Métodos Numéricos para Engenharia

MÓDULO 4 - RAÍZES DE POLINÔMIOS

PROFESSOR LUCIANO NEVES DA FONSECA

Cálculo de raízes de polinômios

- Seja um polinômio $P_n(x)$ de ordem n:

$$P_n(x) = a_o + a_1 x + a_2 x^2 + a_3 x^3 + \cdots . + a_{n-1} x^{n-1} + a_n x^n$$

$$a_o, a_1 \ldots a_n \in R \quad ; \quad com\ a_n \neq 0$$

- $P_n(x)$ terá n raízes $z = x(k)$, com $1 \leq k \leq n$

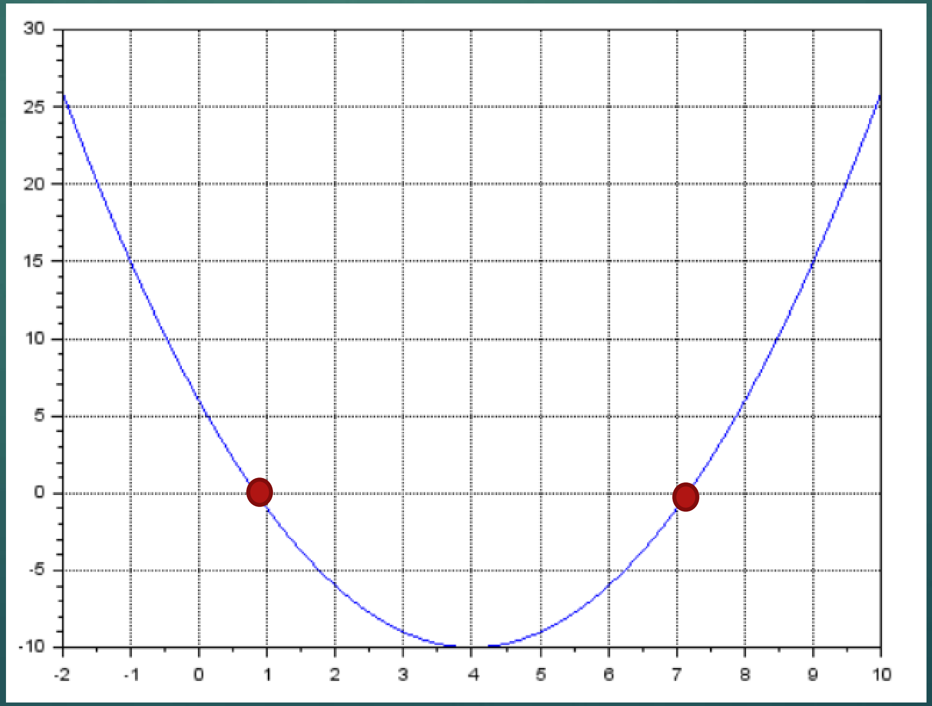
Podemos representar um polinômio como uma função

```
--> deff ('y=f(x)' , 'y=x^2-8*x+6')

--> x=linspace(-2,10,1000);

--> plot(x,f(x))

--> xgrid()
```



Ou podemos usar a representação especial (mais eficiente) de polinômios

```
--> ps=poly([6,-8,1],'s','coeff')
 ps  =

  6 -8s +s²

--> x=linspace(-2,10,1000);

--> plot(x,horner(ps,x))

--> xgrid()
```

Polinômio de Ordem 2: Fórmula de Báskara

$\Delta = a_1^2 - 4a_0a_2$

$P_2(x) = a_0 + a_1x + a_2x^2$

$$r_1 = \frac{-a_1 + \sqrt{\Delta}}{2a_2}$$

$$r_2 = \frac{-a_1 - \sqrt{\Delta}}{2a_2}$$

$$r_2 = \frac{c}{a\, r_1}$$

```
function [r1,r2,delta]=baskara(ps)
    u = coeff(ps);
    delta=u(2)^2-4*u(3)*u(1);
    r1=(-u(2) + sqrt(delta))/(2*u(3))
    r2=(-u(2) - sqrt(delta))/(2*u(3))
endfunction
```

$\Delta > 0$

```
--> ps=poly([6,5,1],"s","coeff")
  6 +5s +s²

--> [r1,r2,delta]=baskara(ps)
 r1  =

  -2.
 r2  =

  -3.
 delta  =

   1.
```

2 raízes reais distintas

$\Delta = 0$

```
--> ps=poly([1,-2,1],"s","coeff")
  1 -2s +s²

--> [r1,r2,delta]=baskara(ps)
 r1  =

   1.
 r2  =

   1.
 delta  =

   0.
```

2 raízes reais iguais

$\Delta < 0$

```
--> ps=poly([13,-4,1],"s","coeff")
 ps  =

  13 -4s +s²

--> [r1,r2,delta]=baskara(ps)
 r1  =

   2. + 3.i
 r2  =

   2. - 3.i
 delta  =

  -36.
```

2 raízes complexas conjugadas

Polinômio de Ordem 3 - Cardano

$$P_3(x) = a_0 + a_1 x + a_2 x^2 + x^3 \qquad mudança\ variável:\ y = x - \frac{a_2}{3}$$

$$P_3(y) = y^3 + \frac{Q}{3}y - \frac{R}{2} \quad onde:$$

$$Q = \frac{3a_1 - {a_2}^2}{9} \qquad R = \frac{9a_2 a_1 - 27a_0 - 2{a_2}^3}{54} \qquad \Delta = Q^3 + R^2$$

$$S_1 = \sqrt[3]{R + \sqrt{\Delta}}$$

$$S_2 = \sqrt[3]{R - \sqrt{\Delta}}$$

$$S_2 = \begin{cases} 0; & se\ Q = 0 \\ -\frac{Q}{S_1}; & cc \end{cases}$$

$$k = [1;\ 2; 3]$$

```
--> k=[1;2;3];
--> exp(%i*2*%pi*k/3)
 -0.5 + 0.8660254i
 -0.5 - 0.8660254i
  1.  + 0i
```

$$rts_k = -\frac{a_2}{3} + S_1 e^{\frac{j2\pi k}{3}} + S_2 e^{\frac{-j2\pi k}{3}}$$

```
function [rts,delta]=CardanoFormula(ps)
    u=coeff(ps)/coeff(ps)(4);
    Q=(3*u(2)-u(3)^2)/9
    R=(9*u(3)*u(2)-27*u(1)-2*u(3)^3)/54
    delta=Q^3+R^2
    S1=(R+sqrt(delta))^(1/3)
    if (Q==0)  then  S2=0;
    else           S2=-Q/S1     end
    k=[1;2;3]
    rts=clean(-u(3)/3+S1*exp(%i*2*%pi*k/3)...
                    +S2*exp(-%i*2*%pi*k/3))
endfunction
```

$\Delta = 0$

```
--> ps=s^3-3*s^2+3*s-1
 ps  =
  -1 +3s -3s² +s³

--> [rts,delta]=CardanoFormula(ps)
 rts  =
    1. + 0.i
    1. + 0.i
    1. + 0.i
 delta  =
    0.
```

3 raízes reais iguais

$\Delta = 0$

```
--> ps=s^3-60*s^2+900*s-4000
 ps  =
  -4000 +900s -60s² +s³

--> [rts,delta]=CardanoFormula(ps)
 rts  =
    10. + 0.i
    10. + 0.i
    40. + 0.i
 delta  =
    0.
```

2 raízes reais iguais, 1 distinta

$\Delta < 0$

```
--> ps=s^3-60*s^2+1100*s-6000
 ps  =
  -6000 +1100s -60s² +s³

--> [rts,delta]=CardanoFormula(ps)
 rts  =
    10. + 0.i
    20. + 0.i
    30. + 0.i
 delta  =
  -37037.03704
```

3 raízes reais distintas

$\Delta > 0$

```
--> ps=s^3-40*s^2+1125*s-14500
 ps  =
  -14500 +1125s -40s² +s³
--> [rts,delta]=CardanoFormula(ps)
 rts  =
    10. + 25.i
    10. - 25.i
    20. + 0.i
 delta  =
    12167245.37
```

1 raiz real e 2 complexas conjugadas

Polinômio de Ordem $n >= 4$ – Newton Raphson

$$ps5 = s^5 + 6s^4 - 48s^3 - 106s^2 + 687s - 540$$

```
--> ps5=s^5+6*s^4-48*s^3-106*s^2+687*s-540
 ps5  =

  -540 +687s -106s² -48s³ +6s⁴ +s⁵

--> xr=NewtonRaphson_pol(ps5, 0.1+%i*0.1)
 xr  =

   1. + 0.i
```

```
function x1=NewtonRaphson_pol(ps,x0,prt)
  erro = 1;
   if (prt)
       if(imag(x0)<>0)    printf ( '%i\t%.10f+%.10fi\t%.1e\n',1,real(x0),imag(x0),erro)
       else               printf ( '%i\t%.10f\t%.1e\n',1,x0,erro)  end
   end
  for (k=2:100)
     f0=horner(ps,x0)
     df0=horner(derivat_fga(ps),x0)
     x1=x0-f0/df0
     if (x1<>0) erro =abs((x1-x0)/x1) end
      if (prt)
         if(imag(x1)<>0)   printf ('%i\t%.10f+%.10fi\t%.1e\n',k,real(x1),imag(x1),erro)
          else             printf ( '%i\t%.10f\t%.1e\n',k,x0,erro)   end
      end
     if  ( (erro<1e-16) | (f(x1)==0) )  break  end
     x0=x1
  end
  x1=clean(x1,1e-10)
endfunction
```

Como $x_r = 1$ é uma raiz de $ps5$, podemos fatorar o termo $(s-1)$ do polinômio original $ps5$ de ordem 5, e obter um polinômio de ordem 4. Por $x_r = 1$ ser uma raiz, a divisão será exata, por consequência não haverá resto.

$$ps4 = \frac{ps5}{s-1} = 540 - 147s - 41s^2 + 7s^3 + s^4$$

Mas, como fazer esta divisão polinomial?

Divisão por um polinômio de 1ª ordem

$$ps5 = s^5 + 6s^4 - 48s^3 - 106s^2 + 687s - 540$$

$$\frac{ps5}{s-1} = ?$$

```
function [ps_out, resto, b]=fatorar_p1(ps,p_div)
    a=coeff(ps)
    d=coeff(p_div)
    n=length(a)
    b(n)=a(n);
    for k=n-1:-1:1
        b(k)=a(k )-d(1)*b(k+1);
    end
    resto = d(1)
    ps_out=poly(b(2:n),'s','coeff')
endfunction
```

$$s^5 + 6s^4 - 48s^3 - 106s^2 + 687s - 540 \quad | \quad s - 1$$

$$s^5 - s^4$$

$$s^4 + 7s^3 - 41s^2 - 147s + 540$$

$$7s^4 - 48s^3$$

$$7s^4 - 7s^3$$

$$-41s^3 - 106s^2$$

$$-41s^3 + 41s^2$$

$$-147s^2 + 687s$$

$$-147s^2 + 147s$$

$$540s - 540s$$

$$540s - 540s$$

$$0$$

resto

ps_out

b=[0; 540; -147;- 41; 7; 1]

```
--> ps5=s^5+6*s^4-48*s^3-106*s^2+687*s-540

  -540 +687s -106s² -48s³ +6s⁴ +s⁵

--> xr=NewtonRaphson_pol(ps5, 0.1+%i*0.1)

   1. + 0.i

--> ps4=fatorar_p1(ps5,(s-xr))

  540 -147s -41s² +7s³ +s⁴
```

Podemos retirar sucessivamente todos as raízes

$$ps5 = s^5 + 13s^4 + 30s^3 - 74s^2 - 255s - 675$$

```
function [r1,r2,delta]=baskara(ps)
    a = coeff(ps);
    delta=a(2)^2-4*a(3)*a(1);
    r1=(-a(2) + sqrt(delta))/(2*a(3))
    r2=(-a(2) - sqrt(delta))/(2*a(3))
endfunction
```

```
--> ps5=s^5+6*s^4-48*s^3-106*s^2+687*s-540
  -540 +687s -106s² -48s³ +6s⁴ +s⁵

--> raiz=NewtonRaphson_pol(ps5, 0.1+%i*0.1)
   1. + 0.i

--> ps4= fatorar_p1(ps5,(s-raiz) )
  540 -147s -41s² +7s³ +s⁴

--> raiz=NewtonRaphson_pol(ps4, 0.1+%i*0.1)
   3. + 0.i

--> ps3= fatorar_p1(ps4,(s-raiz) )
  -180 -11s +10s² +s³

--> raiz=NewtonRaphson_pol(ps3, 0.1+%i*0.1)
  -9. + 0.i

--> ps2=fatorar_p1(ps3,(s-raiz))
  -20 +s +s²

--> [r1,r2]=baskara(ps2)
 r1  =
   4.
 r2  =
  -5.
```

Raiz 1

$ps5 = (540 - 147s - 41s^2 + 7s^3 + s^4)(s-1)$

Raiz 3

$ps5 = (-180 - 11s + 10s^2 + s^3)(s-1)(s-3)$

Raiz -9

$ps5 = (-20 + s + s^2)(s-1)(s-3)(s+9)$

Raiz 4

Raiz -5

$ps5 = (s-1)(s-3)(s+9)(s-4)(s+5)$

$$ps5 = s^5 + 13s^4 + 30s^3 - 74s^2 - 255s - 675 = (s-1)\,(s-3)\,(s+9)\,(s-4)\,(s+5)$$

Divisão por um polinômio de 2ª ordem

$ps5 = s^5 + 13s^4 + 30s^3 - 74s^2 - 255s - 675$

```
--> ps5=-675-255*s-74*s^2+30*s^3+13*s^4+s^5
 ps5  =

  -675 -255s -74s² +30s³ +13s⁴ +s⁵

--> xr=NewtonRaphson_pol(ps5,complex(0.1,0.1))
 xr  =

   3. + 0.i

--> ps4 = fatorar_p1(ps5,(s-xr))
 ps4  =

  225 +160s +78s² +16s³ +s⁴

--> xr=NewtonRaphson_pol(ps4,complex(0.1,0.1))
 xr  =

  -5. + 0.i

--> ps3 = fatorar_p1(ps4,(s-xr))
 ps3  =

  45 +23s +11s² +s³

--> xr=NewtonRaphson_pol(ps3,complex(0.1,0.1))
 xr  =

  -1. + 2.i

--> ps1 = fatorar_p2(ps3,(s-xr)*(s-conj(xr)))
 ps1  =

  9 +s
```

```
function [ps_out, resto, b]=fatorar_p2(ps,p_div)
    a=coeff(ps)
    d=coeff(p_div)
    n=length(a)
    b(n)=a(n);
    b(n-1)=a(n-1)-d(2)*b(n);
    for k=n-2:-1:1
        b(k)=a(k)-b(k+1)*d(2)-b(k+2)*d(1);
    end
    resto = b(1)+b(2)*(s+d(2))
    ps_out=poly(b(3:n),'s','coeff')
endfunction
```

Raiz 3

$ps5 = (225 + 160s + 78s^2 + 16s^3 + s^4)(s-3)$

Raiz -5

$ps5 = (45 + 23s + 11s^2 + s^3)(s-3)(s+5)$

$$ps1 = \frac{ps3}{s^2+2s+5} = s+9$$

Raízes -1+j2 e -1-j2

$ps5 = (9+s)(s-3)(s+5)\,(s^2+2s+5)$

$s^2+2s+5 = (s+1+j2)(s+1-j2)$

$$\begin{array}{l|l} s^3 + 11s^2 + 23s + 45 & s^2 + 2s + 5 \\ \hline \underline{s^3 + 2s^2 + 5s} & s+9 \\ \quad 9s^2 + 18s - 45 & \\ \quad \underline{9s^2 + 18s^2 + 45} & \\ \qquad 0 & \end{array}$$

Raiz -9 fator

$ps5 = (s-3)(s+5)\,(s^2+2s+5)\,(s+9)$

)

$$ps5 = s^5 + 13s^4 + 30s^3 - 74s^2 - 255s - 675 = (s-3)\,(s+5)\,(s^2+2s+5)\,(s+9)$$

Polinômio de Ordem n - Método de Muller

- O método de Muller é um generalização deste procedimento, fatorando-se 1 raiz por vez, no caso de uma raiz real; ou fatorando-se 2 raízes simultaneamente no caso de um par complexo conjugado
- Podemos assim calcular e fatorar sucessivamente todos as raízes.

```
function rts=muller_iterativo(ps,prt)
    n=length(coeff(ps)) // polinônio ordem n-1
    while n>3  // repita enquanto ordem do polinômio for maior que 2
            raiz=NewtonRaphson_pol(ps,0.1+%i*0.1);
            rts(n-1)=clean(raiz,1e-8)
            if(abs(imag(rts(n-1))) < 1e-8 ) then
                 divisor =  s-rts(n-1);
                 if (prt) disp([ps, divisor, [rts(n-1)]]) end
                 ps = fatorar_p1(ps, divisor)    // ps=pdiv(ps,divisor)
                 n=n-1;   // diminua em 1 a ordem do polinômio
            else  // se a raiz for complexa elimine também o conjugado
                rts(n-2)=conj(rts(n-1))
                r=real(rts(n-1)),
                i=imag(rts(n-1))
                divisor = (s^2-2*r*s+r^2+i^2)
                if (prt) disp([ps, divisor, [rts(n-1) rts(n-2)]]) end
                ps = fatorar_p2(ps, divisor)  //  ps=pdiv(ps,divisor)
                n=n-2;   // diminua em 2 a ordem do polinômio
            end
    end;
    if(n==3)   // último polinômio de ordem 1 ou 2
        [rts(n-1),rts(n-2),delta]=baskara(ps);
        if (prt) disp([ps, [rts(n-1) rts(n-2)],  1]) end
    else
        a=coeff(ps)
        rts(n-1)=-a(1)/a(2)
        if (prt)  disp([ps, [rts(n-1)]]) end
    end;
    [g,k]=gsort(real(rts),'g','i')
    rts = clean(rts(k),1e-8)
endfunction
```

```
--> ps5=s^5+6*s^4-48*s^3-106*s^2+687*s-540
 ps5  =

  -540 +687s -106s^2 -48s^3 +6s^4 +s^5

--> rts=muller_iterativo(ps5,%t)

  -540 +687s -106s^2 -48s^3 +6s^4 +s^5   -1 +s   1

  540 -147s -41s^2 +7s^3 +s^4  -3 +s   3

  -180 -11s +10s^2 +s^3   9 +s   -9

  -20 +s +s^2   4   -5   1
 rts  =

  -9.
  -5.
   1.0000000
   3.0000000
   4.
```

```
--> ps5=s^5+13*s^4+30*s^3-74*s^2-255*s-675
 ps5  =

  -675 -255s -74s^2 +30s^3 +13s^4 +s^5

--> rts=muller_iterativo(ps5,%t)

  -675 -255s -74s^2 +30s^3 +13s^4 +s^5   -3 +s   3

  225 +160s +78s^2 +16s^3 +s^4   5 +s   -5

  45 +23s +11s^2 +s^3   5 +2s +s^2   -1+2i   -1-2i

  9 +s   -9
 rts  =

  -9. + 0.i
  -5. + 0.i
  -1. - 2.i
  -1. + 2.i
   3. + 0.i
```

Método do Fator Quadrático

Com o método de Newton-Raphson encontramos 1 raiz de um polinômio $P_n(x)$ por iteração.
Com o método do fator quadrático encontrar simultaneamente 2 raízes, através de um processo iterativo.

```
function [ps2_out,pdiv]=encontrar_divisor_quadratico(p_in)
    u= [0.1;0.1;1.0];//inicialização ps2_out= s^2+0.1*s+0.1
    du=[0.0;0.0;0.0]
    for (k=1:500)
        ps2_out=poly(u,'s','coeff')
        [pdiv, Rb, b] = fatorar_p2(p_in,ps2_out)
        pb_in=poly(b,'s','coeff')
        [pc   , Rc, c] = fatorar_p2(pb_in,ps2_out)
        du(2) =  ( b(1)*c(4)-c(3)*b(2) ) / (c(2)*c(4)-c(3)^2);
        du(1) =  ( c(2)*b(2)-b(1)*c(3) ) / (c(2)*c(4)-c(3)^2);
        if (norm(du)<1e-16) break end
        u=u+du
    end
endfunction
```

- O método consiste em se procurar iterativamente os coeficiente (u_0, u_1) do fator quadrático $P_{2\_out}(x) = u_0 + u_1\,x + x^2$ de $P_{in}(x)$.
- Se $P_{2\_out}(x)$ for uma fator quadrático exato, não haverá resto na divisão, isto é $\frac{P_{in}(x)}{P_{2\_out}(x)} = P_{div}(x)$, e as raízes de $P_{2\_out}(x)$ serão também raízes de $P_{in}(x)$.
- O detalhamento deste algoritmo é um conteúdo opcional mostrado no final deste módulo .

$$P_4(s) = -60 + 13s + 35s^2 + 11s^3 + s^4$$

```
--> ps4=-60+13*s+35*s^2+11*s^3+s^4;

--> [ps2_out,pdiv]=encontrar_divisor_quadratico(ps4)
 ps2_out  =

  -3 +2s +s²
 pdiv  =

  20 +9s +s²
```

```
--> [r1,r2]=baskara(p_out)
 r1  =

  -4.0000000
 r2  =

  -5.0000000

--> [r3,r4]=baskara(pq)
 r3  =

   1.0000000
 r4  =

  -3.0000000
```

```
--> [r1;r2;r3;r4]
 ans  =

   1.000000000
  -3.000000000
  -4.000000000
  -5.000000000
```

$$P_{div}(s) = \frac{P_4(s)}{P_{s2out}(s)} = \frac{-60 + 13s + 35s^2 + 11s^3 + s^4}{s^2 + 2s - 3} = s^2 + 9s + 20$$

$$P_4(s) = (s^2 + 9s + 20)(s^2 + 2s - 3)$$

Exemplo 2 - $ps6 = -675 - 930s - 329s^2 - 44s^3 + 43s^4 + 14s^5 + s^6$

```
--> ps6=s^6+14*s^5+43*s^4-44*s^3-329*s^2-930*s-675
 ps6  =

  -675 -930s -329s² -44s³ +43s⁴ +14s⁵ +s⁶

--> [ps2_out,ps4]=encontrar_divisor_quadratico(ps6)
 ps2_out  =

  5 +6s +s²
 ps4  =

  -135 -24s -10s² +8s³ +s⁴

--> [r1,r2]=baskara(ps2_out)
 r1  =

  -1.000000000
 r2  =

  -5.000000000
```

$ps6 = (-135 - 24s - 10s^2 + 8s^3 + s^4)(5 + 6s + s^2)$

```
--> [ps2_out,ps2]=encontrar_divisor_quadratico(ps4)
 ps2_out  =

  -27 +6s +s²
 ps2  =

  5 +2s +s²

--> [r3,r4]=baskara(ps2_out)
 r3  =

   3.000000000
 r4  =

  -9.
```

$ps6 = (-27 + 6s + s^2)(5 + 2s + s^2)(5 + 6s + s^2)$

```
--> [r5,r6]=baskara(ps2)
 r5  =

  -1. + 2.i
 r6  =

  -1. - 2.i
```

```
--> [r1;r2;r3;r4;r5;r6]
 ans  =

  -1. + 0.i
  -5. + 0.i
   3. + 0.i
  -9. + 0.i
  -1. + 2.i
  -1. - 2.i
```

$$ps9(x) = -184704 - 52784s + 119100s^2 - 21990s^3 - 5292s^4 + 3383s^5 - 500s^6 - 10s^7 - 4s^8 + s^9$$

```
--> ps9
 ps9  =

  -184704 -52784s +119100s² -21990s³ -5292s⁴ +3383s⁵
   -500s⁶ -10s⁷ -4s⁸ +s⁹

--> [ps2_out,ps7]=encontrar_divisor_quadratico(ps9)
 ps2_out  =

  -2 -s +s²
 ps7  =

  92352 -19784s -3482s² +2844s³ -517s⁴ -11s⁵ -3s⁶
   +s⁷

--> [r1,r2]=baskara(ps2_out)
 r1  =

   2.
 r2  =

  -1.
```

$$ps9 = (92352 - 19784s - 3482s^2 + 2844s^3 - 517s^4 - 11s^5 - 3s^6 + s^7)(-2 - s + s^2)$$

```
--> [ps2_out,ps5]=encontrar_divisor_quadratico(ps7)
 ps2_out  =

  -12 -s +s²
 ps5  =

  -7696 +2290s -542s² -s³ -2s⁴ +s⁵

--> [r3,r4]=baskara(ps2_out)
 r3  =

   4.
 r4  =

  -3.
```

$$ps9 = (-7696 + 2290s - 542s^2 - s^3 - 2s^4 + s^5)(-12 - s + s^2)(-2 - s + s^2)$$

```
--> [ps2_out,ps3]=encontrar_divisor_quadratico(ps5)
 ps2_out  =

  13 -4s +s²
 ps3  =

  -592 -6s +2s² +s³

--> [r5,r6]=baskara(ps2_out)
 r5  =

   2. + 3.i
 r6  =

   2. - 3.i
```

$$ps9 = (-592 - 6s + 2s^2 + s^3)(13 - 4s + s^2)(-12 - s + s^2)(-2 - s + s^2)$$

```
--> [ps2_out,ps1]=encontrar_divisor_quadratico(ps3)
 ps2_out  =

  74 +10s +s²
 ps1  =

  -8 +s

--> [r7,r8]=baskara(ps2_out)
 r7  =

  -5. + 7.i
 r8  =

  -5. - 7.i
```

$$ps9 = (-8 + s)(74 + 10s + s^2)(13 - 4s + s^2)(-12 - s + s^2)(-2 - s + s^2)$$

```
--> r9=8
```

```
--> [r1;r2;r3;r4;r5;r6;r7;r8;r9]
 ans  =

   2. + 0.i
  -1. + 0.i
   4. + 0.i
  -3. + 0.i
   2. + 3.i
   2. - 3.i
  -5. + 7.i
  -5. - 7.i
   8. + 0.i
```

Método de Bairstow

- Podemos automatizar o processo, retirando 2 raízes por iteração
- O algoritmo irá parar quando o quociente tiver ordem 2 ou de ordem 1 (n for para ou ímpar)

```
function rts=bairstow_zeros(ps,prt)
    n=length(coeff(ps))
    p_in=ps
    while n>3 //repita enquanto ordem do polinômio > 2
          [ps2_out,p_div]=encontrar_divisor_quadratico(p_in)
          [rts(n-1),rts(n-2),delta]=baskara(ps2_out);
          if (prt)
              disp(p_in)
              disp(ps2_out)
              disp(rts(n-2:n-1)')
              printf("\n")
          end
          p_in=p_div
          n=n-2;   // diminua em 2 a ordem do polinômio
    end;
    if (prt)  disp(p_in) end
    if(n==3)  // última equação ordem 2
        [rts(n-1),rts(n-2),delta]=baskara(p_in);
        if (prt) disp(rts(n-2:n-1)') end
    else     // última equação ordem 1
        a=coeff(p_in) //linear
        rts(n-1)=-a(1)/a(2)
        if (prt) disp(rts(n-1)) end
    end;
    [g,k]=gsort(real(rts),'g','i')
    rts = clean(rts(k),1e-8)
endfunction
```

$$P_4(s) = -60 + 13s + 35s^2 + 11s^3 + s^4$$

```
--> ps4=-60+13*s+35*s^2+11*s^3+s^4
 ps4  =

  -60 +13s +35s² +11s³ +s⁴

--> rts=bairstow_zeros(ps4,%t)

  -60 +13s +35s² +11s³ +s⁴

  -3 +2s +s²

  -3.   1.


  20 +9s +s²

  -5.  -4.
 rts  =

  -5.000000000
  -4.000000000
  -3.000000000
   1.000000000
```

$$P_4(s) = (20 + 9s + s^2)(-3 + 2s + s^2)$$

$$ps6 = -675 - 930s - 329s^2 - 44s^3 + 43s^4 + 14s^5 + s^6$$

```
--> ps6=s^6+14*s^5+43*s^4-44*s^3-329*s^2-930*s-675
 ps6  =

  -675 -930s -329s² -44s³ +43s⁴ +14s⁵ +s⁶

--> rts=bairstow_zeros(ps6,%t)

  -675 -930s -329s² -44s³ +43s⁴ +14s⁵ +s⁶

  5 +6s +s²

  -5.   -1.


  -135 -24s -10s² +8s³ +s⁴

  -27 +6s +s²

  -9.    3.


  5 +2s +s²

  -1. + 2.i   -1. - 2.i
 rts  =

  -9. + 0.i
  -5. + 0.i
  -1. - 2.i
  -1. + 2.i
  -1. + 0.i
   3. + 0.i
```

$$ps6 = (-135 - 24s - 10s^2 + 8s^3 + s^4)\,(5 + 6s + s^2)$$

$$ps6 = (5 + 2s + s^2)\,(-27 + 6s + s^2)\,(5 + 6s + s^2)$$

$$ps9(x) = -184704 - 52784s + 119100s^2 - 21990s^3 - 5292s^4 + 3383s^5 - 500s^6 - 10s^7 - 4s^8 + s^9$$

```
--> ps9
 ps9  =

  -184704 -52784s +119100s² -21990s³ -5292s⁴ +3383s⁵ -500s⁶ -10s⁷ -4s⁸ +s⁹

--> rts=bairstow_zeros(ps9,%t)

  -184704 -52784s +119100s² -21990s³ -5292s⁴ +3383s⁵ -500s⁶ -10s⁷ -4s⁸ +s⁹

  -2 -s +s²

  -1.   2.

  92352 -19784s -3482s² +2844s³ -517s⁴ -11s⁵ -3s⁶ +s⁷

  -12 -s +s²

  -3.   4.

  -7696 +2290s -542s² -s³ -2s⁴ +s⁵

  13 -4s +s²

   2. + 3.i    2. - 3.i

  -592 -6s +2s² +s³

  74 +10s +s²

  -5. + 7.i   -5. - 7.i

  -8 +s

   8.
 rts  =

  -5. - 7.i
  -5. + 7.i
  -3. + 0.i
  -1. + 0.i
   2. - 3.i
   2. + 3.i
   2. + 0.i
   4. + 0.i
   8. + 0.i
```

$$ps9 = (92352 - 19784s - 3482s^2 + 2844s^3 - 517s^4 - 11s^5 - 3s^6 + s^7)(-2 - s + s^2)$$

$$ps9 = (-7696 + 2290s - 542s^2 - s^3 - 2s^4 + s^5)(-12 - s + s^2)(-2 - s + s^2)$$

$$ps9 = (-592 - 6s + 2s^2 + s^3)(13 - 4s + s^2)(-12 - s + s^2)(-2 - s + s^2)$$

$$ps9 = (-8 + s)(74 + 10s + s^2)(13 - 4s + s^2)(-12 - s + s^2)(-2 - s + s^2)$$

Conteúdo Opcional

Algoritmo do Método do Fator Quadrático (conteúdo opcional)

Este método consiste em se procurar os coeficiente (u_0, u_1) do fator quadrático $P_{2\_out}(x) = u_0 + u_1\, x + \, x^2$, que seja um divisor exato de $P_n(x)$. Se $P_{2\_out}(x)$ for uma fator exato, não haverá resto na divisão, isto é $\frac{P_n(x)}{P_{2\_out}(x)} = P_q(x)$, e as raízes de $P_{2\_out}(x)$ serão também raízes de $P_n(x)$.

$$P_n(x) = a_o + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{n-1} x^{n-1} + a_n x^n$$

$$P_{2\_out}(x) = u_0 + u_1\, x + \, x^2$$

$$P_q(x) = \, b_2 + b_3 x + b_4 x^2 + b_5 x^3 + \cdots + b_{n-1} x^{n-3} + b_n x^{n-2}$$

```
function [ps2_out,pq]=encontrar_divisor_quadratico(p_in)
    u= [0.1;0.1;1.0];//inicialização ps2_out= s^2+0.1*s+0.1
    du=[0.0;0.0;0.0]
    for (k=1:500)
       ps2_out=poly(u,'s','coeff')
       [pq, Rb, b] = fatorar_p2(p_in,ps2_out)
       pb_in=poly(b,'s','coeff')
       [pc    , Rc, c] = fatorar_p2(pb_in,ps2_out)
       du(2) =  ( b(1)*c(4)-c(3)*b(2) ) / (c(2)*c(4)-c(3)^2);
       du(1) =  ( c(2)*b(2)-b(1)*c(3) ) / (c(2)*c(4)-c(3)^2);
       if (norm(du)<1e-16) break end
       u=u+du
    end
endfunction
```

```
function [ps2_out,pq]=encontrar_divisor_quadratico(p_in)
    u= [0.1;0.1;1.0];//inicialização ps2_out= s^2+0.1*s+0.1
    du=[0.0;0.0;0.0]
    for (k=1:500)
        ps2_out=poly(u,'s','coeff')
        [pq, Rb, b] = fatorar_p2(p_in,ps2_out)
        pb_in=poly(b,'s','coeff')
        [pc   , Rc, c] = fatorar_p2(pb_in,ps2_out)
        du(2)  =  ( b(1)*c(4)-c(3)*b(2) ) / (c(2)*c(4)-c(3)^2);
        du(1)  =  ( c(2)*b(2)-b(1)*c(3) ) / (c(2)*c(4)-c(3)^2);
        if (norm(du)<1e-16) break end
        u=u+du
    end
endfunction
```

- Começamos escolhendo arbitrariamente os coeficientes $u = \begin{bmatrix} u_o \\ u_1 \end{bmatrix}$ de um polinômio $P_{2\_out}(x)$.
- A divisão de $P_n(x)$ por $P_{2\_out}(x)$ gera um novo polinômio $P_q(x)$ e um resto $R(x)$

$$\frac{P_n(x)}{P_{2\_out}(x)} = P_q(x) + \frac{R(x)}{P_{2\_out}(x)}$$

$$P_q(x) = b_2 + b_3 x + b_4 x^2 + b_5 x^3 + \cdots + b_{n-1} x^{n-3} + b_n x^{n-2}$$

$$R(x) = b_o + b_1 (x + u_1)$$

- O resto $R(x)$ depende da escolha dos coeficientes $u = \begin{bmatrix} u_o \\ u_1 \end{bmatrix}$ de $P_{2\_out}(x) = u_0 + u_1 x + x^2$.
- Se $P_{2\_out}(x)$ for um fator exato, o resto será nulo $R(x) = 0$
- Os coeficientes $(b_2 \; a \; b_n)$ de $P_q(x)$ e $(b_o \; e \; b_1)$ do resto $R(x)$, podem ser obtidos por fatoração

- Os coeficientes $(b_2\ a\ b_n)$ de $P_q(x)$ e $(b_o\ e\ b_1)$ do resto $R(x)$, podem ser obtidos pelo programa fatorar_p2.

```
function [ps_out, resto, b]=fatorar_p2(ps,p_div)
    a=coeff(ps)
    d=coeff(p_div)
    n=length(a)
    b(n)=a(n);
    b(n-1)=a(n-1)-d(2)*b(n);
    for k=n-2:-1:1
        b(k)=a(k)-b(k+1)*d(2)-b(k+2)*d(1);
    end
    resto = b(1)+b(2)*(s+d(2))
    ps_out=poly(b(3:n),'s','coeff')
endfunction
```

$$P_q(x) = b_2 + b_3x + b_4x^2 + b_5x^3 + \cdots + b_{n-1}x^{n-3} + b_nx^{n-2}$$

$$R(x) = b_o + b_1(x + u_1)$$

```
--> [Pq, R , b] = fatorar_p2(Pn , P2out)
```

Como a divisão não é exata, então: $P_{(n)}(x) = P_q(x) * P_{2\_out}(x) + R(x)$

- Devemos ajustar os coeficientes $u = \begin{bmatrix} u_o \\ u_1 \end{bmatrix}$, pela soma somando um valor $\Delta u = \begin{bmatrix} \Delta u_o \\ \Delta u_1 \end{bmatrix}$, até que o resto $R(x) = b_0 + b_1(x + u_1)$ se anule. Para encontramos Δu, precisamos das derivadas parciais de $b_o\ e\ b_1$ (coeficientes do resto) em relação a $u = \begin{bmatrix} u_o \\ u_1 \end{bmatrix}$. Da expansão de Taylor teremos:

$$b_0(u + \Delta u) = b_o(u) + \frac{\partial b_o}{\partial u_0}\Delta u_0 + \frac{\partial b_0}{\partial u_1}\Delta u_1$$

$$b_1(u + \Delta u) = b_1(u) + \frac{\partial b_1}{\partial u_0}\Delta u_0 + \frac{\partial b_1}{\partial u_1}\Delta u_1$$

- Se $R(x) = 0$, então $b_0(u + \Delta u) = b_1(u + \Delta u) = 0$

$$\begin{cases} \dfrac{\partial b_o}{\partial u_0}\Delta u_0 + \dfrac{\partial b_0}{\partial u_1}\Delta u_1 = -b_0 \\ \dfrac{\partial b_1}{\partial u_0}\Delta u_0 + \dfrac{\partial b_1}{\partial u_1}\Delta u_1 = -b_1 \end{cases}$$

- Baristow demonstrou que as derivadas parciais podem ser obtidas pelo mesmo programa fatorar_p2., substituindo o polinômio $P_n(x)$ pelo polinômio $P_{nb}(x)$ formado pelos coeficientes b.

```
function [ps_out, resto, b]=fatorar_p2(ps,p_div)
    a=coeff(ps)
    d=coeff(p_div)
    n=length(a)
    b(n)=a(n);
    b(n-1)=a(n-1)-d(2)*b(n);
    for k=n-2:-1:1
        b(k)=a(k)-b(k+1)*d(2)-b(k+2)*d(1);
    end
    resto = b(1)+b(2)*(s+d(2))
    ps_out=poly(b(3:n),'s','coeff')
endfunction
```

```
--> Pnb = poly(b,'s','coeff');

--> [Pc, Rc , c] = fatorar_p2(Pnb , P2out)
```

$$\boxed{\begin{matrix} \dfrac{\partial b_o}{\partial u} = -c_1 & \dfrac{\partial b_o}{dv} = -c_2 \\ \dfrac{\partial b_1}{\partial u} = -c_2 & \dfrac{\partial b_1}{dv} = -c_3 \end{matrix}}$$

$$\begin{cases} \dfrac{\partial b_o}{\partial u_0}\Delta u_0 + \dfrac{\partial b_0}{\partial u_1}\Delta u_1 = -b_0 \\ \dfrac{\partial b_1}{\partial u_0}\Delta u_0 + \dfrac{\partial b_1}{\partial u_1}\Delta u_1 = -b_1 \end{cases}$$

- Resolvendo este sistema, encontramos $\Delta u_0 \; e \; \Delta u_1$ par a próxima aproximação de $P_{2\_out}(x) = u_0 + u_1\, x + \, x^2$
- Este processo deve ser repetido até que $R(x)$ seja nulo

$$\begin{cases} c_1\Delta u_o + c_2\Delta u_1 = b_0 \\ c_2\Delta u_o + c_3\Delta u_1 = b_1 \end{cases}$$

Resolvendo o sistema temos:

$$\Delta u = \begin{bmatrix} \Delta u_0 \\ \Delta u_1 \end{bmatrix} = \begin{bmatrix} \dfrac{b_o c_3 - c_2 b_1}{c_1 c_3 - c_2^2} \\ \dfrac{c_1 b_1 - b_o c_2}{c_1 c_3 - c_2^2} \end{bmatrix}$$

$$\mathrm{u_{novo}} = \mathrm{u} + \Delta u = \begin{bmatrix} u_0 + \Delta u_o \\ u_1 + \Delta u_1 \end{bmatrix}$$

- Com o novo u_{novo} teremos o novo polinômio $P_{2\_out}(n)$.
- Repetimos o processo até que $R(x)$ se anule.
- Quando isso acontecer, $P_{2\_out}(n)$ será um fator exato de $P_n(x)$, de modo que as raízes de $P_{2\_out}(n)$ serão também raízes de $P_n(x)$.
- As raízes de $P_{2\_out}(n)$ são facilmente calculadas pela fórmula de Báskara.

```
--> ps4=-60+13*s+35*s^2+11*s^3+s^4;

--> p_in=ps4
 p_in  =

  -60 +13s +35s² +11s³ +s⁴

--> u=[0.1;0.1];
```

Exemplo: Encontre Primeiro fator quadrático de $ps4(s) = -60 + 13s + 35s^2 + 11s^3 + s^4$

1

```
--> pq=u(2)+u(1)*s + s^2
 pq  =

  0.1 +0.1s +s²

--> [p_out, Rb, b] = fatorar_p2(ps4,pq)
 p_out  =

  33.81 +10.9s +s²
 Rb  =

  -63.381 +8.529s
 b  =

  -64.233900
   8.529
   33.810000
   10.9
   1.

--> pb_in=poly(b,'s','coeff')
 pb_in  =

  -64.2339 +8.529s +33.81s² +10.9s³ +s⁴

--> [pc    , Rc, c] = fatorar_p2(pb_in,pq);

--> du(1)  =  ( b(1)*c(4)-c(3)*b(2) ) / (c(2)*c(4)-c(3)^2);

--> du(2)  =  ( c(2)*b(2)-b(1)*c(3) ) / (c(2)*c(4)-c(3)^2);

--> u=u+du
 u  =

   1.0534278
  -1.9908657
```

2

```
--> pq=u(2)+u(1)*s + s^2
 pq  =

  -1.9908657 +1.0534278s +s²

--> [p_out, Rb, b] = fatorar_p2(ps4,pq)
 p_out  =

  26.51287 +9.9465722s +s²
 Rb  =

  -7.2164359 +4.8728949s
 b  =

  -12.349679
   4.8728949
   26.512870
   9.9465722
   1.

--> pb_in=poly(b,'s','coeff')
 pb_in  =

  -12.349679 +4.8728949s +26.51287s² +9.9465722s³ +s⁴

--> [pc    , Rc, c] = fatorar_p2(pb_in,pq);

--> du(1)  =  ( b(1)*c(4)-c(3)*b(2) ) / (c(2)*c(4)-c(3)^2);

--> du(2)  =  ( c(2)*b(2)-b(1)*c(3) ) / (c(2)*c(4)-c(3)^2);

--> u=u+du
 u  =

   1.6426538
  -2.7107697
```

3

```
--> pq=u(2)+u(1)*s + s^2
 pq  =

  -2.7107697 +1.6426538s +s²

--> [p_out, Rb, b] = fatorar_p2(ps4,pq)
 p_out  =

  22.339889 +9.3573462s +s²
 Rb  =

  0.5582944 +1.6689059s
 b  =

  -2.1831402
   1.6689059
   22.339889
   9.3573462
   1.

--> pb_in=poly(b,'s','coeff')
 pb_in  =

  -2.1831402 +1.6689059s +22.339889s² +9.3573462s³ +s⁴

--> [pc    , Rc, c] = fatorar_p2(pb_in,pq);

--> du(1)  =  ( b(1)*c(4)-c(3)*b(2) ) / (c(2)*c(4)-c(3)^2);

--> du(2)  =  ( c(2)*b(2)-b(1)*c(3) ) / (c(2)*c(4)-c(3)^2);

--> u=u+du
 u  =

   1.9186564
  -2.9372828
```

Exemplo: Calcule as 4 raízes de $P_4(x) = -60 + 13x + 35x^2 + 11x^3 + x^4$

3

```
--> pq=u(2)+u(1)*s + s^2
 pq  =

  -2.7107697 +1.6426538s +s²

--> [p_out, Rb, b] = fatorar_p2(ps4,pq)
 p_out  =

  22.339889 +9.3573462s +s²
 Rb  =

  0.5582944 +1.6689059s
 b  =

  -2.1831402
   1.6689059
   22.339889
   9.3573462
   1.

--> pb_in=poly(b,'s','coeff')
 pb_in  =

  -2.1831402 +1.6689059s +22.339889s² +9.3573462s³ +s⁴

--> [pc    , Rc, c] = fatorar_p2(pb_in,pq);

--> du(1)  =   ( b(1)*c(4)-c(3)*b(2)  )  /  (c(2)*c(4)-c(3)^2);

--> du(2)  =   ( c(2)*b(2)-b(1)*c(3)  )  /  (c(2)*c(4)-c(3)^2);

--> u=u+du
 u  =

   1.9186564
  -2.9372828
```

4

```
--> pq=u(2)+u(1)*s + s^2
 pq  =

  -2.9372828 +1.9186564s +s²

--> [p_out, Rb, b] = fatorar_p2(ps4,pq)
 p_out  =

  20.513305 +9.0813436s +s²
 Rb  =

  0.2533765 +0.3164906s
 b  =

  -0.3538603
   0.3164906
   20.513305
   9.0813436
   1.

--> pb_in=poly(b,'s','coeff')
 pb_in  =

  -0.3538603 +0.3164906s +20.513305s² +9.0813436s³ +s⁴

--> [pc    , Rc, c] = fatorar_p2(pb_in,pq);

--> du(1)  =   ( b(1)*c(4)-c(3)*b(2)  )  /  (c(2)*c(4)-c(3)^2);

--> du(2)  =   ( c(2)*b(2)-b(1)*c(3)  )  /  (c(2)*c(4)-c(3)^2);

--> u=u+du
 u  =

   1.9937238
  -2.9948384
```

5

```
--> pq=u(2)+u(1)*s + s^2
 pq  =

  -2.9948384 +1.9937238s +s²

--> [p_out, Rb, b] = fatorar_p2(ps4,pq)
 p_out  =

  20.038811 +9.0062762s +s²
 Rb  =

  0.0130003 +0.0204866s
 b  =

  -0.0278444
   0.0204866
   20.038811
   9.0062762
   1.

--> pb_in=poly(b,'s','coeff')
 pb_in  =

  -0.0278444 +0.0204866s +20.038811s² +9.0062762s³ +s⁴

--> [pc    , Rc, c] = fatorar_p2(pb_in,pq);

--> du(1)  =   ( b(1)*c(4)-c(3)*b(2)  )  /  (c(2)*c(4)-c(3)^2);

--> du(2)  =   ( c(2)*b(2)-b(1)*c(3)  )  /  (c(2)*c(4)-c(3)^2);

--> u=u+du
 u  =

   1.9999553
  -2.9999612
```

Exemplo: Calcule as 4 raízes de $P_4(x) = -60 + 13x + 35x^2 + 11x^3 + x^4$

5

```
--> pq=u(2)+u(1)*s + s^2
 pq  =

  -2.9948384 +1.9937238s +s²

--> [p_out, Rb, b] = fatorar_p2(ps4,pq)
 p_out  =

  20.038811 +9.0062762s +s²
 Rb  =

  0.0130003 +0.0204866s
 b  =

  -0.0278444
   0.0204866
   20.038811
   9.0062762
   1.

--> pb_in=poly(b,'s','coeff')
 pb_in  =

  -0.0278444 +0.0204866s +20.038811s² +9.0062762s³ +s⁴

--> [pc   , Rc, c] = fatorar_p2(pb_in,pq);

--> du(1) =  ( b(1)*c(4)-c(3)*b(2) ) / (c(2)*c(4)-c(3)^2);

--> du(2) =  ( c(2)*b(2)-b(1)*c(3) ) / (c(2)*c(4)-c(3)^2);

--> u=u+du
 u  =

   1.9999553
  -2.9999612
```

6

```
--> pq=u(2)+u(1)*s + s^2
 pq  =

  -2.9999612 +1.9999553s +s²

--> [p_out, Rb, b] = fatorar_p2(ps4,pq)
 p_out  =

  20.000274 +9.0000447s +s²
 Rb  =

  0.0000465 +0.0001308s
 b  =

  -0.0002151
   0.0001308
   20.000274
   9.0000447
   1.

--> pb_in=poly(b,'s','coeff')
 pb_in  =

  -0.0002151 +0.0001308s +20.000274s² +9.0000447s³ +s⁴

--> [pc   , Rc, c] = fatorar_p2(pb_in,pq);

--> du(1) =  ( b(1)*c(4)-c(3)*b(2) ) / (c(2)*c(4)-c(3)^2);

--> du(2) =  ( c(2)*b(2)-b(1)*c(3) ) / (c(2)*c(4)-c(3)^2);

--> u=u+du
 u  =

   2.0000000
  -3.0000000
```

7

```
--> pq=u(2)+u(1)*s + s^2
 pq  =

  -3 +2s +s²

--> [p_out, Rb, b] = fatorar_p2(ps4,pq)
 p_out  =

  20 +9s +s²
 Rb  =

  1.208D-09 +6.524D-09s
 b  =

  -1.184D-08
   6.524D-09
   20.000000
   9.0000000
   1.
```

$$ps4 = p_{out}\, p_q = (-3 + 2s + s^2)(20 + 9s + s^2)$$

```
--> [r1,r2]=baskara(p_out)
 r1  =

  -4.0000000
 r2  =

  -5.0000000

--> [r3,r4]=baskara(pq)
 r3  =

   1.0000000
 r4  =

  -3.0000000
```

Exemplo: Calcule as 4 raízes de $P_4(x)$

$$P_4(s) = -60 + 13s + 35s^2 + 11s^3 + s^4$$

$$P_{out}(s) = 0.1 + 0.1s + s^2$$

$u_0 = 0.1$	1.0534278	1.6426538	1.9186564	1.9937238	1.9999553	2.
$u_1 = 0.1$	-1.9908657	-2.7107697	-2.9372828	-2.9948384	-2.9999612	-3.

$$P_{out}(s) = -3 + 2s + s^2$$

$$P_{div}(s) = \frac{P_4(s)}{P_{out}(s)} = 20 + 9s + s^2$$

$$P_4(s) = P_{out}(s)\, P_{div}(s) = (-3 + 2s + s^2)(20 + 9s + s^2)$$

Raízes

$$z_4 = 1.$$
$$z_3 = -3.$$
$$z_2 = -4.$$
$$z_1 = -5.$$

```
--> [r1,r2]=baskara(p_out)
 r1  =

  -4.0000000
 r2  =

  -5.0000000

--> [r3,r4]=baskara(pq)
 r3  =

   1.0000000
 r4  =

  -3.0000000
```