Pregunta 2 - Dirigida 6

June 6, 2019

1 Pregunta 2

Aplique los métodos anteriores a estos casos

a) $$e^x 3x^2 = 0 $$

b) $x^3 = x^2 + x + 1$

c) $x^210x + 23 = 0$

d) $e^x2x = 0$

e) $cos(x) + 1x = 0$

f) $ln(x)5 + x = 0$

g) $e^x = 1/(0.1 + x^2)$

Importando las librerias

```
In [1]: #!/usr/bin/env python3
        # -*- coding: utf-8 -*-

        import __metodos_No_Lineales as  mnl
        import numpy as np
        import pandas as pd
```

```
toolNick se ha importado correctamente.
__metodos_No_Lineales se ha importado correctamente.
```

a) Metodo Biseccion : $e^x 3x^2 = 0 $

```
In [2]: f = lambda x :  np.e ** x - 3 * x ** 2
        x = mnl.solve_Biseccion(f, a_0=0, b_0=1,v='True')
```

```
Data Frame - Metodo Biseccion

              a         c         b      f(c)
```

```
k_0   0.000000  0.500000  1.000000  0.500000
k_1   0.500000  0.750000  1.000000  0.250000
k_2   0.750000  0.875000  1.000000  0.125000
k_3   0.875000  0.937500  1.000000  0.062500
k_4   0.875000  0.906250  0.937500  0.031250
k_5   0.906250  0.921875  0.937500  0.015625
k_6   0.906250  0.914062  0.921875  0.007812
k_7   0.906250  0.910156  0.914062  0.003906
k_8   0.906250  0.908203  0.910156  0.001953
k_9   0.908203  0.909180  0.910156  0.000977
k_10  0.909180  0.909668  0.910156  0.000488
k_11  0.909668  0.909912  0.910156  0.000244
k_12  0.909912  0.910034  0.910156  0.000122
k_13  0.909912  0.909973  0.910034  0.000061
k_14  0.909973  0.910004  0.910034  0.000031
k_15  0.910004  0.910019  0.910034  0.000015
k_16  0.910004  0.910011  0.910019  0.000008
k_17  0.910004  0.910007  0.910011  0.000004
k_18  0.910007  0.910009  0.910011  0.000002
```

b) Metodo Newton : $x^3 = x^2 + x + 1$

```
In [3]: f = lambda x : x**3 - x**2 - x - 1
        df = lambda x: 3*x**2 - 2*x - 1
        x = mnl.solve_Newton(f, df, 3, v = 'True')
```

```
             x          f(x)      f'(x)
k_0  3.000000  1.400000e+01  20.000000
k_1  2.300000  3.577000e+00  10.270000
k_2  1.951704  6.734778e-01   6.524037
k_3  1.848474  5.063832e-02   5.553618
k_4  1.839356  3.771453e-04   5.470977
k_5  1.839287  2.147008e-08   5.470354

Converge en iter:5
Resultado:1.8392867552141612
```

c) Metodo Secante: $x^210x + 23 = 0$

```
In [5]: f = lambda x : x**2 - 10*x + 23
        x = mnl.solve_Secante(f, 10, v = 'True')
```

```
Data Frame - Metodo Secante

             x          f(x)
```

```
k_0       10  2.300000e+01
k_1  7.72277  5.413489e+00
k_2  7.02179  2.087655e+00
k_3  6.58179  5.020450e-01
k_4  6.44247  8.071103e-02
k_5  6.41578  4.430365e-03
k_6  6.41423  4.376966e-05
k_7  6.41421  2.421238e-08

Converge en iter:7
Resultado:6.414213562373141
```

d) Metodo Posicion Falsa $e^x 2x = 0$

```
In [7]: f = lambda x : np.e**x - 2 - x
        x = mnl.solve_ReguleFalsi(f, -4, 4, v='True')
```

```
Data Frame - Metodo Regular falsi

      a         c         b      f(c)
k_0 -4 -4.346642  4.000000  2.359592
k_1 -4 -4.346642 -4.346642  2.359592
k_2 -4 -1.949953 -1.949953  0.092234
k_3 -4 -1.851783 -1.851783  0.008740
k_4 -4 -1.842440 -1.842440  0.000870
k_5 -4 -1.841509 -1.841509  0.000087
k_6 -4 -1.841416 -1.841416  0.000009

Converge en iter:7
Resultado:-1.841406697768061
```

e) Metodo Biseccion $cos(x) + 1x = 0$

```
In [12]: f = lambda x : np.cos(x) + 1 - x
         x = mnl.solve_Biseccion(f,a_0=-3,b_0=3,v='True')
         print("\n\nResultado:{}".format(x))
```

```
Data Frame - Metodo Biseccion

              a         c         b      f(c)
k_0  -3.000000  0.000000  3.000000  3.000000
k_1   0.000000  1.500000  3.000000  1.500000
k_2   0.000000  0.750000  1.500000  0.750000
k_3   0.750000  1.125000  1.500000  0.375000
```

```
k_4   1.125000  1.312500  1.500000  0.187500
k_5   1.125000  1.218750  1.312500  0.093750
k_6   1.218750  1.265625  1.312500  0.046875
k_7   1.265625  1.289062  1.312500  0.023438
k_8   1.265625  1.277344  1.289062  0.011719
k_9   1.277344  1.283203  1.289062  0.005859
k_10  1.283203  1.286133  1.289062  0.002930
k_11  1.283203  1.284668  1.286133  0.001465
k_12  1.283203  1.283936  1.284668  0.000732
k_13  1.283203  1.283569  1.283936  0.000366
k_14  1.283203  1.283386  1.283569  0.000183
k_15  1.283386  1.283478  1.283569  0.000092
k_16  1.283386  1.283432  1.283478  0.000046
k_17  1.283386  1.283409  1.283432  0.000023
k_18  1.283409  1.283421  1.283432  0.000011
k_19  1.283421  1.283426  1.283432  0.000006
k_20  1.283426  1.283429  1.283432  0.000003
k_21  1.283426  1.283428  1.283429  0.000001


Resultado:1.2834277153
```

f) Metodo Punto Fijo $ln(x)5 + x = 0$

```
In [17]: f = lambda x : x - (np.log(x) - 5 + x)
         g = lambda x : np.log(x) - 5 + x
         x = mnl.solve_PuntoFijo(f,g,x_0=1,max_iter=300,v='True',graphic='True')

Max. iteraciones alcanzado.
Resultado:nan


C:\Users\ADMIN\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: RuntimeWarning: invalid va
```

g) Metodo Secante $e^x = 1/(0.1 + x^2)$

```
In [19]: f = lambda x : np.e**x - (1/(0.1+x**2))
         x = mnl.solve_Secante(f, x_0=5, v='True')


Data Frame - Metodo Secante

              x          f(x)
k_0           5  1.483733e+02
k_1     4.02515  5.592757e+01
k_2     3.43539  3.095962e+01
```

```
k_3    2.70411  1.480605e+01
k_4    2.03382  7.407203e+00
k_5    1.36278  3.396109e+00
k_6   0.794626  8.464315e-01
k_7   0.606012 -3.070701e-01
k_8   0.656223  4.293970e-02
k_9   0.650063  2.085155e-03
k_10  0.649749 -1.456872e-05
k_11  0.649751  4.925255e-09

Converge en iter:11
Resultado:0.6497506818006871
```