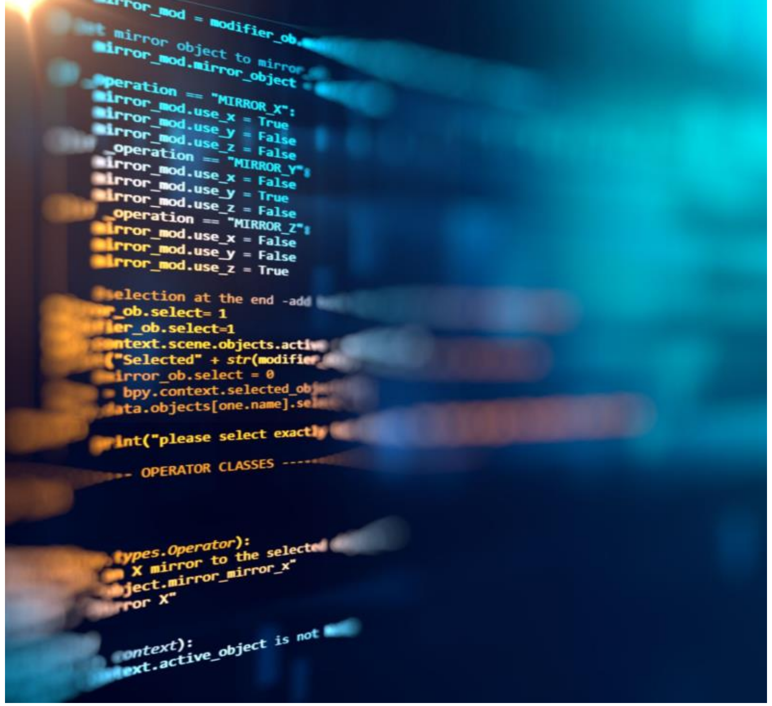
SEMANA 4



COMPUTAÇÃO CIENTÍFICA

ZEROS DE FUNÇÕES REAIS: MÉTODO DE BISSEÇÃO, NEWTON-RAPHSON E SECANTE

PARTE II

MÉTODO DE NEWTON-RAPHSON

MÉTODO DA SECANTE

Computação Científica

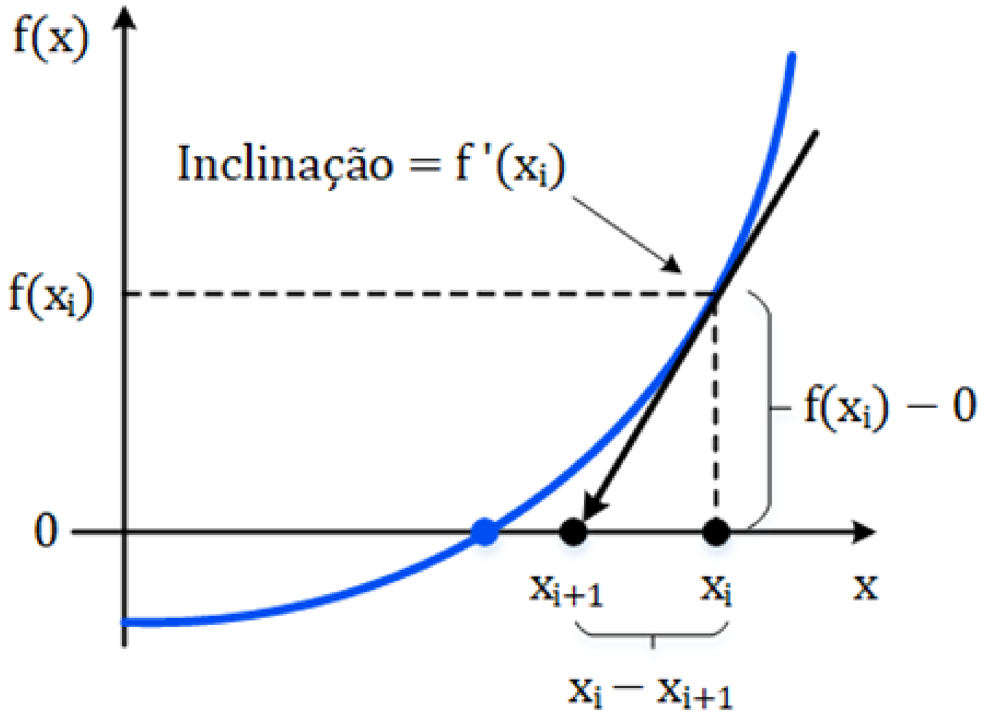
prof. Marco Villaça

Métodos abertos

- Os métodos abertos são baseados em fórmulas que exigem um ou dois valores iniciais da variável independente;
- Algumas vezes se afastam da raiz (divergem) à medida que os cálculos avançam;
- Quando convergem, são mais rápidos que os métodos intervalares.

Método de Newton-Raphson

- **Método mais utilizado para localizar raízes de uma função.**
- Na Figura, supondo x_i a aproximação inicial da raiz de $f(x)$, o ponto onde a tangente ao ponto $[x_i, f(x_i)]$ intercepta o eixo x normalmente é uma aproximação melhor da raiz.



Método de Newton-Raphson

- Utilizando a interpretação geométrica da tangente, obtém-se:

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

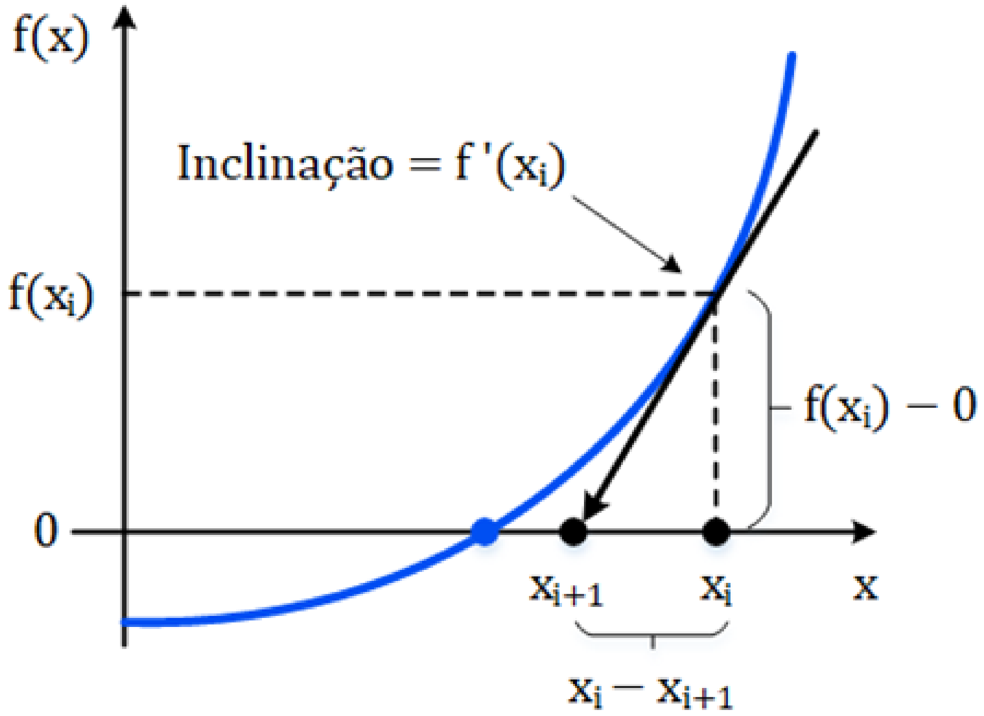
$$\Downarrow$$

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

$$\Downarrow$$

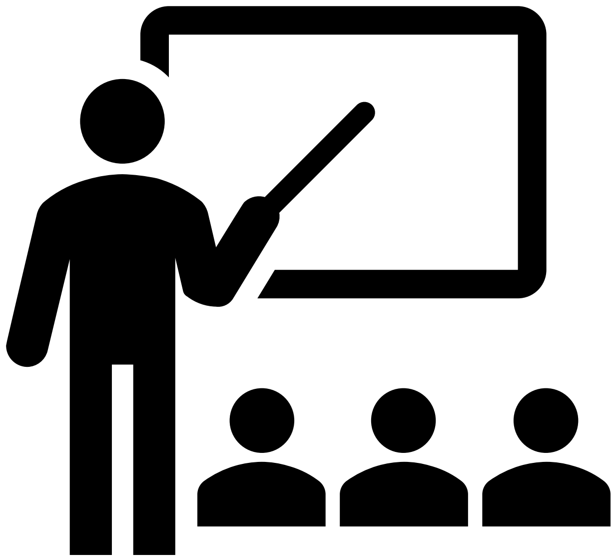
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

que é fórmula de recorrência de Newton-Raphson



EXEMPLO 1

- Utilizar o método de Newton-Raphson para encontrar a raiz de:
$$f(x) = e^{-x} - x$$
utilizando como critério de parada $\varepsilon_s = 1\,\%$.



EXEMPLO 1

- Para aplicar o método de Newton-Raphson, deve ser encontrada $f'(x)$:
- Primeira iteração ($k = 1$)

$$f(x) = e^{-x} - x \Rightarrow f'(x) = -e^{-x} - 1$$

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

$f(x)$ $\Downarrow$ $f'(x)$

$$x_{i+1} = x_i + \frac{e^{-x_i} - x_i}{e^{-x_i} + 1}$$

$i = 0$, $x_0 = 0$ $\Rightarrow$

$$x_1 = 0 + \frac{1 - 0}{1 + 1} = 0{,}5$$

$$|e_a^k| = |e_a^1| = \left|\frac{0{,}5 - 0}{0{,}5}\right| \times 100 = 100\%$$

EXEMPLO 1

- Segunda iteração ($k = 2$)

$$i = 1$$

$$x_{i+1} = x_i + \frac{e^{-x_i} - x_i}{e^{-x_i} + 1} \quad \Rightarrow \quad x_2 = 0{,}5 + \frac{0{,}606531 - 0{,}5}{0{,}606531 + 1} = 0{,}566311$$

$$x_1 = 0{,}5$$

$$\left|e_a^k\right| = \left|e_a^2\right| = \left|\frac{0{,}566311 - 0{,}5}{0{,}566311}\right| \times 100 = 11{,}7\%$$

EXEMPLO 1

- Terceira iteração ($k = 3$)

$i = 2$

$$x_{i+1} = x_i + \frac{e^{-x_i} - x_i}{e^{-x_i} + 1} \Rightarrow x_3 = 0{,}566311 + \frac{0{,}567616 - 0{,}566311}{0{,}567616 + 1} = 0{,}567143$$

$x_2 = 0{,}566311$

$$\Downarrow$$

$$\left|e_a^k\right| = \left|e_a^3\right| = \left|\frac{0{,}567143 - 0{,}566311}{0{,}567143}\right| \times 100 = 0{,}147\%$$

EXEMPLO 1

- A tabela ao lado apresenta

  os resultados para $|\varepsilon_t|_{\%} = 1 \times 10^{-8}\ \%$

- Raiz verdadeira: x = 0,56714329.
- A razão da rápida convergência, é apontada por Chapra e Canale (2010, p. 150), que mostram que o erro deve ser proporcional ao quadrado do erro anterior:

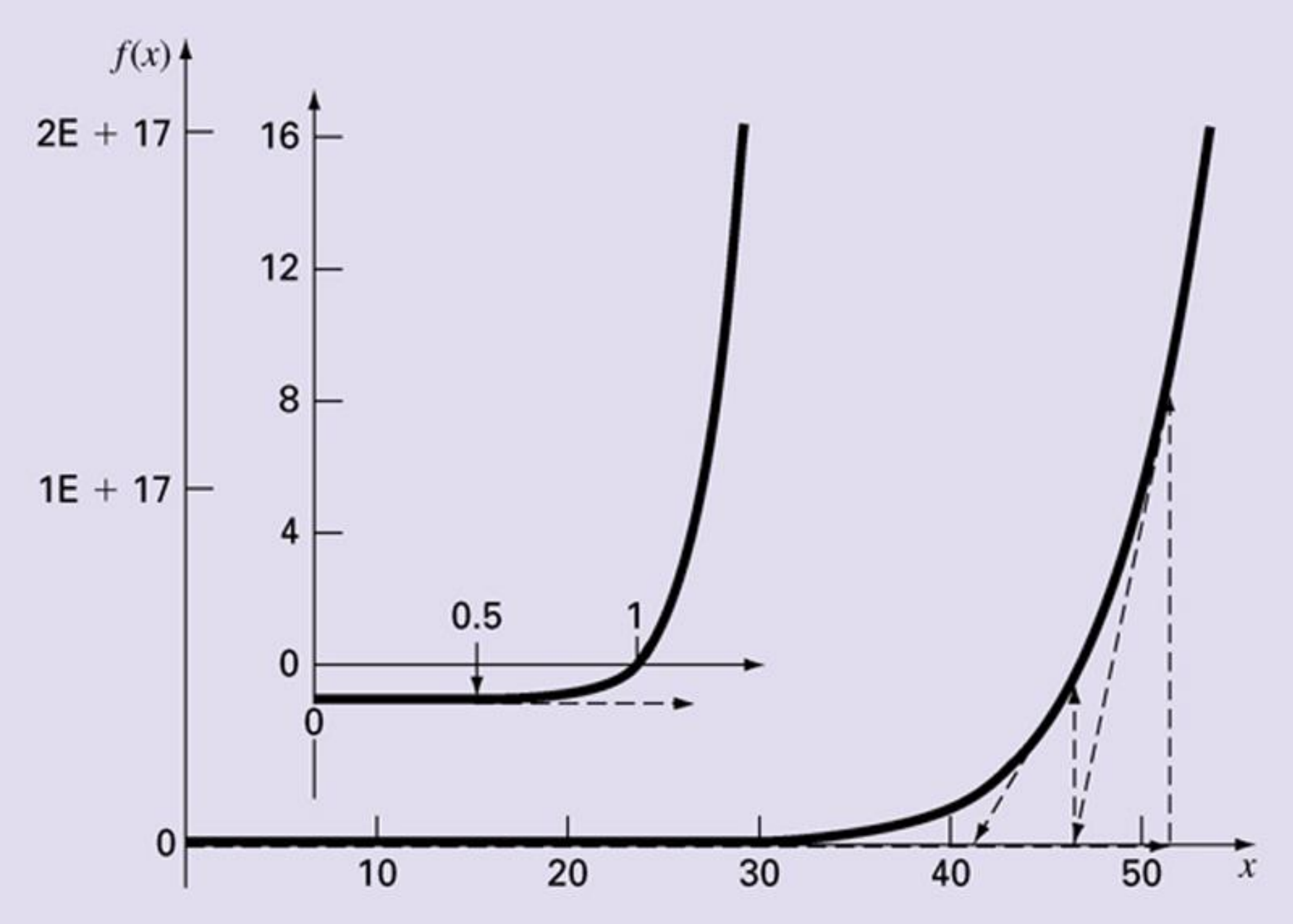
$$E_t^{i+1} = \frac{-f''(x_r)}{2\,f'(x_r)} \cdot {E_t^{i}}^2$$

i	x_i	$\lvert\varepsilon_t\rvert$, %
0	0	100
1	0.500000000	11.8
2	0.566311003	0.147
3	0.567143165	0.0000220
4	0.567143290	$<10^{-8}$

o que significa que o número de algarismos decimais corretos dobra a cada iteração.
Na equação, x_r é a raiz exata de $f(x)$.

Armadilhas do método

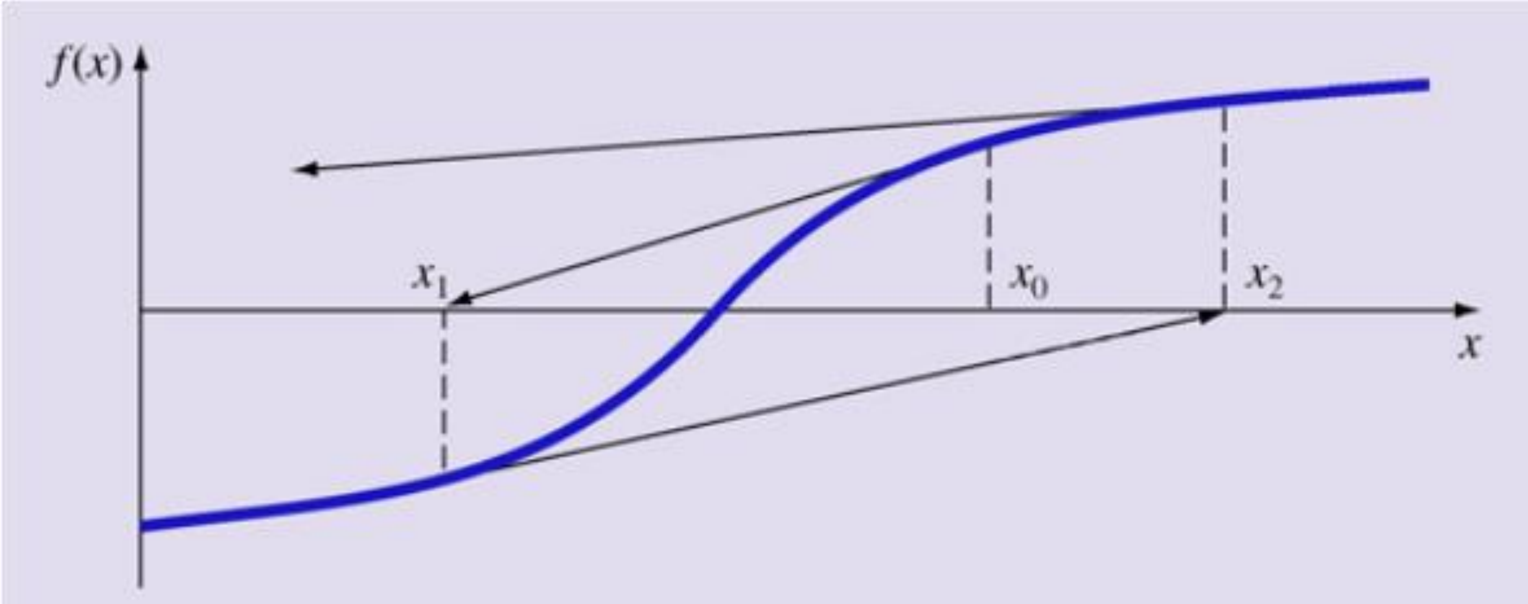
- Representação gráfica do método de Newton-Raphson para um caso com convergência lenta.



Armadilhas do método

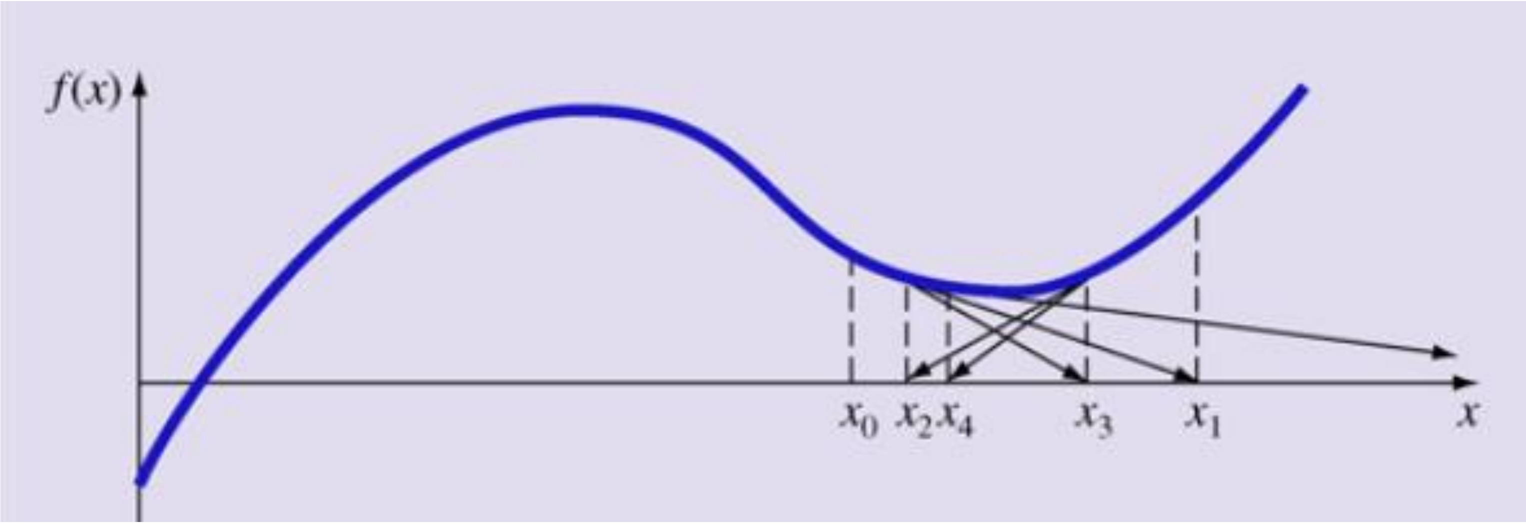
- Raiz perto de um ponto de inflexão ($f''(x) = 0$):

as iterações se afastam progressivamente da raiz.



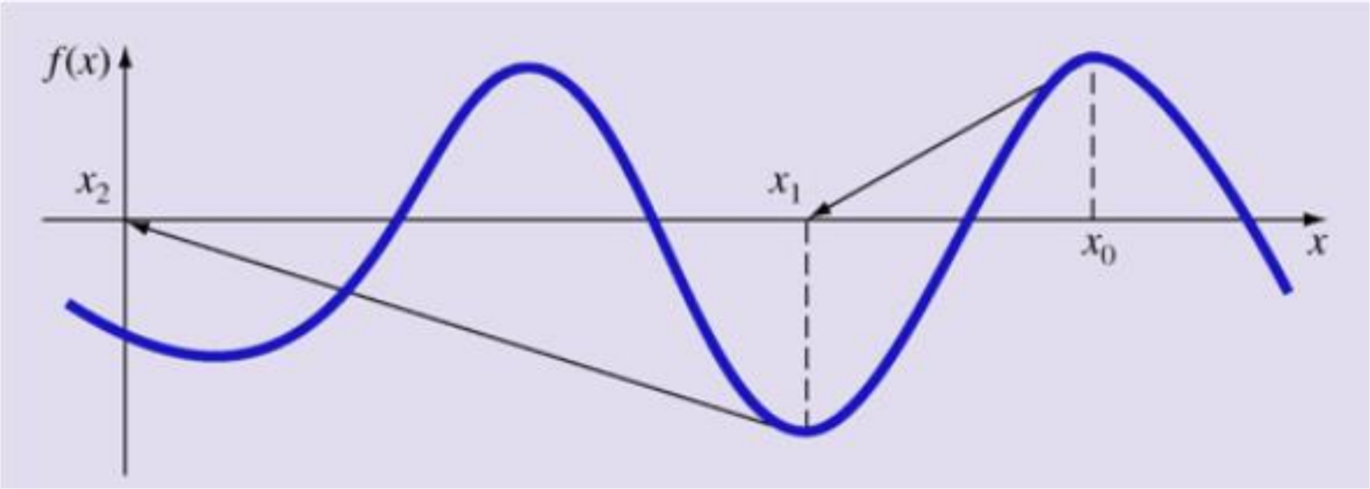
Armadilhas do método

- Oscilações em torno de uma posição de máximo ou de mínimo, afastam a solução da raiz procurada.



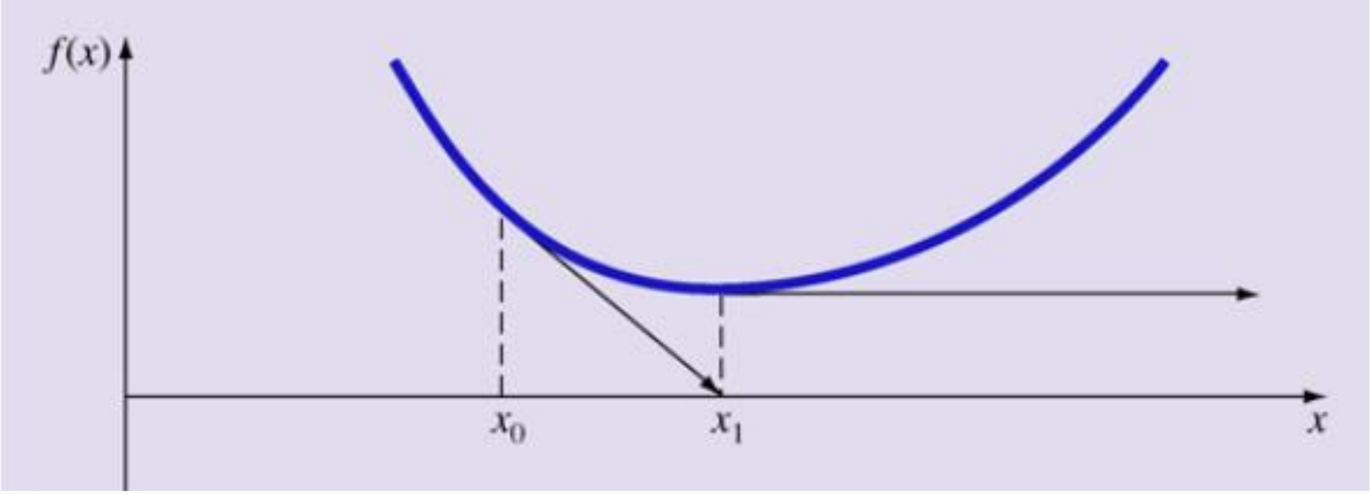
Armadilhas do método

- Uma aproximação próxima a uma raiz pode pular para uma posição a várias raízes de distância, porque uma inclinação próxima de zero foi encontrada.



Armadilhas do método

- Foi encontrada uma inclinação nula ($f'(x) = 0$) e a solução dispara horizontalmente, nunca atingindo o eixo x ($\div 0$).



Método de Newton-Raphson
Considerações

- Devido aos seus problemas potenciais (convergência lenta ou oscilante e divergência em alguns casos) um programa para localizar raízes com o método de Newton-Raphson deve:
  - ✓ Incorporar uma rotina gráfica para auxiliar na escolha condição inicial;
  - ✓ Ao final da rotina, checar a raiz encontrada substituindo-se na função original. Um pequeno valor de ε_a não garante que a solução esteja próxima da raiz;
  - ✓ Definir um limite no número de iterações;
  - ✓ Deve considerar a possibilidade que $f'(x)$ pode tornar-se zero durante a computação.

Função Scilab

Método de Newton-Raphson

Função Scilab

```
function [raiz, iter]=new_raphson(funcao, derivada, es, maxi)
// Cálculo das raizes por Newton-Raphson
// onde raiz é a raiz procurada de funcao
//      iter é o número de iterações realizadas para o erro especificado
//      funcao é a função de entrada literal em x
//      derivada é a derivada da função de entrada literal em x
//      es é o criterio de parada que é opcional
//      maxi é o numero maximo de iterações
//      A condição inicial x0 é escolhida com o auxilio de um grafico
// Exemplo de chamada:
//
// fun = ‘log(x) + x‘
// dxdt = ‘(1/x) + 1‘
// [raiz,iter]=new_raphson(fun, dxdt, 0.0001,50)
//
```

Função Scilab

```
// Construção do gráfico da função
    a = input("Entre com o limite inferior de x a = ");
    b = input("Entre com o limite superior de x b = ");
    x = linspace(a,b,100);
    f = evstr(funcao)
    plot2d(x,f);
    xgrid;
// escolha do valor inicial
    x0 = input("Entre com o valor inicial x0 = ");
    i = 0;  x = x0;
```

Função Scilab

```
// se es nao foi estabelecido usa 0.0001%
   if argn(2) < 3 then
      es = 0.0001;
   end
 // se maxi nao foi estabelecido usa 50
   if argn(2) < 4 then
      maxi = 50;
   end
```

Função Scilab

```
printf("Iter\tRaiz        \terro aproximado %% \n");
// inicio do processo iterativo
 while 1 do
    fxi = evstr(funcao);        dxi = evstr(derivada);
    if dxi == 0
       error('Derivada igual a zero, o processo divergiu');
    end
    xi = x - (fxi/dxi);
    i= i+1;
    if xi ~=0 then // xi não pode ser zero
       ea = abs((xi - x)/xi)*100;
    end
    printf("%d\t%.10f\t%f\n",i,xi,ea);
    if ea < es | i >= maxi then
       break;
    end
    x = xi;
 end
```

Função Scilab

```
  if i == maxi then
      raiz = 'divergiu';
   else
      raiz = xi;
      printf("\nf(%f) = %f\n",xi,fxi);
   end
   iter = i;
endfunction
```

Método da Secante

- Para aplicar o método de Newton-Raphson é necessário calcular a derivada da função.
- O cálculo da derivada de certas funções pode ser extremamente laborioso.
- Para esses casos, indica-se o uso do Método da Secante.

Método da Secante

- Sabe-se que a derivada pode ser aproximada por uma diferença dividida regressiva:

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

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

equação conhecida como Método da Secante, que exige duas estimativas iniciais

Método da Secante Modificado

- Em vez de utilizar dois valores arbitrários, o Método da Secante Modificada utiliza uma pequena perturbação da variável independente:

$$x_{i-1} = x_i + \delta\, x_i$$

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)} \Rightarrow x_{i+1} = x_i - \frac{f(x_i)\delta\, x_i}{f(x_i + \delta\, x_i) - f(x_i)}$$

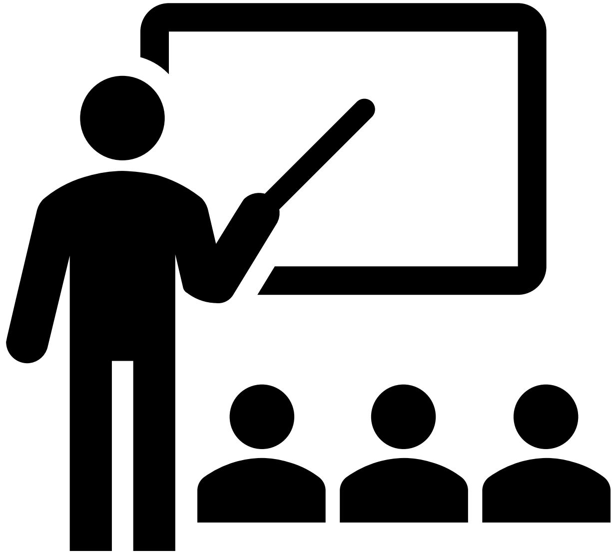
que é a equação que implementa o Método da Secante Modificado.

EXEMPLO 2

- Use o método da secante modificado para estimar a raiz de

$$f(x) = \cos(x) - x \cdot e^x$$

no intervalo $-2 \leq x \leq 0$, utilizando como critério de parada $\varepsilon_s = 1\,\%$.



EXEMPLO 2

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

$$f(x) = \cos(x) - x \cdot e^x$$

$$x_{i+1} = x_i - \frac{[\cos(x_i) - x_i \cdot e^{x_i}]\,\delta x_i}{[\cos(x_i + \delta x_i) - (x_i + \delta x_i) \cdot e^{x_i + \delta x_i}] - [\cos(x_i) - x_i \cdot e^{x_i}]}$$

$$\delta = 0{,}001$$

$$x_0 = -1$$

$$i = 0$$

$$x_1 = -1 - \frac{[\cos(-1) + 1 \cdot e^{-1}] \cdot 0{,}001 \cdot (-1)}{[\cos(-1{,}001) + 1{,}001 \cdot e^{-1{,}001}] - [\cos(-1) + 1 \cdot e^{-1}]}$$

EXEMPLO 2

$$x_1 = -1 - \frac{[\cos(-1) + 1 \cdot e^{-1}] \cdot 0{,}001 \cdot (-1)}{[\cos(-1{,}001) + 1{,}001 \cdot e^{-1{,}001}] - [\cos(-1) + 1 \cdot e^{-1}]}$$

$$\Downarrow$$

$$x_1 = -1 - \frac{0{,}908182 \cdot (-0{,}001)}{0{,}907340 - 0{,}908182} = -2{,}078601$$

- As aproximações seguintes da raiz são calculadas e os dados são colocados na tabela a seguir. Após apenas mais duas iterações, encontra-se a raiz

$x = -1{,}863986$ com precisão de 0,52 %.

EXEMPLO 1

- Processo de obtenção da raiz para a função $f(x) = \cos(x) - x \cdot e^x$

Iteração k	x_k	$\varepsilon_a^{\mathrm{k}}$ %
1	-2.078601	51,89
2	-1.854263	12,10
3	-1.863986	0,52

Função Scilab

Método da Secante Modificado

Função Scilab

```
function [raiz, iter]=sec_mod(funcao, dxi, es, maxi)
   // Cálculo das raizes pelo método da secante modificado
   // function [raiz,iter]=new_raphson(funcao, derivada, x0, es, it)
   // onde raiz é a raiz procurada de funcao
   // iter é o n. de iterações realizadas para o erro especificado
   // funcao é a função de entrada literal em x
   // dxi é a perturbação em torno de x e é opcional
   // es é o criterio de parada que é opcional
   // maxi é o numero maximo de iterações e é opcional
   // A cond. inicial x0 é escolhida com auxilio de um gráfico
   // Exemplo de chamada:
  //
  // fun = ‘log(x) + x‘
  // [raiz,iter]=new_raphson(fun, 0.001, 0.0001,50)
  //
```

Função Scilab

```
// Construção do gráfico da função
   a = input("Entre com o limite inferior de x a = ");
   b = input("Entre com o limite superior de x b = ");
   x = linspace(a,b,100);
   f = evstr(funcao)
   plot2d(x,f);
   xgrid;
// escolha do valor inicial
   x0 = input("Entre com o valor inicial x0 = ");
   i = 0;  x = x0;
```

Função Scilab

```
// se dxi não for estabelecido, adota-se 1e-6
if argn(2) < 2 then
    dxi = 1e-6;
end
// se es nao foi estabelecido usa 0.0001%
if argn(2) < 3 then
    es = 0.0001;
end
// se maxi nao foi estabelecido usa 50
if argn(2) < 4 then
    maxi = 50;
end
```

Função Scilab

```
printf("Iter\tRaiz\t\terro aprox. %% \n");
 // inicio do processo iterativo
 while 1 do
    fxi = evstr(funcao);
    p = x*dxi       // calcula a perturbação
    x = x + p;      // soma a perturbação
    fdxi = evstr(funcao);
    x = x - p       // desconta a perturbação
    xi = x - fxi*p/(fdxi-fxi);
    i= i+1;
    if xi ~=0 then // xi não pode ser zero
       ea = abs((xi - x)/xi)*100;
    end
    printf("%d\t%.10f\t%f\n",i,xi,ea);
    if ea < es | i >= maxi then
       break;
    end
    x = xi;
 end
```

Função Scilab

```
  if i == maxi then
      raiz = 'divergiu';
   else
      raiz = xi;
      printf("\nf(%f) = %f\n",xi,fxi);
   end
   iter = i;
endfunction
```

Raízes de polinômios no Scilab

- Quando se busca for por todas as raízes de um polinômio, o *Scilab* oferece a função *roots*.
  - Polinômio definido pelos seus coeficientes:

```
-->p1 = poly([-6 1 1],'x','c')
 p1  =
                2
  - 6 + x + x

-->roots(p1)
 ans  =


  - 3.
    2.
```

Raízes de polinômios no Scilab

- Outra forma de definir o mesmo polinômio:

```
-->x = poly(0,'x')
 x  =
    x

-->p1 = x^2 + x - 6
 p1  =
            2
  - 6 + x + x

-->roots(p1)
 ans  =

  - 3.
    2.
```

Raízes de polinômios no Scilab

- Valor numérico de um polinômio:

```
-->p1 = poly([-6 1 1],'x','c')
 p1  =
                2
  - 6 + x + x

-->horner(p1,1) // valor de p1 em x = 1
 ans  =

  - 4.
```

Raízes de polinômios no Scilab

- Um polinômio pode também ser definido pelas suas raízes:

```
-->p1 = poly([-3 2],'x','r')
 p1  =
 
                2
   - 6 + x + x
```

Exercício 1

- Use os métodos de Newton-Raphson e da secante para estimar as raízes de

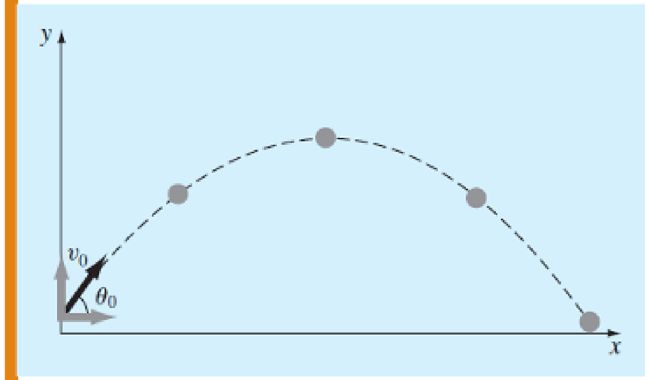
$$f(x) = x^3 - 10$$

no intervalo $0 \leq x \leq 4$

- Manualmente, utilizando como critério de parada $|\varepsilon_a| < 1\ \%$.
- Com o auxílio das funções Scilab, utilizando como critério de parada $|\varepsilon_a| < 0{,}0001\ \%$.

- Em um circuito RLC paralelo, encontre numericamente a frequência ω que resulta em uma impedância Z de 100 Ω utilizando o método da secante modificado, considerando $R = 225\ \Omega$, $C = 0{,}6\ \mu F$ e $L = 0{,}5\ H$.

Exercício 2



$$y = tg(\theta_0)\cdot x - \frac{g}{2\cdot v_0^2\cdot \cos^2\theta_0}\cdot x^2 + y_0$$

Exercício 3

- Os engenheiros aeroespaciais algumas vezes calculam a trajetória de projéteis. Um problema relacionada trata da trajetória de uma bola lançada, definida pelas coordenadas (*x,y*), conforme é mostrado na Figura do canto superior esquerdo
- A trajetória pode ser modelada pela equação do canto inferior esquerdo.
- Encontre o ângulo inicial apropriado θ_0, se a velocidade inicial for v_0 = 30 m/s e a distância *x* do receptor for 90 m. A bola deixa a mão do lançador a uma elevação y_0 = 1,8 m e o receptor a recebe a 1 m.

Bibliografia e crédito das figuras

CHAPRA, Steven. **Applied numerical methods with MATHLAB for engineers and scientists.** McGrawHill, 2012.

CHAPRA, Steven e CANALE, Raymond. **Numerical methods for engineers.** McGrawHill, 2010.