Daniel Cipriano RA: 233228 MS211-C Atividade 6

i)

calculando a matriz jacobiana do sistema:

$$J_{(x)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{bmatrix}$$

$\frac{\partial f_1}{\partial x_1} = 3$; $\frac{\partial f_1}{\partial x_2} = x_3 \operatorname{sen}(x_2 x_3)$ (pela regra da cadeia) ; $\frac{\partial f_1}{\partial x_3} = x_2 \operatorname{sen}(x_2 x_3)$ (pela regra da cadeia)

$\frac{\partial f_2}{x_1} = 2x_1$; $\frac{\partial f_2}{\partial x_2} = -162(x_2 + 0{,}1)$ (pela regra da cadeia) ; $\frac{\partial f_2}{\partial x_3} = \cos(x_3)$

$\frac{\partial f_3}{\partial x_1} = -x_2 \cdot e^{-x_1 x_2}$ (regra da cadeia) ; $\frac{\partial f_3}{\partial x_2} = -x_1 \cdot e^{-x_1 x_2}$ (regra da cadeia) ; $\frac{\partial f_3}{\partial x_3} = 20$

$$J_{(x)} = \begin{bmatrix} 3 & x_3 \operatorname{sen}(x_2 x_3) & x_2 \operatorname{sen}(x_2 x_3) \\ 2x_1 & -162(x_2 + 0{,}1) & \cos(x_3) \\ -x_2 e^{-x_1 x_2} & -x_1 e^{-x_1 x_2} & 20 \end{bmatrix}$$

//

ii) Tabela de resultados para o método de Newton com erro tolerado de 1e-5

k (iteração)	x(k)	s(k)	Dr (|| x(k+1) - x(k)||)∞	F(x)
1	(0,50022; 0,01949; -0,52356)	(0,40022; -0,08051; -0,62152)	0,62152	(0,0007; -0,34448; 0,0319)
2	(0,50001; 0,00159; -0,52356)	(-0,0002; 0,0179; -0,00203)	0,0179	(4,3159e-5; 0,02595; 3,6209e-5)
3	(0,5; 1,24977e-5; -0,5236)	(-1,41566e-5; -0,00158; -4,12686e-5)	0,00158	(3,41872e-7; -0,0002; 2,89359e-7)
4	(0,5; 7,82392e-10; -0,5236)	(-1,13943e-7; -1,24969e-5; -3,26888e-7)	1,24969e-5	(2,14078e-11; -1,26499e-8; 1,80957e-11)

Função de iteração:

```
function res = metodo_newton(x, F, J);

  e = 1e-5; # erro permitido
  x_aux = 0; # marca x na iteração k-1
  k = 0; # contador de iterações
  dr = e + 1; # distância relativa
  s = 0; # passo de newton

  while (modulo_infinito(F, 3) > e && dr > e)
    k++;
    s = inv(-J)*F;
    x_aux = x;
    x = x + s;
    dr = modulo_infinito((x - x_aux), 3);

    # atualiza valores de F e J com base em x
    F = [f1(x(1), x(2), x(3)); f2(x(1), x(2), x(3)); f3(x(1), x(2), x(3))];
    J = jacobiana(x(1), x(2), x(3));
  endwhile

  res = x;
end
```