1}(\frac{x_2}{x_1})^{k}+...]}$$

do $x_1 > x_2$

nên: $(\frac{x_2}{x_1})^k,(\frac{x_2}{x_1})^k.....\to 0$ khi $k\to\infty$

vậy thì : $\frac{y_{k+1}}{y_k}\to\infty$ khi $k\to\infty$

Nghĩa là : $x_1=\lim_{k\to\infty}\frac{y_{k+1}}{y_k}$

Nếu phương trình vi phân gồm n+1 hệ số,một nghiệm riêng y_k có thể được xác định từ n giá trị $y_{k-1},y_{k-2},...,y_{n-1}$.Điều cho phép tính toán bằng cách truy hồi các nghiệm riêng của phương trình vi phân.

Để tính nghiệm lớn nhất của đa thức,ta xuất phát từ các nghiệm riêng $y_1 = 0,y_1 = 0,..,y_n = 1$ để tính y_{n+1}.Cách tính này được tiếp tục để tính y_{n+2} xuất phát từ $y_1 = 0,y_2 = 0,..,y_{n+1}$ và tiếp tục cho đến khi y_{k+1}/y_k không biến đổi nữa.Trị số của y_{k+n} được tính theo công thức truy hồi :

$$y_{k+n}=-\frac{1}{a_o}(a_1y_{k+n-1}+...+a_ny_k) \qquad (4)$$

Ví dụ:Tính nghiệm của đa thức $P_n(x) = P_3(x) = x^3 - 10x^2 + 31x - 30$.Như vậy $a_o = 1,a_1 = -10,a_2 = 31$ và $a_3 = -30$.Phương trình sai phân tương ứng là :

$y_{k+3} -10y_{k+2} + 31y_{k+1} - 30y_k = 0$

Ta cho trước các giá trị $y_1 = 0$; $y_2 = 0$ và $y_3 = 1$.Theo (4) ta tính được :

$y_4 = - (-10y_3 + 31y_2 - 30y_1) = 10$
$y_5 = - (-10y_4 + 31y_3 - 30y_2) = 69$
$y_6 = - (-10y_5 + 31y_5 - 30y_3) = 410$
$y_7 = - (-10y_6 + 31y_5 - 30y_4) = 2261$
$y_8 = - (-10y_7 + 31y_6 - 30y_5) = 11970$
$y_9 = - (-10y_8 + 31y_7 - 30y_6) = 61909$
$y_{10} = - (-10y_9 + 31y_8 - 30y_8) = 315850$
$y_{11} = - (-10y_{10} + 31y_9 - 30y_8) = 1598421$
$y_{12} = - (-10y_{11} + 31y_{10} - 30y_9) = 8050130$
$y_{13} = - (-10y_{12} + 31y_{11} - 30y_{10}) = 40425749$
$y_{14} = - (-10y_{13} + 31y_{12} - 30y_{11}) = 202656090$
$y_{15} = - (-10y_{14} + 31y_{13} - 30y_{12}) = 1014866581$
$y_{16} = - (-10y_{15} + 31y_{14} - 30y_{13}) = 5079099490$
$y_{17} = - (-10y_{16} + 31y_{15} - 30y_{14}) = 24409813589$
$y_{18} = - (-10y_{17} + 31y_{16} - 30y_{15}) = 127092049130$
$y_{19} = - (-10y_{18} + 31y_{17} - 30y_{16}) = 635589254740$

Tỉ số các số y_{k+1}/y_k lập thành dãy :

10 ; 6.9 ; 5.942 ; 5.5146 ; 5.2941 ; 5.172 ; 5.1018 ; 5.0607 ; 5.0363 ; 5.0218 ; 5.013 ; 5.0078 ; 5.0047 ; 5.0028 ; 5.0017 ; 5.001

nghĩa là chúng sẽ hội tụ tới nghiệm lớn nhất là 5 của đa thức

Chương trình 8-7

```
//phuong phap Bernoulli
#include <conio.h>
#include <stdio.h>
```

```
#include <math.h>
#include <stdlib.h>
#define max  50

void main()
  {
        float a[max],y[max];
        int k,j,i,n,l;
        float s,e1,e2,x0,x1,x;

        clrscr();
        printf("Cho bac cua da thuc can tim nghiem n = ");
        scanf("%d",&n);
        e1=1e-5;
        printf("Cho cac he so cua da thuc can tim nghiem\n");
        for (i=0;i<=n;i++)
          {
                printf("a[%d] = ",i);
                scanf("%f",&a[i]);
          }
        for (k=0;k<=n;k++)
          a[k]=a[k]/a[0];
          tt: x1=0;
        for (k=2;k<=n;k++)
          y[k]=0;
        y[1]=1;
        l=0;
        do
          {
                l=l+1;
                s=0;
                for (k=1;k<=n;k++)
                  s=s+y[k]*a[k];
                y[0]=-s;
                x=y[0]/y[1];
                e2=fabs(x1 - x);
                x1=x;
                for (k=n;k>=1;k--)
                y[k]=y[k-1];
          }
        while((l<=50)||(e2>=e1));
        if(e2>=e1)
          {
                printf("Khong hoi tu");
                getch();
                exit(1);
          }
        else
          printf("Nghiem x = %.4f\n",x);
        n=n-1;
```

```
        if (n!=0)
          {
                a[1]=a[1]+x;
                for (k=2;k<=n;k++)
                  a[k]=a[k]+x*a[k-1];
                goto tt;
          }
        getch();
}
```

Kết quả nghiệm của đa thức $x^3 - 10x^2 + 31x - 30$ là:5 ; 3 và 2

§8.PHƯƠNG PHÁP LẶP BIRGE - VIETTE

Các nghiệm thực,đơn giản của một đa thức $P_n(x)$ được tính toán khi sử dụng phương pháp Newton

$$x_{i+1} = x_i - \frac{P_n(x_i)}{P'_n(x_i)} \qquad (1)$$

Để bắt đầu tính toán cần chọn một giá trị ban đầu x_o.Chúng ta có thể chọn một giá trị x_o nào đó,ví dụ :

$$x_o = -\frac{a_n}{a_{n-1}}$$

và tính tiếp các giá trị sau :

$$x_1 = x_o - \frac{P_n(x_o)}{P'_n(x_o)}$$

$$x_2 = x_1 - \frac{P_n(x_1)}{P'_n(x_1)}$$

Tiếp theo có thể đánh giá $P_n(x_i)$ theo thuật toán Horner :

$$P_0 = a_0$$
$$P_1 = P_0x_i + a_1 \qquad (2)$$
$$P_2 = P_1x_i + a_2$$
$$P_3 = P_2x_i + a_3$$
$$.....$$
$$P(x_i) = P_n = P_{n-1}x_i + a_n$$

Mặt khác khi chia đa thức $P_n(x)$ cho một nhị thức $(x - x_i)$ ta được :

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

với $b_n = P_n(x_i)$.Đa thức $P_{n-1}(x)$ có dạng :

$$P_{n-1}(x) = b_ox^{n-1} + b_1x^{n-2} + p_3x^{n-3} + ..+ b_{n-2}x + b_{n-1} \qquad (4)$$

Để xác định các hệ số của đa thức (4) ta thay (4) vào (3) và cân bằng các hệ số với đa thức cần tìm nghiệm $P_n(x)$ mà các hệ số a_i đã cho:

$$(x - x_i)(b_ox^{n-1} + b_1x^{n-2} + b_3x^{n-3} + ..+ b_{n-2}x + b_{n-1}) + b_n$$
$$= a_ox^n + a_1x^{n-1} + a_2x^{n-2} + ...+ a_{n-1}x + a_n \qquad (5)$$

Từ (5) rút ra :

$$b_o = a_o$$
$$b_1 = a_1 + b_ox_i \qquad (6)$$
$$b_2 = a_2 + b_1x_i$$
$$.....$$
$$b_k = a_k + b_{k-1}x_i$$
$$.....$$

$b_n = a_n + b_{n-1}x_i = P_n(x_i)$

Đạo hàm (3) ta được :

$$P'_n(x) = (x - x_i)P'_{n-1}(x) + P_{n-1}(x)$$

và

$$P'_n(x_i) = P_{n-1}(x_i) \qquad (7)$$

Như vậy với một giá trị x_i nào đó theo (2) ta tính được $P_n(x_i)$ và kết hợp (6) với (7) tính được $P'_n(x_i)$.Thay các kết quả này vào (1) ta tính được giá trị x_{i+1}.Quá trình được tiếp tục cho đến khi $| x_{i+1} - x_i | < \varepsilon$ hay $P_n(x_{i+1}) \approx 0$ nên $\alpha_1 \approx x_{i+1}$ là một nghiệm của đa thức.

Phép chia $P_n(x)$ cho $(x - \alpha_1)$ cho ta $P_{n-1}(x)$ và một nghiệm mới khác được tìm theo cách trên khi chọn một giá trị x_o mới hay chọn chính $x_o = \alpha_1$.Khi bậc của đa thức giảm xuống còn bằng 2 ta dùng các công thức tìm nghiệm của tam thức để tìm các nghiệm còn lại.

Ví dụ:tìm nghiệm của đa thức $P_3(x) = x^3 - x^2 - 16x + 24$

$a_o = 1$ $a_1 = -1$ $a_2 = -16$ $a_3 = 24$

Chọn $x_o = 3.5$ ta có :

$P_o = a_o = 1$

$P_1 = a_1 + p_o x_0 = -1 + 3.5*1 = 2.5$

$P_2 = a_2 + p_1 x_0 = -16 + 3.5*2.5 = -7.25$

$P_3 = a_3 + p_2 x_0 = 24 + 3.5*(-7.25) = -1.375$

$b_0 = a_0 = 1;$

$b_1 = a_1 + b_o x_0 = -1 + 3.5*1 = 2.5$

$b_2 = a_2 + b_1 x_0 = -16 + 3.5*2.5 = -7.25$

$P_2(3.5) = b_0 x^2 + b_1 x + b_2 = 13.75$

$$x_1 = x_0 - \frac{P_n(x_0)}{P'_n(x_0)} = 3.5 + \frac{1.375}{13.75} = 3.6$$

Lặp lại bước tính trên cho x_1 ta có:

$P_o = a_o = 1$

$P_1 = a_1 + p_o x_1 = -1 + 3.6*1 = 2.6$

$P_2 = a_2 + p_1 x_1 = -16 + 3.6*2.6 = -6.64$

$P_3 = a_3 + p_2 x_1 = 24 + 3.6*(-6.64) = -0.096$

$b_o = a_o = 1$

$b_1 = a_1 + b_o x_1 = -1 + 3.6*1 = 2.6$

$b_2 = a_2 + p_1 x_1 = -16 + 3.6*2.6 = -6.64$

$P_2(3.6) = b_0 x^2 + b_1 x + b_2 = 15.68$

$$x_2 = x_1 - \frac{P_n(x_1)}{P'_n(x_1)} = 3.6 + \frac{0.096}{15.68} = 3.606$$

Quá trình cứ thế tiếp tục cho đến khi sai số chấp nhận được.Chương trình dưới đây mô tả thuật tính trên.

Chương trình 8-8

```
//phuong phap Birge-Viette
#include <conio.h>
#include <stdio.h>
#include <math.h>
#define  max  20
```

```
void main()
 {
        float a[max],p[max],d[max],x[max];
        int k,j,i,n;
        float e1,e2,x0,x1;

        clrscr();
        printf("Cho bac cua da thuc n = ");
        scanf("%d",&n);
        e1=0.0001;
        printf("Cho cac he so cua da thuc can tim nghiem\n");
        for (i=0;i<=n;i++)
          {
                printf("a[%d] = ",i);
                scanf("%f",&a[i]);
          }
        x0=a[0];
        for (i=0;i<=n;i++)
          a[i]=a[i]/x0;
        printf("Nghiem cua phuong trinh  : \n");
        tt:x0=-a[n]/a[n-1];
        j=0;
        do
          {
                j=j+1;
                p[1]=x0+a[1];
                d[1]=1.0;
                for (k=2;k<=n;k++)
                  {
                        p[k]=p[k-1]*x0+a[k];
                        d[k]=d[k-1]*x0+p[k-1];
                  }
                x1=x0-p[n]/d[n];
                e2=fabs(x1-x0);
                if (e2>e1)
                x0=x1;
          }
        while((j<=50)||(e2>=e1));
        if (e2>=e1)
          printf("Khong hoi tu");
        else
          printf(" x = %.4f\n",x1);
        n=n-1;
        if (n!=0)
          {
                for (k=1;k<=n;k++)
                  a[k]=p[k];
                goto tt;
          }
```

```
    getch();
}
```

Dùng chương trình trên để tìm nghiệm của đa thức $x^4 + 2x^3 - 13x^2 - 14x + 24$ ta được các nghiệm là:-4 ; 3 ; -2 và 1.

§9.PHƯƠNG PHÁP NGOẠI SUY AITKEN

Xét phương pháp lặp :

$$x = f(x) \quad (1)$$

với f(x) thoả mãn điều kiện hội tụ của phép lặp,nghĩa là với mọi $x \in [a,b]$ ta có :

$$| f'(x) | \le q < 1 \quad (2)$$

Như vậy :

$$x_{n+1} = f(x_n) \quad (3)$$

$$x_n = f(x_{n-1}) \quad (4)$$

Trừ (3) cho (4) và áp dụng định lí Lagrange cho vế phải với $c \in [a,b]$ ta có :

$$x_{n+1} - x_n = f(x_n) - f(x_{n-1}) = (x_n - x_{n-1})f'(c) \quad (5)$$

Vì phép lặp (1) nên :

$$| x_{n+1} - x_n | \le q | x_n - x_{n-1} | \quad (6)$$

Do (6) đúng với mọi n nên cho n = 1 , 2 , 3 , . . . ta có :

$$| x_2 - x_1 | \le q | x_1 - x_o |$$
$$| x_3 - x_2 | \le q | x_2 - x_1 |$$
. . . . . .
$$| x_{n+1} - x_n | \le q | x_n - x_{n-1} |$$

Điều này có nghĩa là dãy $x_{i+1} - x_i$, một cách gần đúng,là một cấp số nhân . Ta cũng coi rằng dãy $x_n - y$ với y là nghiệm đúng của (1) , gần đúng như một cấp số nhân có công sai q . Như vậy :

$$\frac{x_{n+1} - y}{x_n - y} = q < 1 \qquad (7)$$

hay : $$x_{n+1} - y = q(x_n - y) \qquad (8)$$

Tương tự ta có : $$x_{n+2} - y = q(x_{n+1} - y) \qquad (9)$$

Từ (8) và (9) ta có :

$$q = \frac{x_{n+2} - x_{n+1}}{x_{n+1} - x_n} \qquad (10)$$

Thay giá trị của q vừa tính ở (10) vào biểu thức của q ở trên ta có :

$$y = x_n - \frac{(x_n - x_{n+1})^2}{x_n - 2x_{n+1} + x_{n+1}} \qquad (11)$$

Công thức (11) được gọi là công thức ngoại suy Adam.Như vậy theo (11) trước hết ta dùng phương pháp lặp để tính giá trị gần đúng x_{n+2},x_{n+1},x_n của nghiệm và sau đó theo (11) tìm được nghiệm với sai số nhỏ hơn.

Để làm ví dụ chúng ta xét phương trình :

$$\ln x - x^2 + 3 = 0$$

Ta đưa về dạng lặp :

$$x = \sqrt{\ln(x) + 3}$$

$$f'(x) = \frac{1}{2x\sqrt{\ln x + 3}}$$

Phép lặp hội tụ trong đoạn [0.3,∞].Ta cho $x_1 = 1$ thì tính được :

x_2 = 1,7320508076
x_3 = 1.883960229
x_4 = 1.90614167
y = 1.909934347

Để giảm sai số ta có thể lặp nhiều lần

Chương trình 8-9

```
//phuong phap Aitken
#include <conio.h>
#include <stdio.h>
#include <math.h>

#define  m  5

void main()
 {
      float x[m];
```

```
        float epsi,n,y;
        int i,z;
        float f(float);

        clrscr();
        printf("Cho tri so ban dau x[1] = ");
        scanf("%f",&x[1]);
        printf("Cho tri so sai so epsilon = ");
        scanf("%f",&epsi);
        printf("\n");
        printf( "Ngoai suy Aitken cua ham\n");
        z=0;
        while (z<=20)
            {
                for (i=2;i<=4;i++)
                  x[i]=f(x[i-1]);
                n=x[4]-2*x[3]+x[2];
                if ((fabs(n)<1e-09)||(fabs(x[1]-x[2])<epsi*fabs(x[1])))
                  z=20;
                else
                  {
                        y=x[2]-(x[3]-x[2])*(x[3]-x[2])/n;
                        if (z>20)
                          printf("Khong hoi tu sau hai muoi lan lap\n");
                        x[1]=y;
                  }
                z=z+1;
          }
        printf("Nghiem cua phuong trinh y = %.6f",y);
        getch();
  }

float f(float x)
  {
        float s=sqrt(log(x)+3);
        return(s);
  }
```

Với giá trị ban đầu là 1 và sai số là 1e-8,chương trình cho kết quả y = 1.9096975944

§10.PHƯƠNG PHÁP BAIRSTOW

Nguyên tắc của phương pháp Bairstow là trích từ đa thức $P_n(x)$ một tam thức $Q_2(x) = x^2 - sx + p$ mà ta có thể tính nghiệm thực hay nghiệm phức của nó một cách đơn giản bằng các phương pháp đã biết.

Việc chia đa thức $P_n(x)$ cho tam thức $Q_2(x)$ đưa tới kết quả :

$$P_n(x) = Q_2(x).P_{n-2}(x) + R_1(x)$$

với $P_n(x) = a_o x^n + a_1 x^{n-1} + a_2 x^{n-2} + ... + a_n$

$Q_2(x) = x^2 - sx + p$

$$P_{n-2}(x) = b_o x^{n-2} + b_1 x^{n-3} + b_2 x^{n-4} + ... + b_{n-2}$$
$$R_1(x) = \alpha x + \beta$$

Để có được một thương đúng,cần tìm các giá trị của s và p sao cho $R_1(x) = 0$ (nghĩa là α và β triệt tiêu).Với s và p đã cho,các hệ số b của đa thức $P_{n-2}(x)$ và các hệ số α và β được tính bằng phương pháp truy hồi.Các công thức nhận được khi khai triển biểu thức $P_n(x) = Q_2(x).P_{n-2}(x) + R_1(x)$ và sắp xếp lại các số hạng cùng bậc :

$$a_o x^n + a_1 x^{n-1} + a_2 x^{n-2} + ... + a_n = (x^2 - sx + p)(b_o x^{n-2} + b_1 x^{n-3} + b_2 x^{n-4} + ... + b_{n-2})$$

Số hạng bậc	Hệ số của $P_n(x)$	Hệ số của $Q_2(x).P_{n-2}(x)$
x^n	a_o	b_o
x^{n-1}	a_1	$b_1 - sb_o$
x^{n-2}	a_2	$b_2 - sb_1 + pb_o$
.....	.....	.....
x^{n-k}	a_k	$b_k - sb_{k-1} + pb_{k-2}$
x	a_{n-1}	$\alpha - sb_{n-2} + pb_{n-3}$
x^o	a_n	$\beta + pb_{n-2}$

Như vậy : (1)

$$b_o = a_o$$
$$b_1 = a_1 + sb_o$$
$$b_2 = a_2 + sb_1 - pb_o$$
.....
$$b_k = a_k + sb_{k-1} - pb_{k-2}$$
$$\alpha = a_{n-1} + sb_{n-2} - pb_{n-3}$$
$$\beta = a_n - pb_{n-2}$$

Chúng ta nhận thấy rằng α được tính toán xuất phát từ cùng một công thức truy hồi như các hệ số b_k và tương ứng với hệ số b_{n-1}

$$b_{n-1} = a_{n-1} + sb_{n-2} - pb_{n-3} = \alpha$$

Hệ số b_n là :

$$b_n = a_n + sb_{n-1} - pb_{n-2} = sb_{n-1} + \beta$$

và cuối cùng :

$$R_1(x) = \alpha x + \beta = b_{n-1}(x - s) + b_n$$

Ngoài ra các hệ số b_i phụ thuộc vào s và p và bây giờ chúng ta cần phải tìm các giá trị đặc biệt s^* và p^* để cho b_{n-1} và b_n triệt tiêu.Khi đó $r_1(x) = 0$ và nghiệm của tam thức $x^2 - s^*x + p^*x$ sẽ là nghiệm của đa thức $P_n(x)$.Ta biết rằng b_{n-1} và b_n là hàm của s và p :

$$b_{n-1} = f(s,p)$$
$$b_n = g(s,p)$$

Việc tìm s^* và p^* đưa đến việc giải hệ phương trình phi tuyến:

$$\begin{cases} f(s,p)=0 \\ g(s,p)=0 \end{cases}$$

Phương trình này có thể giải dễ dàng nhờ phương pháp Newton.Thật vậy với một phương trình phi tuyến ta có công thức lặp :

$$x_{i+1} = x_i - f(x_i)/f'(x_i)$$

hay
$$f'(x_i)(x_{i+1} - x_i) = -f(x_i)$$

Với một hệ có hai phương trình,công thức lặp trở thành:

$$J(X_i)(X_{i+1} - X_i) = -F(X_i)$$

với $X_i = \{ s_i,p_i \}^T$ và $X_{i+1} = \{ s_{i+1},p_{i+1} \}^T$

$$F(X_i)=\begin{vmatrix} f(s_i,p_i) \\ g(s_i,p_i) \end{vmatrix}$$

$$J(X_i)=\begin{vmatrix} \frac{\partial f}{\partial s} & \frac{\partial f}{\partial p} \\ \frac{\partial g}{\partial s} & \frac{\partial g}{\partial p} \end{vmatrix}$$

Quan hệ : $J(X_i)\Delta X = -F(X_i)$ với $\Delta X = \{s_{i+1} - s_i, p_{i+1} - p_i\}^T$ tương ứng với một hệ phương trình tuyến tính hai ẩn số $\Delta s = s_{i+1} - s_i$ và $\Delta p = p_{i+1} - p_i$:

$$\begin{cases} \frac{\partial f}{\partial s}\Delta s + \frac{\partial f}{\partial p}\Delta p = -f(s_i, p_i) \\ \frac{\partial g}{\partial s}\Delta s + \frac{\partial g}{\partial p}\Delta p = -g(s_i, p_i) \end{cases}$$

Theo công thức Cramer ta có :

$$\Delta s = \frac{-f\frac{\partial g}{\partial p} + g\frac{\partial f}{\partial p}}{\delta}$$

$$\Delta p = \frac{-g\frac{\partial f}{\partial s} + f\frac{\partial g}{\partial s}}{\delta}$$

$$\delta = \frac{\partial f}{\partial s}\frac{\partial g}{\partial p} - \frac{\partial f}{\partial p}\frac{\partial g}{\partial s}$$

Để dùng được công thức này ta cần tính được các đạo hàm $\frac{\partial f}{\partial s}, \frac{\partial f}{\partial p}, \frac{\partial g}{\partial s}, \frac{\partial g}{\partial p}$.Các đạo hàm này được tính theo công thức truy hồi.

Do $b_o = a_o$ nên

$$\frac{\partial b_o}{\partial s} = 0 \qquad \frac{\partial b_o}{\partial p} = 0$$

$b_1 = a_1 + sb_o$ nên

$$\frac{\partial b_1}{\partial s} = b_o \qquad \frac{\partial b_1}{\partial p} = 0$$

$b_2 = a_2 + sb_1 - pb_o$ nên

$$\frac{\partial b_2}{\partial s} = \frac{\partial a_2}{\partial s} + \frac{\partial (sb_1)}{\partial s} - \frac{\partial (pb_o)}{\partial s}$$

Mặt khác :

$$\frac{\partial a_2}{\partial s} = 0 \qquad \frac{\partial (sb_1)}{\partial s} = s\frac{\partial b_1}{\partial s} + b_1 \qquad \frac{\partial (pb_o)}{\partial s} = 0$$

nên :

$$\frac{\partial b_2}{\partial s} = b_1 + sb_o$$

$b_3 = a_3 + sb_2 - pb_1$ nên

$$\frac{\partial b_3}{\partial s} = b_2 + s\frac{\partial b_2}{\partial s} - p\frac{\partial b_1}{\partial s}$$

Nếu chúng ta đặt :

$$\frac{\partial b_k}{\partial s} = c_{k-1}$$

thì :

$$c_o = b_o \qquad (2)$$
$$c_1 = b_1 + sb_o = b_1 + sc_o$$

$$c_2 = b_2 + sc_1 - pc_o$$
.....
$$c_k = b_k + sc_{k-1} - pc_{k-2}$$
$$c_{n-1} = b_{n-1} + sc_{n-2} - pc_{n-3}$$

Như vậy các hệ số cũng được tính theo cách như các hệ số b_k.Cuối cùng với $f = b_{n-1}$ và $g = b_n$ ta được:

$$\frac{\partial f}{\partial s} = c_{n-2} \quad \frac{\partial f}{\partial s} = c_{n-3} \quad \frac{\partial f}{\partial s} = c_{n-1} \quad \frac{\partial f}{\partial s} = c_{n-2}$$

$$\Delta s = \frac{b_{n-1}c_{n-2} - b_n c_{n-3}}{c_{n-1}c_{n-3} - c_{n-2}^2} \qquad (3)$$

$$\Delta p = \frac{b_{n-1}c_{n-1} - b_n c_{n-2}}{c_{n-1}c_{n-3} - c_{n-2}^2} \qquad (4)$$

Sau khi phân tích xong $P_n(x)$ ta tiếp tục phân tích $P_{n-2}(x)$ theo phương pháp trên Các bước tính toán gồm :

- Chọn các giá trị ban đầu bất kì s_0 và p_0
- Tính các giá trị $b_o,..,b_n$ theo (1)
- Tính các giá trị $c_o,...,c_n$ theo (2)
- Tính Δs_o và Δp_o theo (3) và (4)
- Tính $s_1 = s_0 + \Delta s_o$ và $p_1 = p_o + \Delta p_o$
- Lặp lại bước 1 cho đến khi $p_{i+1} = p_i = p$ và $s_{i+1} = s_i = s$
- Giải phương trình $x_2 - sx + p$ để tìm 2 nghiệm của đa thức
- Bắt đầu quá trình trên cho đa thức $P_{n-2}(x)$

Ví dụ : Tìm nghiệm của đa thức $P_4(x) = x^4 - 1.1x^3 + 2.3x^2 + 0.5x^2 + 3.3$.
Với lần lặp ban đầu ta chọn s = -1 và p =1,nghĩa là tam thức có dạng $x^2 + x + 1$

	a_0	a_1	a_2	a_3	a_4
	1	-1.1	2.3	0.5	3.3
sb_i		-1	2.1	-3.4	0.8
$-pb_{i-1}$			-1	2.1	-3.4
b_i	1	-2.1	3.4	$-0.8 = b_{n-1}$	$0.7 = b_n$
sb_i		-1.0	3.1	-5.5	
$-pb_{i-1}$			-1.0	3.1	
c_i	1	-3.1	5.5	-3.2	

$$\Delta s = \frac{\begin{vmatrix} 0.8 & -3.1 \\ -0.7 & 5.5 \end{vmatrix}}{\begin{vmatrix} 5.5 & -3.1 \\ -3.2 & 5.5 \end{vmatrix}} = 0.11$$

$$\Delta p = \frac{\begin{vmatrix} 5.5 & 0.8 \\ -3.2 & 0.7 \end{vmatrix}}{\begin{vmatrix} 5.5 & -3.1 \\ -3.2 & 5.5 \end{vmatrix}} = 0.06$$

$s^* = -1 + 0.11 = -0.89$
$p^* = 1 + 0.06 = 1.06$

Tiếp tục lặp lần 2 với $s_1 = s^*$ và $p_1 = p^*$ ta có :

	a_0	a_1	a_2	a_3	a_4
	1	-1.1	2.3	0.5	3.3
sb_i		-0.89	1.77	-2.68	0.06
$-pb_{i-1}$			-1.06	2.11	-3.17
b_i	1	-1.99	3.01	$-0.07 = b_{n-1}$	$0.17 = b_n$
sb_i		-0.89	2.56	-4.01	
$-pb_{i-1}$			-1.0	3.1	
c_i	1	-2.88	4.51	-1.03	

$$\Delta s = \frac{\begin{vmatrix} 0.07 & -2.88 \\ -0.7 & 5.5 \end{vmatrix}}{\begin{vmatrix} 4.51 & -2.88 \\ -1.03 & 4.51 \end{vmatrix}} = -0.01$$

$$\Delta p = \frac{\begin{vmatrix} 4.51 & 0.07 \\ -1.03 & -0.17 \end{vmatrix}}{\begin{vmatrix} 4.51 & -2.88 \\ -1.03 & 4.51 \end{vmatrix}} = 0.04$$

$s^* = -0.89 - 0.01 = -0.9$

$p^* = 1.06 + 0.04 = 1.1$

Như vậy $P_4(x) = (x^2+0.9x+1.1)(x^2+2x+3)$

Chương trình sau áp dụng lí thuyết vừa nêu để tìm nghiệm của đa thức.

Chương trình 8-10

```
//phuong phap Bairstow
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define  m   10

void main()
  {
        float a[m],b[m],c[m];
        int i,n,v;
        float s,e1,t,p,q,r,p1,q1;

        clrscr();
        printf("Cho bac cua da thuc n = ");
        scanf("%d",&n);
        printf("Cho cac he so cua da thuc can tim nghiem\n");
        for (i=n;i>=0;i--)
          {
                printf("a[%d] = ",n-i);
                scanf("%f",&a[i]);
          }
        printf("\n");
```

```
e1=0.0001;
if (n<=2)
    if (n==1)
        {
          printf("Nghiem cua he\n");
          printf("%.8f",(a[0]/(-a[1])));
          getch();
          exit(1);
        }
do
  {
        v=0;
        p=1;
        q=-1;
        b[n]=a[n];
        c[n]=a[n];
        do
          {
                b[n-1]=b[n]*p+a[n-1];
                c[n-1]=b[n-1]+b[n]*p;
                for (i=n-2;i>=0;i--)
                  {
                        b[i]=b[i+2]*q+b[i+1]*p+a[i];
                        c[i]=c[i+2]*q+c[i+1]*p+b[i];
                   }
                r=c[2]*c[2]-c[1]*c[3];
                p1=p-(b[1]*c[2]-b[0]*c[3])/r;
                q1=q-(b[0]*c[2]-b[1]*c[1])/r;
                if ((fabs(b[0])<e1)&&(fabs(b[1])<e1))
                  goto tt;
                v=v+1;
                p=p1;
                q=q1;
          }
        while (v<=40);
        if(v>40)
          {
                printf("Khong hoi tu sau 40 lan lap");
                getch();
                exit(1);
          }
        tt:s=p1/2;
        t=p1*p1+4*q1;
        if(t<0)
          {
                printf("Nghiem phuc\n");
                printf("%.8f+%.8fj\n",s,(sqrt(-t)/2));
                printf("%.8f-%.8fj\n",s,(sqrt(-t)/2));
                printf("\n");
          }
```

```
            else
              {
                    printf("Nghiem thuc\n");
                    printf("%.8f\n",(s+sqrt(t)/2));
                    printf("%.8f\n",(s-sqrt(t)/2));
                    printf("\n");
              }
            for (i=2;i<=n;i++)
              a[i-2]=b[i];
              n=n-2;
      }
    while ((n>2)&(r!=0.0));
    s=-a[1]/(2*a[2]);
    t=a[1]*a[1]-4*a[2]*a[0];
    if (t<0)
      {
            printf("Nghiem phuc\n");
            printf("%.8f+%.8fj\n",s,(sqrt(-t)/(2*a[2])));
            printf("%.8f-%.8fj\n",s,(sqrt(-t)/(2*a[2])));
            printf("\n");
      }
    else
      {
            printf("Nghiem thuc\n");
            printf("%.8f\n",(s-sqrt(t)/(2*a[2])));
            printf("%.8f\n",(s-sqrt(t)/(2*a[2])));
            printf("\n");
      }
    getch();
}
```

Dùng chương trình trên để xác định nghiệm của đa thức :

$$x^6 - 2x^5 - 4x^4 + 13x^3 - 24x^2 + 18x - 4 = 0$$

ta nhận được các nghiệm :

$x_1 = 2.61903399$

$x_2 = -2.73205081$

$x_3 = 0.732050755$

$x_4 = 0.381966055$

$x_5 = 0.500011056 + i*1.3228881$

$x_6 = 0.500011056 - i*1.3228881$

§11.HỆ PHƯƠNG TRÌNH PHI TUYẾN

Phương pháp Newton có thể được tổng quát hoá để giải hệ phương trình phi tuyến dạng :

$$\begin{cases} f_1(x_1,x_2,x_3,...,x_2)=0 \\ f_2(x_1,x_2,x_3,...,x_2)=0 \\ f_3(x_1,x_2,x_3,...,x_2)=0 \\ ..... \\ f_n(x_1,x_2,x_3,...,x_2)=0 \end{cases}$$

hay viết gọn hơn dưới dạng :

$$F(X) = 0$$

Trong đó : $X = (x_1,x_2,x_3,.....,x_n)$

Với một phương trình một biến,công thức Newton là :

$$x_{i+1}=x_i-\frac{f(x_i)}{f'(x_i)}$$

hay : $f'(x_i).\Delta x = -f(x_i)$

với $\Delta x = x_{i+1} - x_i$

Đối với hệ,công thức lặp là :

$$J(X_i)\Delta x = -F(X_i)$$

Trong đó $J(X_i)$ là toán tử Jacobi.Nó là một ma trận bậc n (n - tương ứng với số thành phần trong vectơ X) có dạng :

$$J(x_i)=\begin{vmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3}..... & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3}..... & \frac{\partial f_2}{\partial x_n} \\ . & & & \\ . & & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \frac{\partial f_n}{\partial x_3}..... & \frac{\partial f_n}{\partial x_n} \end{vmatrix}$$

và $\Delta X = X_{i+1} - X_i$

Phương pháp Newton tuyến tính hoá hệ và như vậy với mỗi bước lặp cần giải một hệ phương trình tuyến tính (mà biến là Δx_i) xác định bởi công thức lặp cho tới khi vectơ $X(x_1,x_2,x_3,.....,x_n)$ gần với nghiệm.

Dưới đây là chương trình giải hệ phương trình phi tuyến

$$\begin{cases} x_1^3-x_2^3-3x_1x_2x_4-8=0 \\ x_1+x_2+x_3+x_4-5=0 \\ \sqrt{25-x_1^2}+8x_3+4=0 \\ 2x_1x_2x_3-x_4+8=0 \end{cases}$$

Ma trận đạo hàm riêng $J(x_i)$là :

$$\begin{pmatrix} 3x_1^2-3x_2x_4 & -3x_2^2-3x_1x_4 & 0 & -3x_1x_2 \\ 1 & 1 & 1 & 1 \\ \frac{-x_1}{\sqrt{25-x_1^2}} & 0 & 8 & 0 \\ 2x_2x_3 & 2x_2x_3 & 2x_2x_3 & -1 \end{pmatrix}$$

Ma trận này được chương trình đọc vào nhờ thủ tục doc.Trong thủ tục này,các hệ số a[i,5] là các hàm $f_i(x)$.Vectơ nghiệm ban đầu được chọn là $\{ 0,-1,-1,1\}^T$.Kết quả tính cho ta : x = $\{0.01328676,-1.94647929,-1.12499779,8.05819031 \}^T$ với độ chính xác 0.000001.Vectơ số dư r = $\{ 0.00000536,-0.00000011,-0.00000001,-0.00000006\}^T$.

Chương trình 8-11

```
//giai he pt phi tuyen
#include <conio.h>
#include <stdio.h>
```

```
#include <math.h>
#include <stdlib.h>
#define n  4

float a[n+1][n+2];
float x[n+1],y[n+1];
int i,j,k,l,z,r;
float e,s,t;

void main()
 {
        void doc();
   clrscr();
        printf("Cho cac gia tri nghiem ban dau\n");
        for (i=1;i<=n;i++)
          {
                printf("x[%d] = ",i);
                scanf("%f",&x[i]);
          }
        e=1e-6;
        z=30;
        for (r=1;r<=z;r++)
          {
                doc();
                for (k=1;k<=n-1;k++)
                  {
                        s=0 ;
                        for (i=k;i<=n;i++)
                          {
                                t=fabs(a[i][k]);
                                if (s<=t)
                                  {
                                        s=t;
                                        l=i;
                                  }
                          }
                        for (j=k;j<=n+1;j++)
                          {
                                s=a[k][j];
                                a[k][j]=a[l][j];
                                a[l][j]=s;
                          }
                        if (a[1][1]==0)
                          {
                                printf("Cac phan tu duong cheo cua ma tran bang khong");
                                getch();
                                exit(1);
                          }
                        else
                          {
```

```
                        if (fabs(a[k][k]/a[1][1])<(1e-08))
                          {
                                printf("Ma tran suy bien");
                                goto mot;
                          }
                      }
                    for (i=k+1;i<=n;i++)
                      {
                        if (a[k][k]==0)
                          {
                                printf("Cac phan tu duong cheo cua ma tran bang khong\n");
                                goto mot;
                          }
                        s=a[i][k]/a[k][k];
                        a[i][k]=0;
                        for (j=k+1;j<=n+1;j++)
                         a[i][j]=a[i][j]-s*a[k][j];
                      }
                    y[n]=a[n][n+1]/a[n][n];
                    for (i=n-1;i>=1;i--)
                      {
                        s=a[i][n+1];
                        for (j=i+1;j<=n;j++)
                          s=s-a[i][j]*y[j];
                        if (a[i][i]==0)
                          {
                                printf("Cac phan tu duong cheo cua ma tran bang khong\n");
                                goto mot;
                          }
                        y[i]=s/a[i][i];
                      }
                  }
                if (r!=1)
                  for (i=1;i<=n;i++)
                        {
                          if (fabs(y[i])<e*fabs(x[i]))
                                goto ba;
                        }
                for (i=1;i<=n;i++)
                  x[i]=x[i]-y[i];
                printf("\n");
          }
        printf("Khong hoi tu sau %d lan lap\n",z);
        goto mot;
        clrscr();
        ba:printf("Vec to nghiem\n");
        for (i=1;i<=n;i++)
          printf("%.5f\n",(x[i]-y[i]));
```

```
        printf("\n");
        printf("Do chinh xac cua nghiem la %.5f: \n", e);
        printf("\n");
        printf("Vec to tri so du :\n");
        for (i=1;i<=n;i++)
          printf("%.5f\n",(a[i][n+1]));
        mot:printf("\n");
        getch();
  }

void doc()
  {
        a[1][1]=3*x[1]*x[1]-3*x[2]*x[4];
        a[1][2]=-3*x[2]*x[2]-3*x[1]*x[4];
        a[1][3]=0;
        a[1][4]=-3*x[1]*x[2];
        a[1][5]=x[1]*x[1]*x[1]-x[2]*x[2]*x[2]-3*x[1]*x[2]*x[4]-8;

        a[2][1]=1;
        a[2][2]=1;
        a[2][3]=1;
        a[2][4]=1;
        a[2][5]=x[1]+x[2]+x[3]+x[4]-5;

        a[3][1]=-x[1]/sqrt(25-x[1]*x[1]);
        a[3][2]=0;
        a[3][3]=8;
        a[3][4]=0;
        a[3][5]=sqrt(25-x[1]*x[1])+8*x[3]+4;

        a[4][1]=2*x[2]*x[3];
        a[4][2]=2*x[1]*x[3];
        a[4][3]=2*x[1]*x[2];
        a[4][4]=-1;
        a[4][5]=2*x[1]*x[2]*x[3]-x[4]+8;

  }
```