Disciplina Métodos Computacionais

Professor Dr. Wanderlei Malaquias Pereira Junior

Exemplo 021 - *Resolução de sistemas não lineares: Newton - Raphson*

Determinando as derivadas com auxílio de ferramentas computacionais

```
import sympy as sy
import numpy as np
sy.init_printing(pretty_print = False)
```

Vamos aplicar os conceitos do Método de Newton-Raphson nas equações abaixo.
Primeiramente deveremos determinar a primeira derivada das funções:

$$x_1^2 + x_1 . x_2 = 10$$
$$x_2 + 3.x_1 . x_2^2 = 57$$

```
# Declaração do símbolo
X_0 = sy.Symbol('X_0')
X_1 = sy.Symbol('X_1')
# Declaração das funções
F_0 = X_0 ** 2 + X_0 * X_1 - 10
F_1 = X_1 + 3 * X_0 * X_1 ** 2 - 57
```

Derivandos das funções $f(x_1, x_2)$ e escrevendo a matriz jacobiana da função.

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{bmatrix}$$

```
sy.diff(F_0, X_0)
```

```
sy.diff(F_0, X_1)
```

```
sy.diff(F_1, X_0)
```

```
sy.diff(F_1, X_1)
```

Montagem da matriz Jacobiana e matriz da função

```
# FUNÇÃO PARA DECLARAÇÃO DA EQUAÇÃO QUE DESEJA-SE ENCONTRAR A RAÍZ
def F E JACOBIANA(X):
```

```
    X_0 = X[0]
    X_1 = X[1]
    J = np.array([[2*X_0 + X_1, X_0], [3*X_1**2, 6*X_0*X_1 + 1]])
    F = np.array([[X_0**2+X_0*X_1-10], [X_1 + 3*X_0*X_1**2 - 57]])
    return J, F

# FUNÇÃO DE DETERMINAÇÃO DO ERRO
def ERRO_NORMA_MAX(X, X_NEW, D):
    AUX_0 = [0] * D
    AUX_1 = [0] * D
    AUX_2 = [0] * D
    for I_CONT in range(D):
        AUX_0[I_CONT] = abs(X_NEW[I_CONT])
        AUX_1[I_CONT] = abs(X[I_CONT])
        AUX_2[I_CONT] = AUX_0[I_CONT] - AUX_1[I_CONT]
    ERRO = max(AUX_2) / max(AUX_0)
    return ERRO


# TOLERÂNCIA DO PROCESSO ITERATIVO
TOL = 1E-8
# CHUTE INICIAL
X_0 = [1.50, 3.50]


def NEWTON_RAPHSON(FUNCOES, TOL, X_0, D):
    ERRO = 1000
    I = 0
    RESULTADOS = []
    X_NEW = [0] * D
    X = X_0.copy()
    while ERRO > TOL:
            print('iteração: ', I)
            print('x velho: ', X)
            F_LINHAX, FX = FUNCOES(X)
            print(np.linalg.inv(F_LINHAX))
            AUX = np.dot(np.linalg.inv(F_LINHAX), FX)
            S = - AUX.flatten()
            print('S: ', S)
            for I_CONT in range(D):
                X_NEW[I_CONT] = X[I_CONT] + S[I_CONT]
            print('x novo: ', X_NEW)
            ERRO = ERRO_NORMA_MAX(X, X_NEW, D)
            print('erro: ', ERRO)
            I += 1
            X = X_NEW.copy()
    return X


X = NEWTON_RAPHSON(F_E_JACOBIANA, TOL, X_0, 2)


X
```

```
     [2.0, 3.0]
```