

Raízes de Equações de Grau Superior

Mário Leite

...

Um dos assuntos mais interessantes na Matemática estudado no Ensino Médio, é Álgebra; em especial as equações, embora fique um pouco frustrante quando se trata de equações de grau superior a 2. A resolução de uma equações do 1º Grau do tipo $ax + b = 0$ ($a \neq 0$) é muito fácil: basta fazer $-b/a$ e pronto: o valor da incógnita x é mostrado imediatamente. Para equações do 2º Grau, embora um pouco mais difícil basta aplicar a fórmula de Báskara para obter as duas raízes: $x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ e $x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$, tendo o cuidado de colocar o produto $2a$ entre parêntese, pois senão, caso o valor da constante a seja diferente de 1 os resultados sairão incorretos.

Para equações de grau 3 ainda tem a fórmula de "Cardano", mas que de tanto complicada ele é muitos desistem e partem para soluções mais complexas. Assim, nesta postagem apresentamos o programa "RaizesDePolinomio" (codificado em Visualg) que resolve o problema para equações a partir do grau 2 até o grau 10, usando o "Método de Birge-Vieta". Este programa é modular: com três sub-rotinas do tipo *procedure* ("função sem retorno" em outras linguagens), além do programa principal.

As figuras 1a, 1b e 1c mostram o programa em ação: lendo os dados da equação e mostrando as raízes de uma equação do 5º Grau.

Algoritmo "RaizesDePolinomio"

```
{
-----
{Programa modular para calcular as raízes reais de um polinômio
completo de grau n pelo "Método de Birge-Vieta"
Autor: Mário Leite
Data: 11/06/2023
-----
Xo: valor inicial de raiz dada pelo usuário.
Xi: um valor de raiz dentro do loop de refinamento.
Erro: erro cometido no refinamento das raízes.
tol: tolerância permitida no erro cometido ao calcular a raiz.
Ite: número de iterações para verificar se há convergência.
Iter: número total de iterações para detectar as raízes finais.
n: grau do polinômio.
NR: número de raízes do polinômio
Coef0[i], i=0,n: coeficientes originais do polinômio (lidos pelo teclado).
Coef1[i], i=1,3: coeficiente derivativos auxiliares para os cálculos.
VetRaizes: vetor que armazena as raízes do polinômio.
-----
}
//Elementos globais
Const MaxIte=1000 //define o máximo de iterações para detectar possíveis raízes
Var j, n, Iter: inteiro
    Coef0, Coef1, Coef2, Coef3: vetor[0..10] de real
    VetRaizes: vetor[1..10] de real //limita o grau do polinômio
    a, b, c, Xo, Xi, Y, tol, Erro: real
    x1R, x1C, x2R, x2C, Delta, Delta2: real
    nada, x1S, x2S: caractere
    Convergiu: logico
//-----
```

```

Procedimento LePolinomio (n:inteiro)
//Lê os dados básicos do polinômio.
    var k:inteiro
Inicio
    Escreval("")
    {Entradas dos coeficientes do polinômio}
    Para k De n Ate 0 Passo -1 Faca
        Se (k>0) Entao
            Escreva("Digite o coeficiente do termo de potência", k, ": ")
        Senao
            Escreva("Digite o valor do termo independente: ")
        FimSe
        Leia(Coef1[k])
        Coef0[k] <- Coef1[k]
    FimPara
FimProcedimento //fim do procedimento "LePolinomio"
//-----

Procedimento MontaPolinomio (n:inteiro)
//Monta e exhibe o polinômio.
    var k: inteiro
Inicio
    Para k De n Ate 0 Passo -1 Faca
        Se (k<>0) Entao
            Se ((Coef0[k]>0) e (k=n)) Entao
                Se (Coef0[k]=1) Entao
                    Escreva("x^", NumpCarac(k), " ")
                Senao
                    Escreva(NumpCarac(Coef0[k]), "x^", NumpCarac(k), " ")
                FimSe
            FimSe
            Se ((Coef0[k]<0) e (k=n)) Entao
                Se (Coef0[k]=-1) Entao
                    Escreva("-x^", NumpCarac(k), " ")
                Senao
                    Escreva(NumpCarac(Coef0[k]), "x^", NumpCarac(k), " ")
                FimSe
            FimSe
            Se ((Coef0[k]>0) e (k<>n) e (k>1)) Entao
                Escreva("+", NumpCarac(Coef0[k]), "x^", NumpCarac(k), " ")
            FimSe
            Se ((Coef0[k]>0) e (k<>n) e (k=1)) Entao
                Escreva("+", NumpCarac(Coef0[k]), "x ")
            FimSe
            Se ((Coef0[k]<0) e (k<>n) e (k=1)) Entao
                Escreva(NumpCarac(Coef0[k]), "x ")
            FimSe
            Se ((Coef0[k]<0) e (k<>n) e (k<>1)) Entao
                Escreva("-", NumpCarac(Coef0[k]), "x^", NumpCarac(k), " ")
            FimSe
        Senao
            Se (Coef0[0]>0) Entao
                Escreva("+", Coef0[0])
            FimSe
            Se (Coef0[0]<0) Entao
                Escreva(Coef0[0])
            FimSe
        FimSe
    FimPara
FimProcedimento //fim do procedimento "MontaPolinomio"
//-----

```

Procedimento **ProRaiz2G**(a,b,c:real)
//Calcula as raízes do polinômio de grau 2.

Início

```
Delta <- b^2 - (4*a*c)
Se(Delta>=0) Entao
  VetRaizes[1] <- (-b +RaizQ(Delta))/(2*a)
  VetRaizes[2] <- (-b -RaizQ(Delta))/(2*a)
Senao //raízes complexas
  Delta2 <- Abs(Delta)
  x1R <- (-b)/(2*a)
  x1R <- Int(x1R*10^5+0.5)/10^5 //com cinco decimais
  x1C <- RaizQ(Delta2)/(2*a)
  x1C <- Int(x1C*10^5+0.5)/10^5
  x2R <- (-b)/(2*a)
  x2R <- Int(x2R*10^5+0.5)/10^5
  x2C <- RaizQ(Delta2)/(2*a)
  x2C <- Int(x2C*10^5+0.5)/10^5
  x1S <- NumpCarac(x1R) + " + " + NumpCarac(x1C) + "i"
  x2S <- NumpCarac(x2R) + " - " + NumpCarac(x2C) + "i"
```

FimSe

FimProcedimento *//fim do procedimento "ProRaiz2G"*

//-----

Procedimento **RaizNG**(n:inteiro;Xo,tol:real)
//Calcula as raízes do polinômio de grau n.

var i, k, NR: inteiro

Início

LimpaTela

Escreval("")

Escreval("Resultados parciais nas iterações")

NR <- n *//salva o número de raízes em NR*
{Cálculos dos valores refinados de Xi}

Iter <- 0

Repita *//loop para calcular uma nova raiz Xi*

```
  Coef2[n] <- Coef1[n]
  Coef3[n] <- Coef1[n]
```

Repita

```
  Iter <- Iter + 1
  Para i De (n-1) Ate 0 Passo -1 Faca
    Coef2[i] <- Coef1[i] + Coef2[i+1]*X0
    Coef3[i] <- Coef2[i] + Coef3[i+1]*X0
```

FimPara

Xi <- Xo - Coef2[0]/Coef3[1]

Erro <- Abs(Xi - Xo)

Escreval("Iteração:", Iter)

Escreval("Raiz=",Xi:8:5)

Xo <- Xi

Ate((Erro<tol) **ou** (Iter>MaxIte)) *//valida erro cometido com iterações*

Se(Iter>MaxIte) **Entao**

Convergiu <- **Falso**

Senao

Escreval("Raiz detectada = ",Xi:8:6)

Escreval("")

VetRaizes[n] <- Xi

FimSe

n <- n - 1 *//decrementa o grau do polinômio*

Para i **De** n **Ate** 0 **Passo** -1 **Faca**

Coef1[i] <- Coef2[i+1] *//ajeita os coeficientes auxiliares*

FimPara

```

Ate(n=0) //fim do loop para calcular uma nova raiz
{Início dos cálculos de refinamento das raízes}
Se(Convergiu) Entao
    Se(Iter<=MaxIte) Entao //verifica o número de refinamentos com o limite
        Para i De 1 Ate NR Faca
            Xo <- VetRaizes[i]
            Coef2[NR] <- Coef0[NR]
            Coef3[NR] <- Coef0[NR]
            Repita
                Para k De (NR-1) Ate 0 Passo -1 Faca
                    Coef2[k] <- Coef0[k] + Coef2[k+1]*Xo
                    Coef3[k] <- Coef2[k] + Coef3[k+1]*Xo
                FimPara //fimPara-k
                Xi <- Xo - (Coef2[0]/Coef3[1])
                Erro <- Abs(Xi-Xo)
                Xo <- Xi
            Ate(Erro<tol)
            VetRaizes[i] <- Xi
        FimPara //fimPara-NR
    {Fim dos cálculos de refinamento das raízes}
FimSe
FimSe //FimSe-convergiu
FimProcedimento //fim do procedimento "RaizNG"
//=====
//Programa principal
Início
    LimpaTela
    Escreval("")
    Convergiu <- Verdadeiro
    n <- 1
    Enquanto ((n<2) ou (n>10)) Faca //valida o grau do polinômio
        Escreva("Digite o grau do polinômio [min 2 - max 10]: ")
        Leia(n)
    FimEnquanto //fim da validação o grau do polinômio
    LePolinomio(n) //chama rotina para ler os coeficientes do polinômio
    {Analisa o grau do polinômio para calcular as raízes}
    Escolha n
        Caso 2
            a <- Coef1[2]
            b <- Coef1[1]
            c <- Coef1[0]
            ProRaiz2G(a,b,c) //chama rotina para o polinômio de grau 2
        OutroCaso
            Repita
                Escreval("")
                Escreva("Digite a tolerância permitida nos refinamentos: ")
                Leia(tol)
                //tol=0.005 é um bom valor para maior rigor nos cálculos
            Ate(tol>0)
            Escreval("")
            Escreva("Digite um valor inicial para Xo para os refinamentos: ")
            Leia(Xo)
            //xo=0.5 é um bom valor para maior rigor nos cálculos
            Escreval("")
            Escreval("")
            Escreval("")
            Escreval("")
            Escreva("Pressiona <ENTER> para validar os dados e continuar... ")
            Leia(nada) //apenas para provocar uma parada temporária

```

```

    RaizNG(n,Xo,tol) //chama rotina para o polinômio de grau superior a 2
FimEscolha
Escreval("")
LimpaTela
Se(Convergiu) Entao
    Escreval("")
    Escreval("Análise geral dos resultados")
    Escreval("-----")
    Escreva("Polinômio: ")
    Escreval("")
    MontaPolinomio(n) //chama rotina para exibir textualmente o polinômio
    Escreval("")
    Escreval("")
    Se(Delta<0) Entao
        Escreval("Polinômio de grau 2 com raízes complexas e conjugadas:")
        Escreval("x1 = ",x1S)
        Escreval("x2 = ",x2S)
    Senao
        Se(Iter>0) Entao
            Escreval("Raízes do polinômio com", Iter," iterações:")
        Senao
            Escreval("Raízes do polinômio:")
        FimSe
        Para j De 1 Ate n Faca
            Escreval("x",NumpCarac(j), ": ",VetRaizes[j]:5:2)
        FimPara
    FimSe
Senao
    Escreval("")
    Escreval("Não convergiu com",MaxIte," iterações. Pode existir raiz complexa.")
    Escreval("")
    Escreval("")
    Escreva("")
    Escreva("Pressiona <ENTER> para encerrar...")
    Leia(nada) //apenas para provocar uma parada temporária
FimSe
Escreval("")
FimAlgoritmo //fim do programa "RaizesDePolinomio"

```

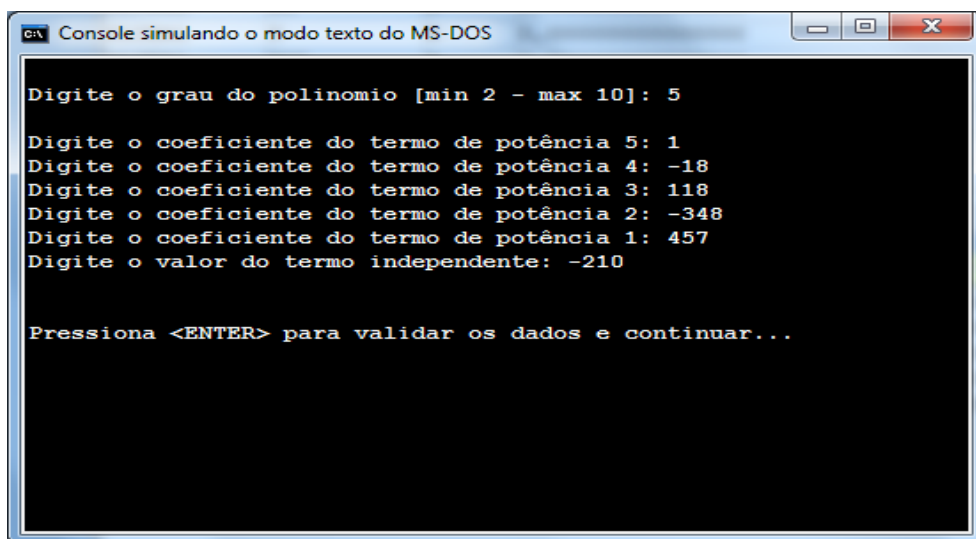
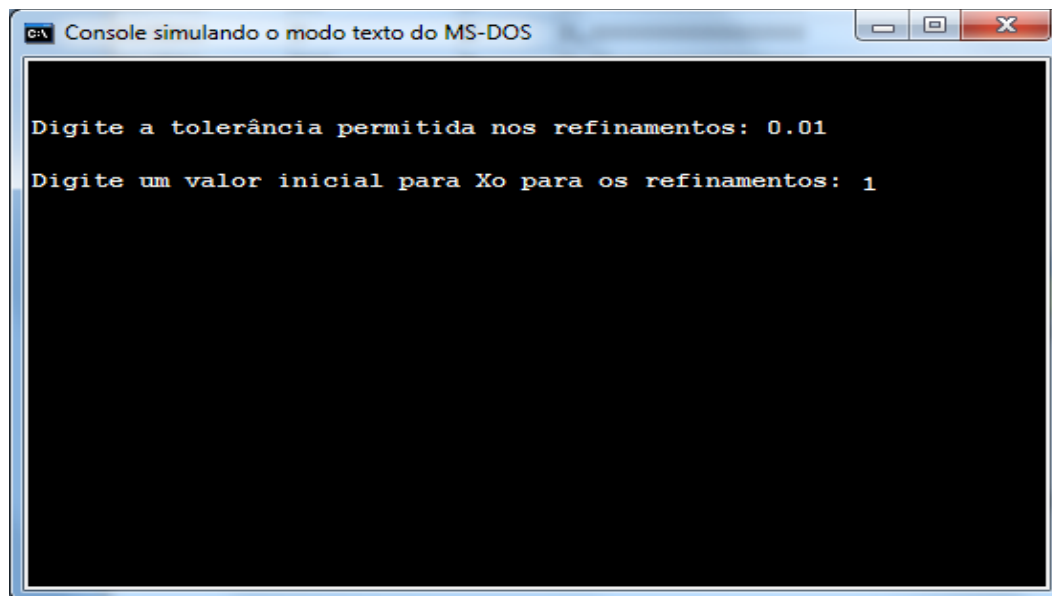


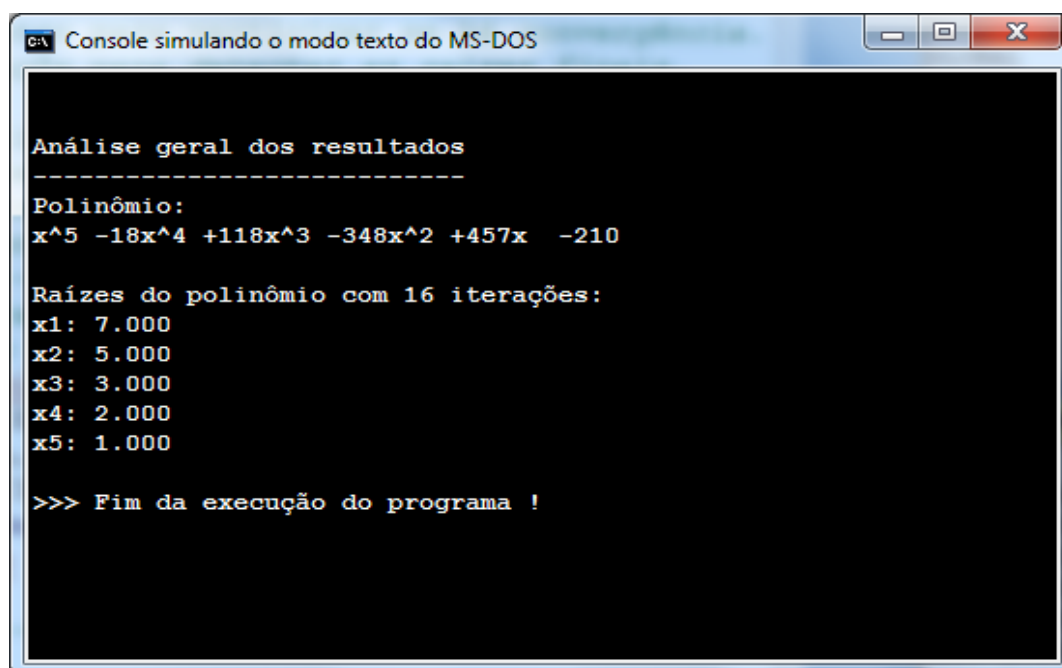
Figura 1a - Entradas dos coeficientes da equação



```
C:\> Console simulando o modo texto do MS-DOS

Digite a tolerância permitida nos refinamentos: 0.01
Digite um valor inicial para Xo para os refinamentos: 1
```

Figura 1b - Entradas da tolerancia (to) e Xinicial



```
C:\> Console simulando o modo texto do MS-DOS

Análise geral dos resultados
-----
Polinômio:
x^5 -18x^4 +118x^3 -348x^2 +457x -210

Raízes do polinômio com 16 iterações:
x1: 7.000
x2: 5.000
x3: 3.000
x4: 2.000
x5: 1.000

>>> Fim da execução do programa !
```

Figura 1c - Obtenção das raízes