Análise Numérica 2023

Resolução da Lista de Exercícios 4

Patrick Duarte Pimenta

5.1.2.

"x1 = "

0.9512725
1.3218976
1.4461858

"x2 = "

-0.5943413
-2.0172002
-3.0083344

"x3 = "

-1.8569877
-3.4316748
-4.0481415

"x4 = "

2.0341596

4.7088105

5.9709026

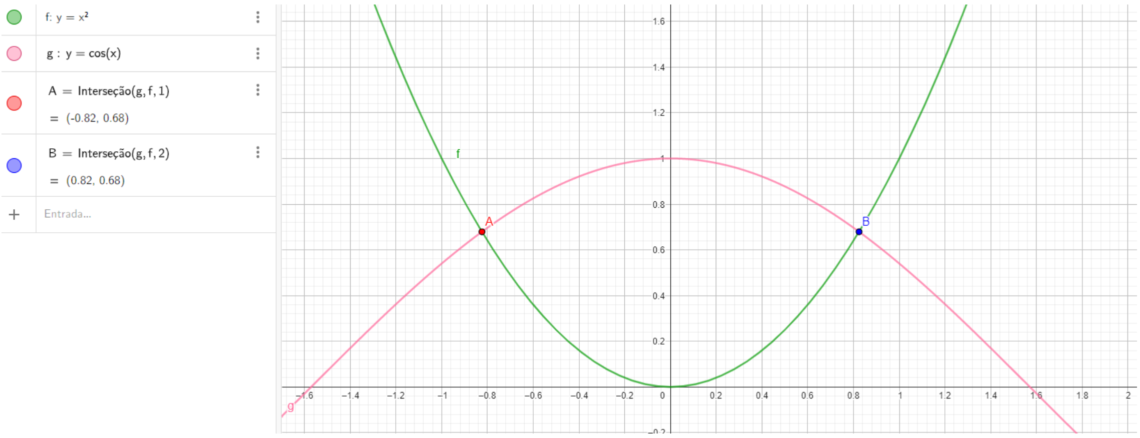
Algoritmo:

```
===============================================================
clc

/*
As funções fun e jac são definidas para calcular o vetor das três
equações não-lineares
e a matriz jacobiana, respectivamente. O loop for é usado para iterar o
processo do métod
de Newton-Raphson 10 vezes. Dentro do loop, a atualização das variáveis
é calculada pela
solução do sistema linear - j\f, onde j é a matriz jacobiana e f é o
vetor das equações
não-lineares.
*/

// Calcula o vetor das três equações não-lineares
function [f]=fun(x)
    f(1) = 2*x(1) - x(2) - cos(x(1));
    f(2) = -x(1) + 2*x(2) - x(3) - cos(x(2));
    f(3) = -x(2) + x(3) - cos(x(3));
endfunction

// Calcula a matriz jacobiana das três equações
function [j]=jac(x)
    j(1,1) = 2 + sin(x(1));
    j(1,2) = -1;
    j(1,3) = 0;
    j(2,1) = -1;
    j(2,2) = 2 + sin(x(2));
    j(2,3) = -1;
    j(3,1) = 0;
    j(3,2) = -1;
    j(3,3) = 1 + sin(x(3));
endfunction

// Calcula a atualização das variáveis pelo métod de Newton-Raphson
function [x]=newtonRaphson(x0, n)
    x = x0
    k = 1;
```

```
    while(k <= n)
        f = fun(x);
        j = jac(x);

        dx = - j \ f;
        x = x + dx;

        k = k + 1;
    end
endfunction

// Aproximações iniciais
x01 = [ 1; 1; 1];
x02 = [-0.5; -2; -3];
x03 = [-2; -3; -4];
x04 = [0; 0; 0];

n = 10;

// Imprimindo resultados

x1 = newtonRaphson(x01, n);
disp("x1 = ");
disp(x1);

x2 = newtonRaphson(x02, n);
disp("x2 = ");
disp(x2);

x3 = newtonRaphson(x03, n);
disp("x3 = ");
disp(x3);

x4 = newtonRaphson(x04, n);
disp("x4 = ");
disp(x4);

=======================================================================
```

5.1.4.

a)



Podemos ver que há dois pontos de intersecção, um no primeiro quadrante (aproximadamente em (0.82,0.68)) e outro no segundo quadrante (aproximadamente em (-0.82,0.68)).

b)

Estimativa:

- No ponto de intersecção do primeiro quadrante, a coordenada x é cerca de 0.82 e a coordenada y é cerca de 0.68.
- No ponto de intersecção do segundo quadrante, a coordenada x é cerca de -0.82 e a coordenada y é cerca de 0.68.

c)

Podemos escrever as equações das curvas na forma $f(x,y) = 0$:

Para a parábola $y = x^2$, temos $f(x,y) = y - x^2 = 0$.

Para a curva $y = \cos(x)$, temos $f(x,y) = y - \cos(x) = 0$.

Assim, podemos escrever o problema como:

$$f([x\ y]^T) = [y - x^2,\ y - \cos(x)]^T = [0,\ 0]^T$$

onde [x y]T é o vetor de coordenadas do ponto de intersecção das duas curvas.

d)

Para encontrar a jacobiana Jf da função f([x y]T) = [y - x^2, y - cos(x)]T, que relaciona a variação das duas funções em relação às variações de x e y, vamos calcular as derivadas parciais das duas funções em relação a x e y:

Jf(x,y) =

| ∂f1/∂x ∂f1/∂y |

| ∂f2/∂x ∂f2/∂y |

Onde f1 = y - x^2 e f2 = y - cos(x).

Assim, temos:

∂f1/∂x = -2x

∂f1/∂y = 1

∂f2/∂x = sen(x)

∂f2/∂y = 1

Portanto, a jacobiana Jf([x y]T) é dada por:

Jf(x,y) =

| -2x 1 |

| sen(x) 1 |

Observe que a jacobiana Jf é uma matriz 2x2, que relaciona as variações de x e y com as variações das funções f1 e f2.

e)

Para construir a iteração do método de Newton, vamos utilizar a fórmula:

$$x_{k+1} = x_k - Jf(x_k)^{-1} * f(x_k)$$

onde x_{k+1} é a aproximação da solução na (k+1)-ésima iteração, x_k é a aproximação na k-ésima iteração, $Jf(x_k)$ é a jacobiana da função f calculada em x_k e $f(x_k)$ é o vetor de valores das funções f1 e f2 calculado em x_k.

Para o nosso problema, temos:

A função $f([x\ y]T) = [y - x^2, y - \cos(x)]T$

A jacobiana Jf(x,y) =

| -2x 1 |

| sen(x) 1 |

Assim, a iteração do método de Newton é dada por:

$$[x_{k+1}\ y_{k+1}]T = [x_k\ y_k]T - Jf(x_k,y_k)^{-1} * f([x_k\ y_k]T)$$

onde:

$Jf(x_k,y_k)^{-1}$ é a inversa da matriz jacobiana $Jf(x_k,y_k)$, que pode ser calculada facilmente utilizando álgebra linear.

f([x_k y_k]T) é o vetor de valores das funções f1 e f2 calculado em [x_k y_k]T.

Assim, a iteração fica:

x_{k+1} = x_k - (-2x_k * (y_k - x_k^2) + (y_k - cos(x_k)))

y_{k+1} = y_k - (sen(x_k) * (y_k - cos(x_k)) + (y_k - x_k^2))

Essa iteração deve ser aplicada repetidamente até que se obtenha uma aproximação suficientemente precisa da solução.

f)

"Os pontos de interseção são:"

"(0.8241323, 0.6791941)"

"(-0.6767798, 0.1438928)"

Algoritmo:

```
clc

// Definindo a função e a jacobiana
function [f, Jf]=funcao(x, y)
    f = [y - x^2; y - cos(x)];
    Jf = [-2*x, 1; cos(x), 1];
endfunction

// Definindo a função do método de Newton
function [x, y]=newton(x0, y0, tol)
    iter = 0;
    [f, Jf] = funcao(x0, y0);
    while norm(f, 1) > tol && iter < 100
        dx = Jf \ (-f);
        x = x0 + dx(1);
```

```
        y = y0 + dx(2);
        [f, Jf] = funcao(x, y);
        x0 = x;
        y0 = y;
        iter = iter + 1;
    end
endfunction

// Chamando o método de Newton para as duas interseções
[x1, y1] = newton(0.5, 0.5, 1e-8);
[x2, y2] = newton(-0.5, 0.5, 1e-8);

// Mostrando os resultados
disp("Os pontos de interseção são:");
disp(["(" + string(x1) + ", " + string(y1) + ")"; "(" + string(x2) + ", " + string(y2) + ")"]);
```

---

g)

Podemos transformar o sistema em um problema de uma única variável, por exemplo, eliminando a variável y de uma das equações e substituindo na outra. Para isso, vamos isolar y na equação da curva y = cos(x):

y = cos(x);

Em seguida, substituindo y na equação da parábola y = x^2, temos:

x^2 = cos(x)

Portanto, temos agora um problema de uma única variável x que pode ser resolvido por métodos numéricos, como o método da bisseção, método da falsa posição, método de Newton, entre outros.

5.1.7.

Primeiramente, vamos determinar os pontos de máximo da função f(x) no primeiro quadrante. Observando que a função f(x) é par, temos que ela tem um ponto máximo em x = 0. Além disso, a função apresenta outros pontos máximos no primeiro quadrante. Para encontrá-los, devemos procurar as raízes da derivada da função f(x):

$f'(x) = (x \cos(x) - \text{sen}(x)) / x^2$

Igualando a derivada a zero, temos:

$x \cos(x) - \text{sen}(x) = 0$

$\sin(x)/x = \cos(x)$

$\tan(x) = x$

Podemos encontrar as soluções dessa equação por meio de métodos numéricos ou por meio de uma análise gráfica. No entanto, podemos utilizar o fato de que a primeira solução é próxima a 2.5, e a segunda é próxima a 4.5, para estimar os pontos máximos próximos a esses valores.

Assim, podemos escolher dois pontos próximos a x = 2.5 e x = 4.5 na curva y = f(x) para determinar a equação da reta tangente a essa curva nesses pontos. Escolhendo, por exemplo, os pontos (2.5, f(2.5)) e (4.5, f(4.5)), temos:

$f(2.5) = \text{sen}(2.5) / 2.5 + 1 \approx 1.128$

$f(4.5) = \text{sen}(4.5) / 4.5 + 1 \approx 0.687$

Para determinar a equação da reta tangente a esses pontos, precisamos determinar a inclinação da reta em cada um deles. A inclinação da reta tangente a um ponto na curva y = f(x) é dada pela derivada da função f(x) nesse ponto. Assim, temos:

$f'(2.5) = (2.5 \cos(2.5) - sen(2.5)) / 2.5^2 \approx -0.287$

$f'(4.5) = (4.5 \cos(4.5) - sen(4.5)) / 4.5^2 \approx 0.052$

A equação da reta tangente ao ponto (2.5, f(2.5)) é dada por:

$y - 1.128 = -0.287(x - 2.5)$

Simplificando, obtemos:

$y = -0.287x + 1.954$

A equação da reta tangente ao ponto (4.5, f(4.5)) é dada por:

$y - 0.687 = 0.052(x - 4.5)$

Simplificando, obtemos:

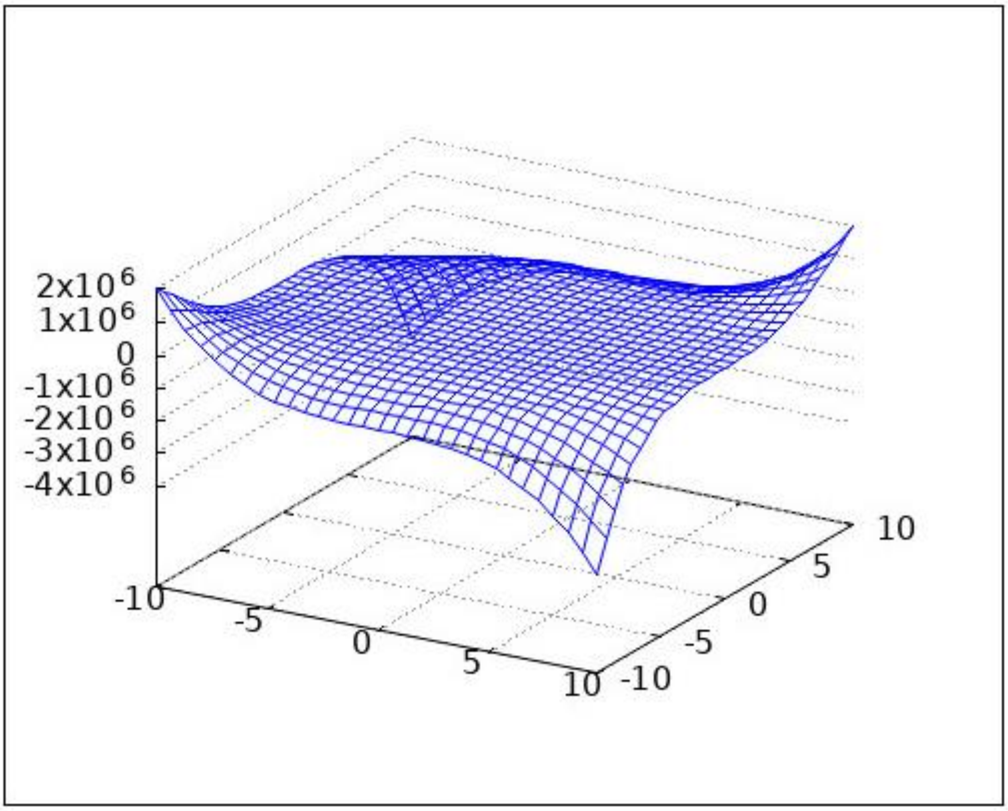
$y = 0.052x - 0.052$

Portanto, a equação das retas tangentes aos pontos próximos aos primeiros dois máximos da função f(x) no primeiro quadrante são:

$y = -0.287x + 1.954$ e $y = 0.052x - 0.052$

5.1.14

a)



b)

Para (x, y) = (-10, -10), o valor aproximado de um ponto de máximo é de 1990010

c)

Quando a função atinge seu valor máximo, o gradiente é nulo. Portanto, precisamos encontrar o gradiente da função e resolvê-lo para obter os valores de x e y. O gradiente da função é dado por:

grad f(x,y) = [∂f/∂x, ∂f/∂y]

Onde

$\partial f/\partial x = -4x^3 + 3y^3 - 1$

$\partial f/\partial y = -6y^5 + 9xy^2$

Então, para encontrar o valor máximo da função, precisamos resolver o seguinte sistema de equações não lineares:

$$-4x^3 + 3y^3 - 1 = 0$$

$$-6y^5 + 9xy^2 = 0$$

d)

"Não foi possível convergir."

Algoritmo:

```
=======================================================================
clc

// Definição da função e sua derivada
function [f, df]=fun(x, y)
    f = -x^4 - y^6 + 3*x*y^3 - x;
    df = [-4*x^3 + 3*y^3 - 1; -6*y^5 + 9*x*y^2];
endfunction

// Definição do ponto de partida
x0 = -1.5;
y0 = -1.5;

// Definição do critério de parada
max_iter = 1000;
tol = 1e-6;

// Iteração do método de Newton
for i = 1:max_iter
    [f_val, df_val] = fun(x0, y0);
    dx = -df_val(1)/df_val(2);
    dy = -f_val/df_val(2);
    x1 = x0 + dx;
    y1 = y0 + dy;
    if norm([dx; dy]) < tol
        break;
    end
    x0 = x1;
    y0 = y1;
```

```
end

// Resultado
if i == max_iter
    disp("Não foi possível convergir.");
else
    disp(["Máximo encontrado em (", x1, ", ", y1, ") com valor de ", -
f_val]);
end
```
===