Métodos Numéricos

Prof. Dr. Jonatha Costa

2024

MN aplicados à Engenharia

- Apresentar conteúdo de Resolução de Equações Não-Lineares
  - Método da Bisseção
  - Método Regula-Falsi
  - Método Newton-Raphson
  - Método da Secante

Organização

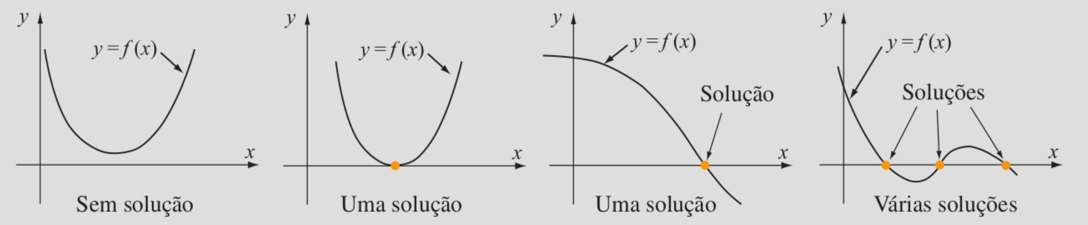
1. Resolução de Sistemas Não-Lineares
   - Fundamentos
   - Método da Bisseção
   - Método Regula-Falsi
   - Método Newton-Raphson
   - Método da Secante

MN aplicados à Engenharia

Fundamentos de Resolução de Sistemas Não-Lineares

- Equações precisam ser resolvidas em todas áreas da engenharia;
- Consiste em determinar o(s) valor(es) de x tal que $f(x) = 0$.

Figura: Existência de solução em curvas



Fonte: GILAT,(2008)

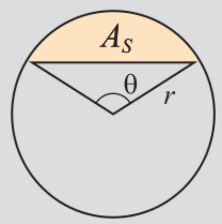
- Um número real ξ é um zero da função f(x) ou uma raiz da equação $f(x) = 0$ se $f(\xi) = 0$.

MN aplicados à Engenharia

Fundamentos

- Em alguns casos as soluções podem ser reais ou imaginárias.
- Seja o exemplo da área do segmento sombreado, dada por $A_s = \frac{1}{2}r^2(\theta - sen(\theta))$.

Figura: Área de setor



Fonte: GILAT,(2008)

Perceba que:

- A determinação de θ em função de A_S e r conhecidos não é possível através de solução analítica.
- Para tanto, utilizam-se os métodos numéricos para encontrar uma solução aproximada para tanto.

MN aplicados à Engenharia

Zeros de funções reais

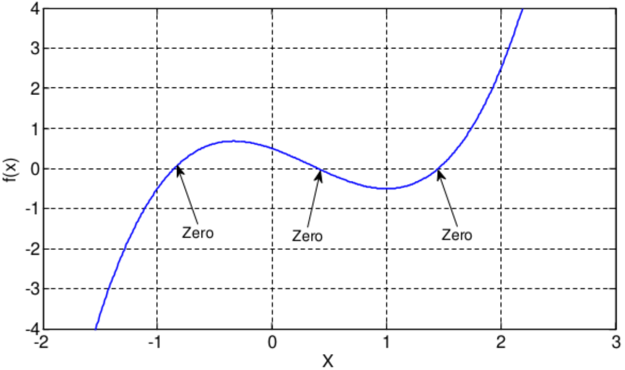
- Quais os zeros das funções:
    - $f(x) = x - 3$;
    - $f(x) = x^2 - 4x - 5$;
    - $f(x) = x^3 - x^2 - x + 1/2$.

MN aplicados à Engenharia

Zeros de funções reais

- Seja uma função $f(x)$ contínua em um intervalo I. Denomina-se zero desta função todo $x \in I$, tal que $f(x) = 0$.

Figura: $f(x) = x^3 - x^2 - x + 1/2$



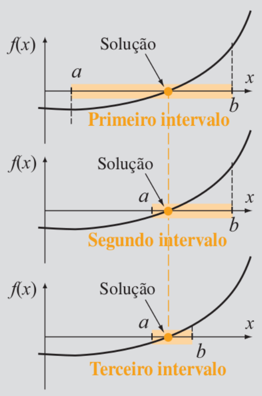
Fonte: DIAS,(2019)

MN aplicados à Engenharia

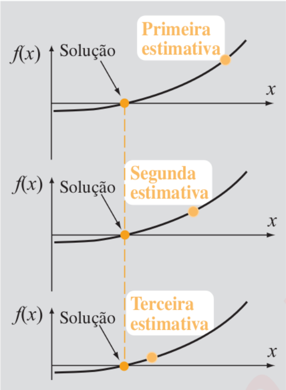
Como obter as raízes de uma equação qualquer?

Figura: Método de obtenção das raízes

(a) Confinamento



(b) Método aberto



Fonte: GILAT,(2008)

MN aplicados à Engenharia

Boas práticas

- Definir a faixa de busca ou isolamento das raízes;
- Estabelecer uma faixa de erro aceitável e tolerância para a aproximação;
- Refinar a busca por meio de um processo iterativo até que a solução tenha uma precisão prefixada.

MN aplicados à Engenharia

- **Estimação de Erros em Soluções Numéricas - Erro real**
  Seja x_{TS} a solução exata tal que $f(x_{TS}) = 0$, e seja x_{NS} uma solução numérica aproximada tal que $f(x_{NS}) = \varepsilon$, em que ε é um número muito pequeno.
  - Deve ser criado algum critério para verificar se uma solução é precisa.
  - *Erro real*

$$erro\ real = x_{TS} - x_{NS}$$

  Este critério não é tão útil, visto que x_{TS} não é conhecido!

- **Estimação de Erros em Soluções Numéricas - Tolerância em $f(x)$:**
  - Ao invés de considerar o erro na solução, verifica-se o desvio de $f(x_{NS})$ em relação a zero.
  - Tolerância em f: $tolerancia = \Big|f(x_{TS}) - f(x_{NS})\Big| = \Big|0 - \varepsilon\Big| = \Big|\varepsilon\Big|$

MN aplicados à Engenharia

- **Estimação de Erros em Soluções Numéricas - Tolerância na solução:**
  - É útil quando métodos de confinamento são usados na determinação numérica;
  - Assume-se que a solução numérica é o ponto central de um intervalo com uma tolerância tal que:

$$x_{NS} = \left(\frac{a+b}{2}\right)^{\pm\left(\frac{a-b}{2}\right)}$$

- **Estimação de Erros em Soluções Numéricas - Erro relativo estimado**
  - Utilizado quando as soluções numéricas são calculadas iterativamente;
  - É dado por:

$$erro\ relativo\ estimado = \left|\frac{x_{NS}^{n} - x_{NS}^{(n-1)}}{x_{NS}^{(n-1)}}\right|$$

MN aplicados à Engenharia

Fase 01 - Confinamento das raízes

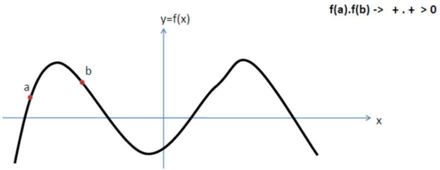
- **Teorema 1**. Seja uma função contínua no intervalo $[a, b]$.

  Se $f(a) \cdot f(b) \leq 0$, então $\exists$ pelo menos um ponto $x = x_0$ entre a e b que é solução de $f(x) = 0$.

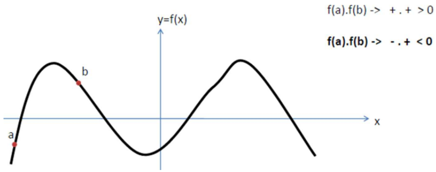
MN aplicados à Engenharia

Figura: Existência de zeros na função

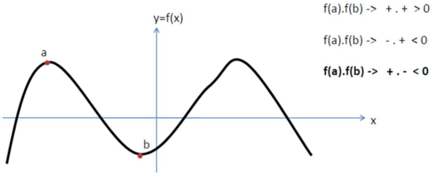
(a) caso 1



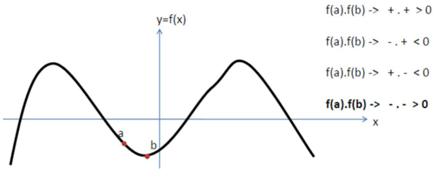
(b) caso 2



(c) caso 3



(d) caso 4



Fonte: DIAS,(2019)

MN aplicados à Engenharia

Fase 01 - Confinamento das raízes

Exemplo: Para determinar os zeros da função $f(x) = x^3 - 9x + 3$, pode-se construir uma tabela de valores de $f(x)$ e proceder à análise dos sinais, como ilustrado na Tabela 1.

Tabela: Confinamento de raízes

$f(-4)f(-3)=(-25).(3)= -75 < 0$

$f(2)f(3)=(-7).(3)= -21 < 0$

x	-4	-3	-2	-1	0	1	2	3
f(x)	-25	3	13	11	3	-5	-7	3

$f(0)f(1)=(3).(-5)= -15 < 0$

Fonte: DIAS, (2019)

Portanto, nos intervalos $[-4, -3]$, $[0, 1]$ e $[2, 3]$, $\exists$ pelo menos uma raiz real da função. Como trata-se de um polinômio do 3º grau, pode-se dizer que existe apenas uma raiz em cada intervalo.

MN aplicados à Engenharia

Fase 02 - Confinamento das raízes

Critério de Parada:

Considerando $\overline{x}$ a raiz aproximada com precisão ε, o qual normalmente é da ordem de 10^{-6}, tem-se:

(i) $|\overline{x} - \xi| < \epsilon$

(ii) $|f(\overline{x})| < \epsilon$

Como realizar o teste (i), uma vez desconhecido o ξ?

Uma forma é reduzir o intervalo que contém a raiz a cada iteração.

Seja $[a, b]$ tal que $\xi \in [a, b], (b - a) < \epsilon$.

Então, $\forall x \in [a, b], |x - \varepsilon| < \xi$ portanto, $\forall x \in [a, b]$ pode ser tomado como $\overline{x}$.

Organização

MN aplicados à Engenharia

Fase 02 - Confinamento das raízes (Método da Bisseção)

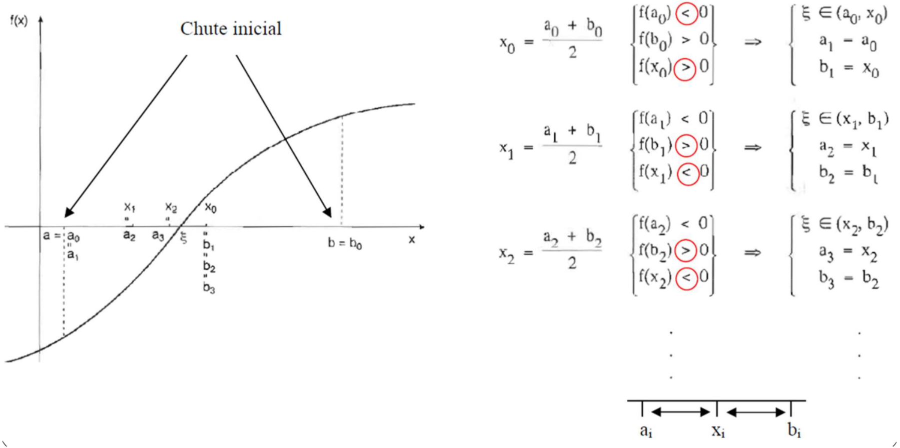
Para se aproximar de uma raiz, o princípio da bisseção consiste em reduzir o intervalo inicial testando o sinal de $f(x)$ para o ponto médio do intervalo.

- Considerando o intervalo $[a, b]$, $x = \frac{a+b}{2}$
- Se $f(a).f(x) < 0$, o novo intervalo é $[a, \frac{a+b}{2}]$
- Se $f(b).f(x) < 0$, o novo intervalo é $[\frac{a+b}{2}, b]$

MN aplicados à Engenharia

Fase 02 - Refinamento (Método da Bisseção)

Figura: Método da Bisseção



Fonte: DIAS,(2019)

MN aplicados à Engenharia

Figura: *Script* Método da Bisseção no Octave/Matlab

```
% Solução de equações não lineares
clear all;clc; pkg load symbolic; format bank
%**************************************************************
F=inline('8-4.5*(x-sin(x))');
%***************************************%********************
a=2;    b=3;                              % Kick-off
imax=50; tol=0.001;
%***************************************%********************
disp("Método da Bisseção!")
fprintf("\niteração \ta\tb\tc\tFa\tFb\tFx\n")
tic
for(i=1:imax)
%***************************************%********************
  x=(a+b)/2;   toli=(b-a)/2;
  fprintf("%5d\t%11.4f %8.4f%8.4f%8.4f%8.4f%8.4f\n",i,a,b,x,F(a),F(b),F(x))
%***************************************%********************
 if (F(a)*F(x)<0)  a=a; b=x; end   %Raiz entre a e xmed => novo 'b'
 if (F(a)*F(x)>0)  a=x; b=b; end   %Raiz entre b e xmed => novo 'a'
%***************************************%********************
  if (toli<tol)
  fprintf("\nSoluçao %.4f alcançada após %d iterações! \n",x,i);
  break
  end
end

fprintf("Tempo de processamento t=%f(s)\n\n",toc)
%***************************************%********************
```

Fonte: AUTOR,(2020)

MN aplicados à Engenharia

Figura: *Script* Método da Bisseção no Python

```python
import numpy as np

f= lambda x: 8-4.5*(x - np.sin(x))     # ou def fun(x): ...
a, b, imax, tol = 2, 3, 50, 0.001
print('iteração  a       b        x      f(a)     f(x)        f(b)')
print(60*'-')
t0 = time.process_time()          #   Ligar cronômetro
# ==========================================================================
if f(a)*f(b)>0:
    print('A raiz não está contida no intervalo dado [%d,%d]!'%(a,b))
    print('Por favor teste um novo intervalo [a,b].')
else:
    X=[]
    for i in range(imax):
        x=(a+b)/2
        X.append(x)
        toli=(b-a)/2
        print('    %d   %.3f    %.3f   %.3f   %.3f   %.3f    %.3f'
              %(i+1,a,b,x,f(a),f(x),f(b)))
        if (f(a)*f(x)<0):   # Raiz localizada entre a e x >> novo b
            b=x
        else:               # Raiz localizada entre b e x >> novo a
            a=x
        if(toli<tol):
            print(60*'-')
            break
    print()
    print('Solução x=',format(x,'.3f'),'encontrada após',i+1,'iterações!')
    print('Processamento computacional em:%.4fs' %(time.process_time()-t0))
```

Fonte: AUTOR,(2024)

MN aplicados à Engenharia

Método da Bisseção

Figura: Resultados Método da Bisseção

```
Método da Bisseção!

iteração        a       b       c       Fa      Fb      Fx
    1        2.0000  3.0000  2.5000  3.0918 -4.8650 -0.5569
    2        2.0000  2.5000  2.2500  3.0918 -0.5569  1.3763
    3        2.2500  2.5000  2.3750  1.3763 -0.5569  0.4341
    4        2.3750  2.5000  2.4375  0.4341 -0.5569 -0.0557
    5        2.3750  2.4375  2.4062  0.4341 -0.0557  0.1907
    6        2.4062  2.4375  2.4219  0.1907 -0.0557  0.0678
    7        2.4219  2.4375  2.4297  0.0678 -0.0557  0.0062
    8        2.4297  2.4375  2.4336  0.0062 -0.0557 -0.0248
    9        2.4297  2.4336  2.4316  0.0062 -0.0248 -0.0093
   10        2.4297  2.4316  2.4307  0.0062 -0.0093 -0.0016

Soluçao 2.4307 alcançada após 10 iterações!
Tempo de processamento t=0.002587(s)
```

Fonte: AUTOR,(2020)

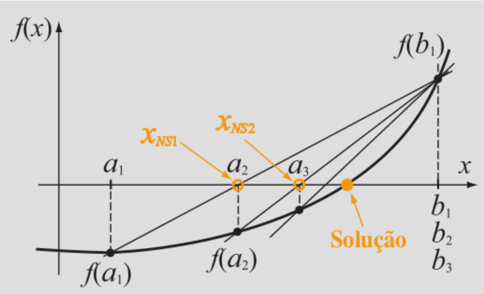
Organização

MN aplicados à Engenharia

Fase 02 - Refinamento (Método regula falsi)

- Também chamado de método da **falsa posição** ou de interpolação linear.
- É um método de confinamento utilizado para se obter a solução de uma equação $f(x) = 0$ quando se tem conhecimento que a solução está dentro de um intervalo $[a, b]$ e $f(x)$ é contínua.

Figura: Método da Bisseção



Fonte: GILAT,(2008)

MN aplicados à Engenharia

Fase 02 - Refinamento (Método regula falsi ou falsa posição)

- Para um intervalo $[a, b]$ a equação da linha reta que conecta os dois pontos $(b, f(b))$ e $(a, f(a))$ é dada por:

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

- O ponto x_{NS}, onde a reta cruza o eixo x, é determinado pela equação a seguir, considerando $f(x) = 0$:

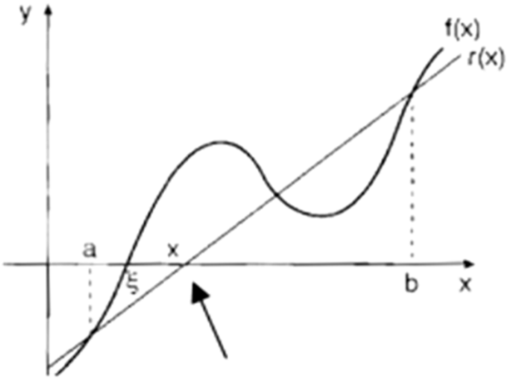
$$x_{NS} = \frac{a.f(b) - b.f(a)}{f(b) - f(a)}$$

MN aplicados à Engenharia

Fase 02 - Refinamento (Método regula falsi)

- Graficamente este ponto x é a interseção entre o eixo das abcissas (x) e a reta $r(x)$ que passa pelos pontos $(a, f(a))$ e $(b, f(b))$.

Figura: Refinamento



Fonte: DIAS,(2019)

MN aplicados à Engenharia

Figura: *Script* Método da Falsa Posição no Octave/Matlab

```
% Solução de equações não lineares
clear all;clc; pkg load symbolic; format bank
F=inline('8-4.5*(x-sin(x))');
%********************************%******************
a=2;    b=3;  imax=30;  tol=0.001;
%********************************%******************
disp("Método da Falsa Posição!")
fprintf("\niteração \ta\tb\tc\tFa\tFb\tFx\n")
%********************************%******************
tic
for(i=1:imax)
x=(a*F(b)-b*F(a))/(F(b)-F(a));
toli=(b-a)/2;
fprintf("%5d\t%11.4f %8.4f%8.4f%8.4f%8.4f%8.4f\n",i,a,b,x,F(a),F(b),F(x))
if (F(a)*F(x)>0) a=x; else b=x; end
%********************************%******************
 if (toli<tol)
  fprintf("\nSoluçao %.4f alcançada após %d iterações! \n",x,i);
  break
  end
end
fprintf("Tempo de processamento t=%f(s)\n\n",toc)
```

Fonte: AUTOR,(2020)

MN aplicados à Engenharia

Figura: *Script* Método da Falsa Posição no Python

```
import numpy as np,time
f= lambda x: 8-4.5*(x - np.sin(x))
a,b,imax,tol = 2, 3, 50, 0.001
print('Método da Falsa Posição!')
print('Intervalo de análise [%d,%d].\n'%(a,b) )
print('iteração  a       b         x      f(a)     f(x)        f(b)')
print(60*'-')
t0 = time.process_time()          #   Ligar cronômetro
# ==============================================================================
if f(a)*f(b)>0:
    print('A raiz não está contida no intervalo dado [%d,%d]!'%(a,b))
    print('Por favor teste um novo intervalo [a,b].')
else:
    X=[]
    for i in range(imax):
        x=( a*f(b) - b*f(a) ) / ( f(b)-f(a) )
        X.append(x)
        toli=(b-a)/2
        print('    %d   %.3f    %.3f   %.4f   %.3f   %.3f    %.3f'
              %(i+1,a,b,x,f(a),f(x),f(b)))
        if f(a)*f(x)>0:  a = x # Raiz localizada entre [a,x] >> [a,b=x]
        else:            b = x # Raiz localizada entre [x,b] >> [a=x,b]
        if(toli<tol): print(60*'-'); break;
    print('\nSolução x=',format(x,'.4f'),'encontrada após',i+1,'iterações!')
    print('Tempo computacional:%.4fs' %(time.process_time()-t0))
```

Fonte: AUTOR,(2024)

MN aplicados à Engenharia

Figura: Resultados da Falsa Posição

```
Método da Falsa Posição!

iteração         a        b        c        Fa        Fb        Fx
    1         2.0000   3.0000   2.3886   3.0918  -4.8650   0.3287
    2         2.3886   3.0000   2.4273   0.3287  -4.8650   0.0252
    3         2.4273   3.0000   2.4302   0.0252  -4.8650   0.0019
    4         2.4302   3.0000   2.4304   0.0019  -4.8650   0.0001
    5         2.4304   3.0000   2.4305   0.0001  -4.8650   0.0000
    6         2.4305   3.0000   2.4305   0.0000  -4.8650   0.0000
    7         2.4305   3.0000   2.4305   0.0000  -4.8650   0.0000
    8         2.4305   3.0000   2.4305   0.0000  -4.8650   0.0000
    9         2.4305   3.0000   2.4305   0.0000  -4.8650   0.0000
   10         2.4305   3.0000   2.4305   0.0000  -4.8650   0.0000
   11         2.4305   3.0000   2.4305   0.0000  -4.8650   0.0000
   12         2.4305   3.0000   2.4305   0.0000  -4.8650   0.0000
   13         2.4305   3.0000   2.4305   0.0000  -4.8650   0.0000
   14         2.4305   3.0000   2.4305   0.0000  -4.8650   0.0000
   15         2.4305   2.4305   2.4305   0.0000   0.0000   0.0000

Soluçao 2.4305 alcançada após 15 iterações!
Tempo de processamento t=0.003625(s)
```

Fonte: AUTOR,(2020)

Organização

MN aplicados à Engenharia

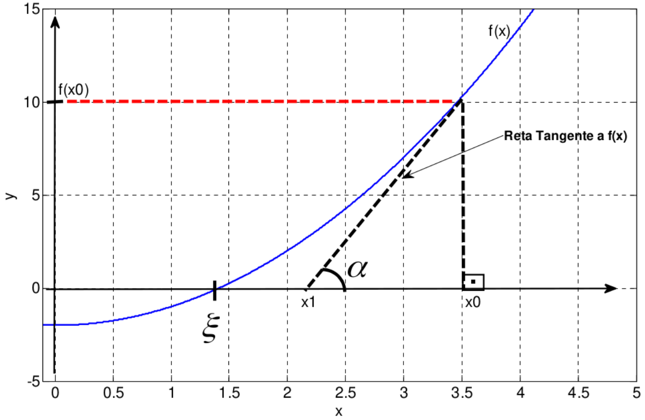
Método de Newton-Raphson

- Dada uma função $f(x)$ contínua no intervalo $[a, b]$ onde existe uma raiz única, é possível determinar uma aproximação de tal raiz a partir de interseção da tangente à curva em um ponto x_0 com o eixo das absissas.
- x_0 - atribuído em função da geometria do método e do comportamento da curva da equação nas proximidade da raiz.

MN aplicados à Engenharia

Método de Newton-Raphson

Figura: Interpretação Gráfica

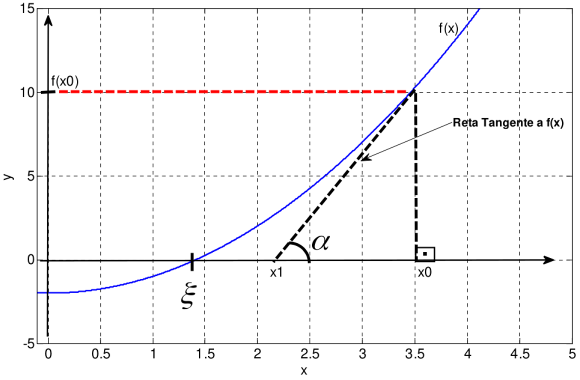


Fonte: DIAS,(2019)

MN aplicados à Engenharia

Método de Newton-Raphson

Figura: Interpretação Gráfica



Fonte: DIAS,(2019)

$$tg(\alpha) = \frac{f(x_0)-0}{x_0-x_1}$$

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

Como: $tg(\alpha) = f'(x_0)$

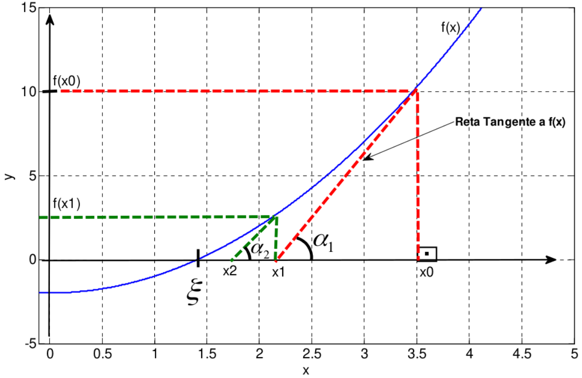
Portanto:

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

MN aplicados à Engenharia

Método de Newton-Raphson

Figura: Interpretação Gráfica



Fonte: DIAS,(2019)

$$tg(\alpha) = \frac{f(x_1)-0}{x_1-x_2}$$

$$tg(\alpha) = \frac{f(x_1)}{x_1-x_2}$$

Como: $tg(\alpha) = f'(x_1)$

Portanto:

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

MN aplicados à Engenharia

Método de Newton-Raphson

Se forem realizadas diversas aproximações

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

$$\vdots \qquad \vdots \qquad \vdots$$

Conclui-se que:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},$$

em que $k = 0, 1, 2, 3, \dots$

MN aplicados à Engenharia

Método de Newton-Raphson

Testes de parada:

- Erro estimado

$$|\varepsilon| = |x_{k+1} - x_k| < \varepsilon_d$$

- Erro relativo estimado

$$|\varepsilon_R| = \left|\frac{x_{k+1} - x_k}{x_{k+1}}\right| < \varepsilon$$

- Tolerância em $f(x)$

$$\left|f(x_{k+1})\right| < \varepsilon_d$$

MN aplicados à Engenharia

Método de Newton-Raphson

Algoritmo:

- Avaliar o $f'(x)$
- Utilizar o x_i para estimar o próximo valor estimado da raiz (x_{i+1})

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

- Encontrar o erro relativo aproximado $\mid \in_a \mid$

$$\mid \in_a \mid = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| x100$$

- Se $\mid \in_a \mid > \in_s$, então retornar ao passo 2, se não interromper o algoritmo.

MN aplicados à Engenharia

Método de Newton-Raphson

Vantagens:

- Rapidez no processo de convergência;
- Desempenho elevado.

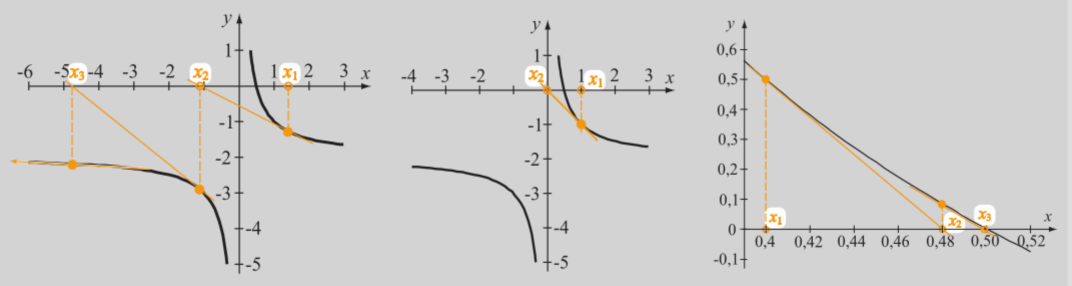
Desvantagens:

- Necessidade da obtenção de $f'(x)$ - o que pode ser impossível em determinados casos;
- O cálculo do valor numérico de $f'(x)$ a cada iteração.

MN aplicados à Engenharia

Erros de Convergência

Figura: Método de Newton usando diferentes pontos de partida

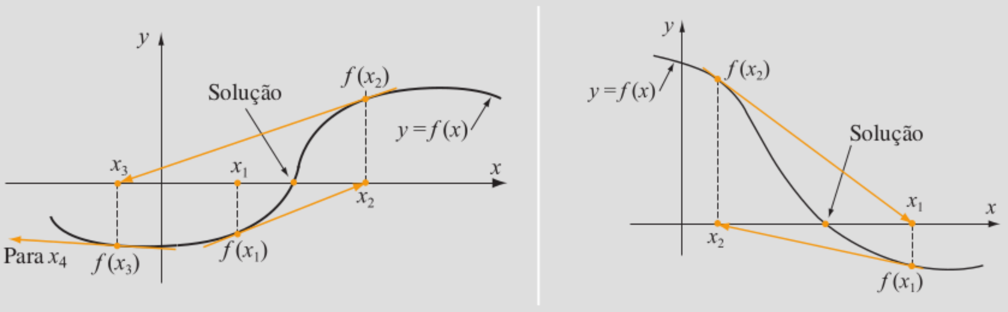


Fonte: GILAT,(2008)

Perceba a relação causa-efeito a partir de ponto de início (x_0) nos itens a,b e c.

MN aplicados à Engenharia

Figura: Erros de Convergência



Fonte: GILAT,(2008)

MN aplicados à Engenharia

Figura: *Script* Método da Newton-Raphson no Octave/Matlab

```
clear all;clc; pkg load symbolic; format bank
%***************************************%********************
f=@(x)(8-4.5*(x-sin(x))); g=(diff(sym(f))); df=matlabFunction(g);clc;
disp("Método de Newton-Raphson")
%***************************************%********************
Err=0.001; tol=Err*0.1; imax=30;
Xest=2; tic              % valor inicial
%***************************************%********************
for(i=1:imax)
  Xsn = Xest-f(Xest)/df(Xest); % Xsn = x(i+1) ; Xest=x(i);
    %***************************************%********************
      if (abs((Xsn-Xest)/Xest) < Err)             % Erro < x(i+1)-xi)/xi|
        Xsn = Xest;
        fprintf("\nSoluçao %.4f alcançada após %d iterações! \n",Xsn,i)
      break
      end
    %***************************************%********************
      if (abs(f(Xsn))< Err)                           % Tol < f(xi)
      fprintf('\nSoluçao %.4f alcançada com %d iteraçoes e tolerancia %6.f!\n',
      Xsn,i,f(Xsn))
      break
      end
    %***************************************%********************
      if i==imax                        % Maximo de iteraçoes
        fprintf("\nSolução não alcançada com %d iterações\n", i)
        Xsn=('Sem resposta!')
      break
      end
    %***************************************%********************
  Xest=Xsn;
end
%***************************************%********************
fprintf("Tempo de processamento t=%f(s)\n\n",toc)
```

Fonte: AUTOR,(2020)

MN aplicados à Engenharia

Figura: *Script* Método da Newton-Raphson no Python

```python
import time
import sympy as sym
# ==========================================================================
x=sym.Symbol('x')
fun  = 8-4.5*(x - sym.sin(x))
f=sym.Lambda(x,fun); df=sym.Lambda(x,sym.diff(f(x),x))
Xest,imax,Err = (2+3)/2 ,30 ,0.001; tol = Err*0.1;
print('Método da Newton Raphson!')
t0 = time.process_time()                        #   Ligar cronômetro
# ==========================================================================
X=[]
for i in range(imax):
    Xsn=Xest- float(f(Xest))/float(df(Xest))        # Xsn=x(i+1);Xest=x(i)
    X.append(Xest)
    if(abs( (Xsn-Xest) / Xest)<Err):
        print(f'Solução {Xsn} alcançada com {i} iterações');break
    if abs(f(Xsn))<Err:
        print(f'Solução {Xsn} alcançada com {i} iterações e ...\
              \... tolerância {f(Xsn)}',f(Xsn)); break
    if i==imax:
        print(f'A solução não foi encontrada após {i} iterações'); break
    Xest=Xsn
print('Solução x=',format(Xest,'.4f'),'encontrada após',i+1,'iterações!')
print('Tempo de processamento computacional:%.4fs' %(time.process_time()-t0))
```

Fonte: AUTOR,(2024)

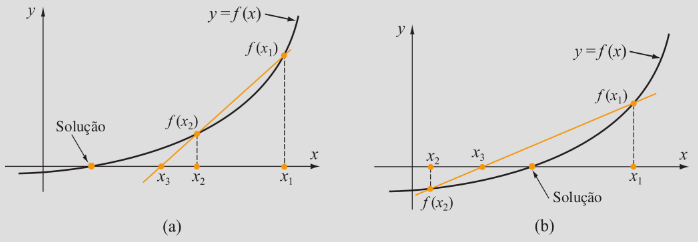
Organização

MN aplicados à Engenharia

Método da Secante

- Utiliza dois pontos na vizinhança da solução para determinar a nova solução estimada.

Figura: Método da Secante-Pontos de vizinhança



Fonte: GILAT,(2008)

- Em que $x_3 = x_2 - \frac{f(x_2)(x_1 - x_2)}{f(x_1) - f(x_2)}$
- Termo geral: $x_{k+1} = x_k - \dfrac{f(x_k)(x_{k-1} - x_k)}{f(x_{k-1}) - f(x_k)}$

MN aplicados à Engenharia

Método da Secante

Figura: *Script* Método da Secante no Octave/Matlab

```
clear all;clc;
%************************************%*******************
f=@(x)(8-4.5*(x-sin(x)));
Err=0.001; tol=Err; imax=30;
x1=2  ;     x2=3;     tic    % valores inicial
%************************************%*******************
for(i=1:imax)
    Xsn=x2-f(x2)*(x1-x2)/(f(x1)-f(x2));
  if (abs((Xsn-x2)/x2) < Err)             % Erro < x(i+1)-xi)/xi|
    Xsn = x2;
    fprintf("\nSoluçao %.4f alcançada após %d iterações! \n",Xsn,i)
  break
  end
  %************************************%*******************
  if (abs(f(Xsn))< Err)                            % Tol < f(xi)
  fprintf("\nSoluçao %.4f alcançada com %d iteraçoes e tolerancia %6.f! \n",
  Xsn,i,f(Xsn))
  break
  end
  %************************************%*******************
  if i==imax                             % Maximo de iteraçoes
    fprintf("\nSolução não alcançada com %d iteraç~oes\n", i)
    Xsn=('Sem resposta!')
  break
  end
%************************************%*******************
x1=x2;  x2=Xsn;
end
%************************************%*******************
fprintf("Tempo de processamento t=%f(s)\n\n",toc)
```

Fonte: AUTOR,(2020)

MN aplicados à Engenharia

Método da Secante

Figura: *Script* Método da Secante no Python

```
import numpy as np
import matplotlib.pyplot as plt, time

f= lambda x: 8-4.5*(x - np.sin(x))
Err,tol,x1,x2,imax = 0.001, 0.001, 2, 3, 50
print('Método da Secante!')
print('Intervalo de análise [%d,%d].\n'%(x1,x2) )
t0 = time.process_time()         #   Ligar cronômetro
# ===========================================================================
X=[]
for i in range(imax):
    Xsn=x2-f(x2)*(x1-x2)/( f(x1)-f(x2) )     # Xsn=x(i+1);Xest=x(i)
    X.append(Xsn)
    if(abs( (Xsn-x2) / x2)<Err):
        break
    if abs(f(Xsn))<Err:
        break
    if i==imax:
        print(f'A solução não foi encontrada após {i} iterações')
        break
    x1,x2=x2,Xsn
print('Solução x=',format(Xsn,'.4f'),'encontrada após',i+1,'iterações!')
print('Tempo de processamento computacional:%.4fs' %(time.process_time()-t0))
```

Fonte: AUTOR,(2024)

MN aplicados à Engenharia

Resultado
Método da Secante!
Intervalo de análise [2,3].

Solução x= 2.4305 encontrada após 3 iterações!
Tempo de processamento computacional: 0.0004s

Exercícios

- **Veja a lista de exercícios na web**
- **Veja a lista de códigos em:** `https://github.com/jonathacosta/NM`