INTRODUÇÃO À COMPUTAÇÃO CIENTÍFICA

Iº Trabalho: Zeros reais de funções reais

Tobias J. D. E. Rosa 2022 1º Período

Resumo

Este trabalho aborda o problema da determinação da largura de um galpão sabendo que duas vigas de comprimentos conhecidos tem suas extremidades inferiores apoiadas nos vértices chão-parede, enquanto que suas extremidades superiores estão apoiadas na parede oposta ao vértice de apoio. A posição em que as vigas se cruzam está localizada a uma altura conhecida. O problema é equacionado utilizando recursos simples de semelhança de triângulo, resultando em uma equação polinomial cuja solução não é trivial. Portanto, são propostas soluções pelos diferentes métodos numéricos da bissecção, posição falsa, Newton Raphson e da secante, ambos assistidos por computador utilizando programação em linguagem Fortran. Por fim, é feita uma abordagem comparativa entre os métodos utilizados, com relação aos parâmetros: precisão da resposta, tempo de processamento, convergência e número de iterações.

Solução do Problema

A informação da altura em que as vigas se cruzam é crucial para a resolução do problema, pois ela será utilizada para assemelhar triângulos retângulos, que são formados pelo seguimento de cada viga com a largura desconhecida e a altura da respectiva parede de apoio. Desta forma, aplicando a fórmula de Pitágoras em ambos os triângulos retângulos, são obtidas duas equações em função das alturas h_1 e h_2 em que as vigas estão apoiadas e a largura L, desconhecidas.

$$30^2 = L^2 + {h_1}^2 \text{ e } 20^2 = L^2 + {h_2}^2$$

Se h_1 e h_2 forem isolados nas equações anteriores, resulta em:

$$h_1 = \sqrt{30^2 - L^2} \text{ e } h_2 = \sqrt{20^2 - L^2}$$

A posição em que as vigas se cruzam está localizada a uma distância L_1 da parede esquerda e L_2 da parede direita, como apresentado no problema ilustrado na Fig 1.

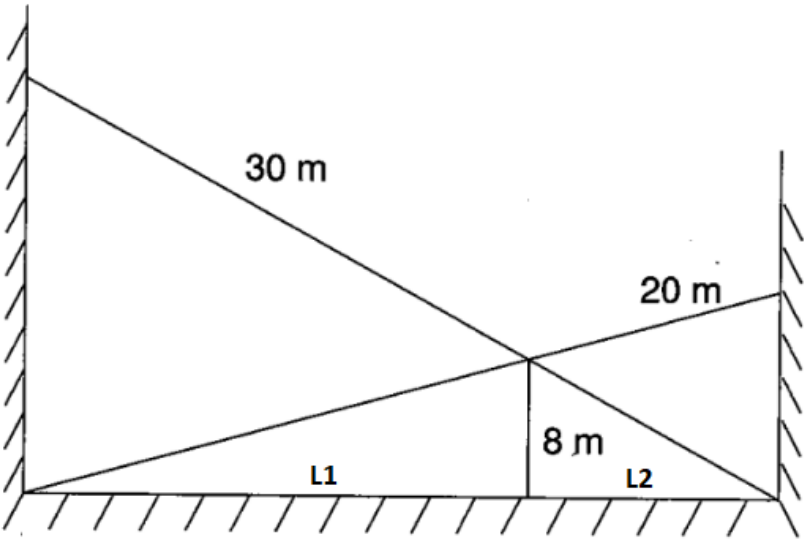

Figura 1 - Situação problema

Por semelhança de triângulos, sabemos que tanto h_1 quanto h_2 são projeções de um triângulo retângulo menor, consequentemente de altura menor $h = 8m$, desta forma:

$$L_1 = 8L/h_2 \text{ e } L_2 = 8L/h_1$$

Substituindo os valores de h_1 e h_2 obtidos anteriormente, chega-se a:

$$L_1 = \frac{8L}{\sqrt{30^2 - L^2}} \quad \text{e} \quad L_2 = \frac{8L}{\sqrt{20^2 - L^2}}$$

Assumindo que $L_1 + L_2 = L$, a equação da largura desconhecida é:

$$L = \frac{8L}{\sqrt{30^2 - L^2}} + \frac{8L}{\sqrt{20^2 - L^2}}$$

Dividindo ambos os lados por L e igualando a zero temos:

$$\frac{8}{\sqrt{30^2 - L^2}} + \frac{8}{\sqrt{20^2 - L^2}} - 1 = 0$$

Para encontrar as raízes reais de funções reais, uma vez conhecida a equação, é importante estudar o gráfico da função para compreender o domínio da função e consequentemente quais os intervalos válidos para aplicação dos métodos numéricos. O gráfico da função do problema a ser resolvido é ilustrado na Fig 2 .

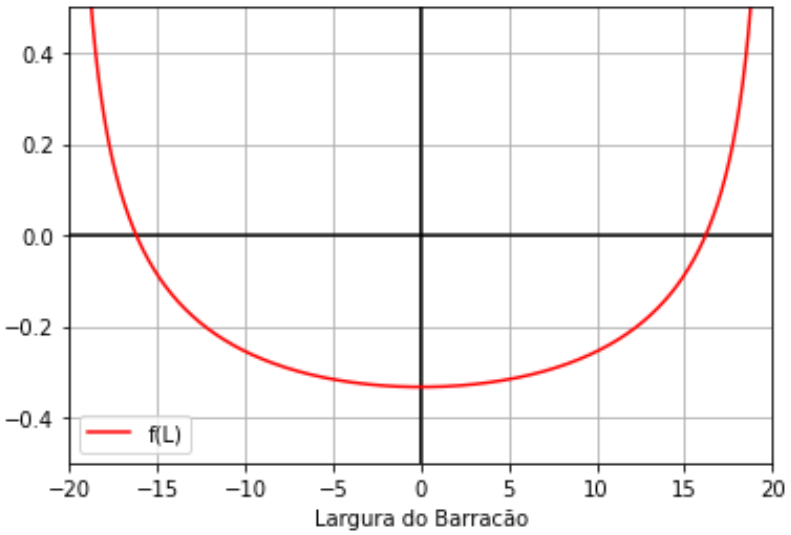

Figura 2 - Gráfico da função

É interessante observar que a função está definida apenas entre -20 e 20 e vale lembrar que não existe largura negativa, portanto o domínio será restringido à parte positiva. Se analisarmos o gráfico visualmente, percebemos que a única raiz positiva da função se encontra em um ponto aproximadamente médio entre 15 e 20, o que permite dizer, a grosso modo, que existe uma raiz próximo de 15, então sabemos que a largura do barracão é de aproximadamente 15m.

Para utilização dos métodos de Newton Raphson e da secante, é necessário estudar além da função da largura desconhecida, a sua derivada, dada pela equação:

$$\frac{8L}{\sqrt{(30^2 - L^2)^3}} + \frac{8L}{\sqrt{(20^2 - L^2)^3}} = 0$$

Quando a derivada da função estudada se aproxima de zero os métodos de Newton e da secante apresentam problemas de convergência. Como podemos observar no gráfico da derivada na Fig 3, a derivada se aproxima de zero quando modulo de L se aproxima de um número menor que 10.

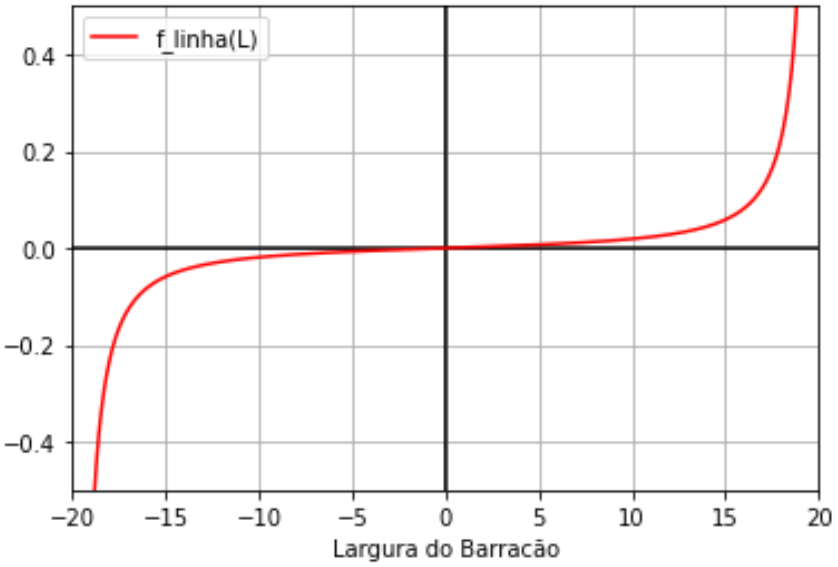

Figura 3 - Gráfico da derivada

Outra limitação dos métodos de Newton e da secante é quando a derivada é muito próxima ou igual a pŕopria função, o que acontece na região em que a derivada e a função se cruzam próximo à raiz negativa em L=-15 ou quando L se aproxima de 20, como ilustrado na Fig 4.

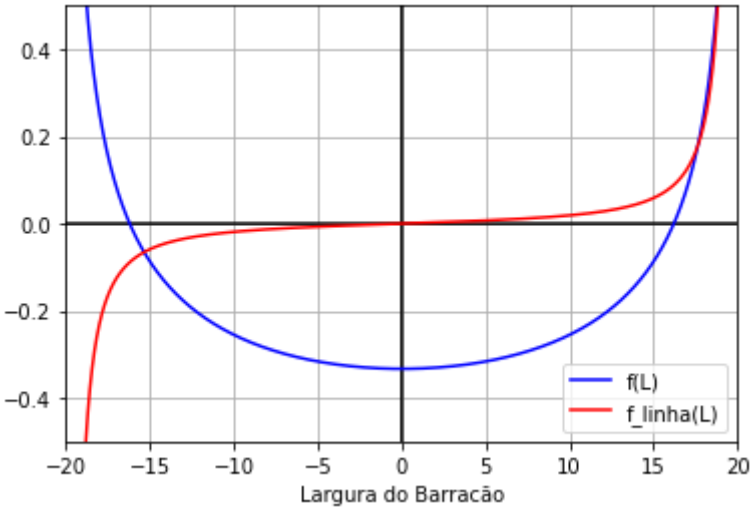

Figura 4 - Gráficos da função e derivada

Uma vez estudada a função e a derivada, já se conhece o seu domínio e o intervalo aproximado em que se encontra a raiz procurada, cabe então recorrer aos métodos numéricos.

O método numérico ideal deveria ter, convergência garantida, ordem de convergência alta e cálculos por iterações simples. Entretanto, o problema a ser resolvido é o principal fator na escolha do método, pois é a equação que define o melhor método. (RUGGIERO; LOPES, 2008).

Uma forma de avaliar o método numérico é através do diagnóstico do programa, onde podem ser medidos a precisão da raíz, o tempo de resposta e o número de iterações.

Apresentação e Discussão de Resultados

O intervalo inicial considerado para os métodos da bissecção e da posição falsa para essa função foi [a , b]=[0 , 19], pois existe uma assíntota em L=20. Já para os métodos de Newton Raphson e da secante, existe ainda um problema de divisão por zero quando a derivada se aproxima de zero. Na prática, isso acontece quando L fica aproximadamente menor que 12, e consequentemente a derivada em torno de 0,05. Por esse motivo, o intervalo foi reajustado para [a , b]=[12 , 19] para todos os métodos, de modo

que fosse possível comparar os parâmetros dos métodos utilizando um mesmo intervalo. No método de Newton Raphson foi utilizado o ponto médio entre 12 e 19 como estimativa inicial. A precisão utilizada para todos os métodos foi de $e = 0,001$.

Tabela 1 - Resultados amostrais

MÉTODO	ESTIMATIVA INICIAL (m)	RESULTADO (m)	TEMPO DE PROCESSAMENTO (s)	NÚMERO DE ITERAÇÕES
Bissecção	[a,b]=[12,19]	16,2121391	5.99999912E-05	17
Posição falsa	[a,b]=[12,19]	16,2116890	1.03000086E-04	18
Newton Raphson	x=(12+19)/2	16,2121277	2.39997171E-05	3
Secante	x1=12, x2=19	16,2131653	4.50001098E-05	34

Como apresentado na Tab 1, todos os métodos convergiram para uma largura do barracão de 16,21 m para o intervalo de [12, 19], entretanto, alguns métodos demandaram maior número de iterações para chegar a precisão requerida.

O método de Newton Raphson foi o que apresentou melhor tempo, como já era esperado, porém, vale lembrar que a estimativa inicial do método está considerando um ponto mais próximo da raíz que os demais métodos.

Uma surpresa foi o método da secante, pelo qual era esperado um número de iterações menor que os métodos da bissecção e da posição falsa. Isso se explica pelo fato da derivada da função estar muito próximo de zero. Quando aplicado no intervalo de [15, 19] o método convergiu em apenas 7 iterações para a mesma precisão anterior.

Mesmo que tenha sido apresentado apenas os dados relacionados ao intervalo de estimativa inicial de [2, 19], vale lembrar que os métodos da posição falsa e da bissecção convergiram para todo o intervalo inicial [0, 19]. Entretanto, apesar da convergência garantida e de um número de iterações bem menor que do método da secante, esses métodos foram os que apresentaram maior tempo de processamento, motivado provavelmente por maior esforço computacional dentro de cada iteração.

Conclusões

Todos os métodos se mostraram independentemente suficientes para solucionar o problema proposto apresentando um valor truncado comum de 16,21m para a precisão requerida. Embora alguns métodos possam convergir mais rapidamente em relação a outros, esse não é o único critério importante na escolha do método, pois a aplicabilidade do método depende também das características da função e de sua derivada. Neste problema, foi observado que o método de Newton raphson, mesmo sendo um método poderoso de rápida convergência, não pode ser aplicado estimando-se qualquer valor inicial aleatoriamente, pois exige o estudo da função e de sua derivada. Em contrapartida, os métodos da bissecção e da posição falsa aceitam qualquer estimativa inicial no intervalo estudado, mas exige maior esforço computacional e tem maior tempo de resposta. Portanto, cada problema deve ser analisado particularmente, para que seja escolhido o método que melhor se adeque às suas condições.

Referências

RUGGIERO, Marcia A. Gomes; LOPES, Vera Lucia da Rocha. *Cálculo numérico: aspectos teóricos e computacionais*. São Paulo: Pearson Makron Books, 2008.

CÓDIGOS-FONTE

bisection.f90

```
!gfortran bisection.f90
!./a.out input.txt
!Na primeira linha do arquivo input.txt deve ter o seguinte comando:12 19 0.001
!Instituto Militar de Engenharia - IME
!Programa utilizado na disciplina de introdução a computação científica
!Aluno: Tobias José Degli Esposte Rosa
Program bisection
   implicit none
   real,external::f
   character:: argument*20
   integer k
   real:: a,b,e,x,time0,time1,dt
   call get_command_argument(1, argument)
   open(1,file=argument,status='old')
   open(2,file='output.txt',status='unknown')
   read(1,*)a,b,e
   x=(a+b)/2
   CALL CPU_TIME(time0)
   IF (f(a)*f(b) > 0 ) THEN
      Print *,"f(a).f(b)>0, o intervalo não é valido!"
   ELSE
   k=0
      DO WHILE ((b-a) > e)
         IF   (f(a)*f(x) < 0) THEN
            b=x
         ELSE IF (f(b)*f(x) < 0) THEN
            a=x
         ENDIF
         x=(a+b)/2
         k = k+1
      END DO
   ENDIF
   CALL CPU_TIME(time1)
   dt=time1-time0
```

```
    write(2,*)"A raiz é:",x
    write(2,*)"O tempo é:",dt
    write(2,*)"O número de iterações é:",k

End Program bisection

real function f(x)
real::x
f=  8*(30**2-x**2)**(-0.5)+8*(20**2-x**2)**(-0.5)-1
end
```

posicao_falsa.f90

```
!gfortran posicao_falsa.f90
!./a.out input.txt
!Na primeira linha do arquivo input.txt deve ter o seguinte comando:12 19 0.001
!Instituto Militar de Engenharia - IME
!Programa utilizado na disciplina de introdução a computação científica
!Aluno: Tobias José Degli Esposte Rosa
program posicao_falsa
implicit none
character:: argument*20
real,external::f
integer::n,k
real::a,b,x,e1,e2,e,time0,time1,dt
call get_command_argument(1, argument)
open(1,file=argument,status='old')
open(2,file='output.txt',status='unknown')
read(1,*)a,b,e
e1=e
e2=e
n=0
k=0
CALL CPU_TIME(time0)
do while (n<1)
    if (b-a < e1) then
        x=(a+b)/2
        n=1
    elseif (((f(a))**2)**0.5 < e2) then
```

```
      x=a
      n=1
   elseif (((f(b))**2)**0.5 < e2) then
      x=b
      n=1
   endif
   x=(a*f(b)-b*f(a))/(f(b)-f(a))
   if (f(a)*f(x) < 0)  then
     b=x
   else
     a=x
   endif
   k=k+1
end do
CALL CPU_TIME(time1)
  dt=time1-time0
  write(2,*)"A raiz é:",x
  write(2,*)"O tempo é:",dt
  write(2,*)"O número de iterações é:",k
end program posicao_falsa

real function f(x)
real::x
f=  8*(30**2-x**2)**(-0.5)+8*(20**2-x**2)**(-0.5)-1
end
```

newton_raphson.f90

```
!gfortran newton_raphson.f90
!./a.out input.txt
!Na primeira linha do arquivo input.txt deve ter o seguinte comando:12 19 0.001
!Institudo Militar de Engenharia - IME
!Programa utilizado na disciplina de introdução a computação científica
!Aluno: Tobias José Degli Esposte Rosa
program newton_raphson
implicit none
character:: argument*20
real,external::f,f_1
```

```
integer:: k
real:: x,e,a,b,time0,time1,dt
call get_command_argument(1, argument)
open(1,file=argument,status='old')
open(2,file='output.txt',status='unknown')
read(1,*)a,b,e
x=(a+b)/2
k=0
CALL CPU_TIME(time0)
 do while (((f(x))**2)**0.5 > e)
   x = x-f(x)/f_l(x)
   k=k+1
 end do
 CALL CPU_TIME(time1)
   dt=time1-time0
   write(2,*)"A raiz é:",x
   write(2,*)"O tempo é:",dt
   write(2,*)"O número de iterações é:",k
end program newton_raphson
! Função
real function f(x)
real::x
f =  8*(30**2-x**2)**(-0.5)+8*(20**2-x**2)**(-0.5)-1
end
! Derivada
real function f_l(x)
real::x
f_l=  8.0*x*(400 - x**2)**(-1.5) + 8.0*x*(900 - x**2)**(-1.5)
end
```

secante.f90

```
!gfortran secante.f90
!./a.out input.txt
!Na primeira linha do arquivo input.txt deve ter o seguinte comando:12 19 0.001
!Institudo Militar de Engenharia - IME
!Programa utilizado na disciplina de introdução a computação científica
!Aluno: Tobias José Degli Esposte Rosa
```

```
program secante
implicit none
character:: argument*20
real,external::f,f_l
integer:: k
real:: x,x0,a,b,e,time0,time1,dt
call get_command_argument(1, argument)
open(1,file=argument,status='old')
open(2,file='output.txt',status='unknown')
read(1,*)a,b,e
x0=a
x=b
k=0
CALL CPU_TIME(time0)
 do while (((f(x))**2)**0.5 > e)
   x = x-f(x)/((f(x)-f(x0))/(x-x0))
   k=k+1
 end do
 CALL CPU_TIME(time1)
   dt=time1-time0
   write(2,*)"A raiz é:",x
   write(2,*)"O tempo é:",dt
   write(2,*)"O número de iterações é:",k
end program secante

real function f(x)
real::x
f=  8*(30**2-x**2)**(-0.5)+8*(20**2-x**2)**(-0.5)-1
end
```