MS211 - Cálculo Numérico - Turma J - Atividade 2

Luiz Henrique Costa Freitas RA: 202403
Gabriel de Freitas Garcia RA: 216179
Vanessa Vitória de Arruda Pachalki RA: 244956

13 de Outubro de 2020

Enunciado Atividade 2

Podemos utilizar os métodos para encontrar raízes de funções para estimar a raiz quadrada de um número. Considere a função $f(x) = x^2 - A$, onde A representa o número que queremos estimar a raiz quadrada. Vamos utilizar o método de Newton para tanto.

1. Escreva um pseudocódigo para o método de Newton e implemente como uma função do Octave.
2. Implemente a função $f(x)$ em Octave deixando A como um parâmetro de entrada.
3. Teste sua função para $A = 13$. Use $x_0 = 3$ como aproximação inicial e erro $\epsilon = 10^{-8}$
4. Teste sua função para um valor de A inteiro à sua escolha, desde que não tenha raiz inteira. Use o mesmo ϵ do item anterior. Como x_0 use um inteiro que deve ser a raiz quadrada de algum número menor que A.
5. Repita esse experimento com o método das secantes. Utilize x_1 um inteiro que é a raiz quadrada de um número maior que A.
6. Compare o comportamento dos dois métodos nos exemplos acima.

1 Respostas

Para resolver o item 1, desenvolvemos o seguinte algoritmo:

```
f = minha_funcao(x)
f_linha = derivada(f)
x = x0
while(f(x) > erro):
  x = x - f(x)/f_linha(x)
```

A partir deste pseudocódigo e assumindo as funções $f(x)$ e $f^{'}(x)$, o valor inicial x_0 e o erro ϵ anteriormente definidas podemos representar este mesmo algoritmo em Octave:

```
# função que calcula o loop do método de Newton
function res = newton_loop(x,A,f_linha,erro)
  k = 0;
  while(abs(f(x,A)) > erro)
    x = passo_newton(x,A,f_linha);
    k++;
    disp("Aproximação atual:");
    disp(x);
  endwhile
  res = x;
  disp("Número de passos:");
  disp(k);
endfunction

# função que calcula o passo de Newton

function x_res = passo_newton(x,A,f_linha)
  x_res = x - f(x,A)/f_linha(x);
endfunction
```

Para o item 2 definimos a função $f(x)$ desta maneira:

```
function res = f(x,A)
  res = x^2 - A;
endfunction
```

E com esta definição de função obtivemos o seguinte código para calcular o Método de Newton para a equação dada:

```
# Jesus quer transformar sua vida
# Arquivo: lab02.m
format long;
A = input("Digite um número inteiro:\n");
f_linha = @(x) 2*x;
x = input("Digite o valor inicial:\n");
erro = input("Digite o valor de erro: \n");
x = newton_loop(x,A,f_linha,erro);
disp("x final:");
disp(x);
disp("f(x) final: ");
disp(f(x,A));
```

Com este programa obtivemos as repostas para o item 3

```
>> lab02

Digite um número inteiro:
13
Digite o valor inicial:
3
Digite o valor de erro:
10^-8
Aproximação atual:
3.666666666666667
Aproximação atual:
3.606060606060606
Aproximação atual:
3.605551311433664
Aproximação atual:
3.605551275463990
Número de passos:
4
x final:
3.605551275463990
f(x) final:
1.776356839400250e-15
```

Semelhantemente obtivemos o resultado para o item 4:

```
>> lab02
Digite um número inteiro:
47
Digite o valor inicial:
6
Digite o valor de erro:
10^-8
Aproximação atual:
6.916666666666667
Aproximação atual:
6.855923694779117
Aproximação atual:
6.855654605682009
Aproximação atual:
6.855654600401045
Número de passos:
4
x final:
6.855654600401045
f(x) final:
7.105427357601002e-15
```

Para a resolução do item 5 repetimos o experimento e obtivemos o seguinte algoritmo:

```
x1 = algum_ponto
x2 = algum ponto_diferente
f = minha_funcao(x2)
g = (f(x2) - f(x1))/(x2-x1)
while(f(x2) > erro):
```

```
auxiliar = x2
x2 = x2 - f(x2)/g(x1,x2)
```

E o seguinte código em Octave:

```
# código principal

# Jesus quer transformar sua vida
# arquivo: lab02p2.m
format long;
A = input("Digite um número inteiro:\n");
g = @(x1,x2) (f(x2,A) - f(x1,A))/(x2-x1);
x1 = input("Digite o valor inicial:\n");
x2 = input("Digite o segundo valor inicial: \n");
erro = input("Digite o valor de erro: \n");
x2 = sec_loop(x1,x2,A,g,erro);
disp("x final:");
disp(x2);
disp("f(x) final: ");
disp(f(x2,A));

#função que executa o loop do método das secantes

function res = sec_loop(x1,x2,A,g,erro)
  k = 0;
  while(abs(f(x2,A)) > erro)
    aux = x2;
    x2 = passo_sec(x1,x2,A,g);
    x1 = aux;
    k++;
    disp("Aproximação atual:");
    disp(x2);
  endwhile
  res = x2;
  disp("Número de passos:");
  disp(k);
endfunction

# função que calcula o passo do método das secantes

function x_res = passo_sec(x1,x2,A,g)
  x_res = x2 - f(x2,A)/g(x1,x2);
endfunction
```

E obtivemos os seguintes resultados:

```
>> lab02p2

Digite um número inteiro:
13
Digite o valor inicial:
3
Digite o segundo valor inicial:
4
Digite o valor de erro:
10^-8
Aproximação atual:
3.571428571428572
Aproximação atual:
3.603773584905661
Aproximação atual:
3.605559729526671
Aproximação atual:
3.605551273379371
Aproximação atual:
3.605551275463987
Número de passos:
5
x final:
3.605551275463987
f(x) final:
-1.776356839400250e-14
>> lab02p2

Digite um número inteiro:
```

```
47
Digite o valor inicial:
6
Digite o segundo valor inicial:
7
Digite o valor de erro:
10^-8
Aproximação atual:
6.846153846153846
Aproximação atual:
6.855555555555555
Aproximação atual:
6.855654669078660
Aproximação atual:
6.855654600400548
Número de passos:
4
x final:
6.855654600400548
f(x) final:
-6.799893981224159e-12
```

Os dois métodos são muito parecidos entre si, porém o Método de Newton trabalha com a derivada da função enquanto o Método das Secantes utiliza a secante da função. Contudo, quanto menor a distância dos dois pontos x no segundo método, mais ele se aproxima do valor da derivada, deixando os métodos bastante semelhantes. O método de Newton, portanto, pode ser entendido como um caso particular do Método das Secantes em que a distância dos dois pontos que cruzam a função é praticamente zero, ou seja, a derivada da função no ponto.

O primeiro método (Newton) costuma ser o mais eficiente e isso foi observado no item 3 e 4. No item 3, isso pode ser observado no número de iterações que cada método precisou para chegar no resultado final (Newton = 4 e Secantes =5). Já no item 4, percebe-se que o número de iterações foi a mesma, entretanto, o erro apresentado no segundo método foi maior (na ordem de 10^3 a mais que o primeiro).