Facultad de Matemática, Astronomía, Física y Computación, U.N.C.

Computación

Aula Virtual: https://famaf.aulavirtual.unc.edu.ar/course/view.php?id=747

Resguardo tutoriales: https://www.famaf.unc.edu.ar/~moreschi/docencia/Computacion/

Tutorial Problema 14 de la Guía N° 3

Problema 14:

Sea $f(x)$ una función continuamente diferenciable. Escriba un programa en PYTHON, en forma de script, que implemente el método de Newton para hallar raíces simples de f. La iteración de Newton debe detenerse cuando $|x_n - x_{n-1}| < \delta$ y $|f(x_n)| < \varepsilon$. Si estas tolerancias no son alcanzadas al cabo de M iteraciones, el programa debe detenerse indicando que no se alcanzó la tolerancia deseada. Si el programa alcanza las tolerancias deseadas, al detenerse debe imprimir en pantalla el valor de la aproximación x_n, el valor de $|f(x_n)|$ y el número de iteraciones utilizadas.

Usando $M = 20$, $\delta = \varepsilon = 10^{-6}$, halle las raíces de

a) $f(x) = 3x - \tan(x)$, usando $x_0 = 1.5$.

b) $f(x) = x^6 - 5x^5 + x^4 - 3x^3 + x^2 - x + 1$, usando $x_0 = 0$.

Tutorial:

- Guarde en el archivo p14-g3.py las siguientes instrucciones:

```
"""
 En esta versión usa la función para el método
 de Newton.
"""
import numpy as np
import matplotlib.pyplot as plt

#-------------------------------------
# Función parte a), cuya raíz queremos hallar
def f1(x):
    return 3*x - np.tan(x)

def df1(x):
    return 3 - 1/np.cos(x)**2

# Función parte b), cuya raíz queremos hallar
def f2(x):
    return x**6 - 5*x**5 + x**4 - 3*x**3 + x**2 - x + 1

def df2(x):
    return 6*x**5 - 25*x**4 + 4*x**3 - 9*x**2 + 2*x - 1

def newton(f,df,xini,tolx,toly,maxn):
    print('------')
    xprev = xini
    x = xprev - f(xprev)/(df(xprev))
```

```
    intervalo = x - xprev
    n = 1
    while ((abs(intervalo) > tolx) or ( abs(f(x)) > toly) and (n<=maxn)) :
        n += 1
        xprev = x
        x = xprev - f(xprev)/(df(xprev))
        intervalo = x - xprev
    print("\nLa solución es: %27.20e" % (x))
    print("\nEl programa realizó %d iteraciones de bisección\n" % (n))
    print("      intervalo = %27.20e" % (intervalo) )
    print("      abs(f(c)) = %27.20e" % (abs(f(x)) ) )
    print()
    return x

# Caso a) -----------------------------------------
# punto inicial
x0 = 1.5

# intervalo
a = 0.1
b = 1.5

t1 = np.arange(a, b, (b-a)/150)  # para el gráfico

# Tolerancia para el intervalo que contiene la raiz
epsilon = 1.0e-6
delta   = 1.0e-6

# máximo número de pasos
M = 20

x1 = newton(f1,df1,x0,epsilon,delta,M)

print(' x =',x1)

# Caso b) -----------------------------------------
# punto inicial
x0 = 0.

# intervalo
a = 0.
b = 1.

t2 = np.arange(a, b, (b-a)/150)  # para el gráfico

# Tolerancia para el intervalo que contiene la raiz
epsilon = 1.0e-6
delta   = 1.0e-6
M = 20

x2 = newton(f2,df2,x0,epsilon,delta,M)

print(' x =',x2)

print('------')
#-----------------------------------------------
```

```
"""
   Hagamos un gráfico para 'ver' las raíces.
"""

plt.figure( figsize=(12, 8))
plt.subplot(211)
plt.title('3*x - tan(x)')
plt.plot(t1, f1(t1))
plt.grid()
plt.axhline(y=0, color='#000000') # línea horizontal para ver la raíz
plt.axvline(x=x1, ymin=0.05, ymax=0.95, linewidth=2, color='#d62728')

plt.subplot(212)
plt.title('x**6 - 5*x**5 + x**4 - 3*x**3 + x**2 - x + 1')
plt.plot(t2, f2(t2))
plt.grid()
plt.axhline(y=0, color='#000000') # línea horizontal para ver la raíz
plt.axvline(x=x2, ymin=0.05, ymax=0.95, linewidth=2, color='#d62728')

plt.subplots_adjust(hspace=0.5)
plt.savefig('graficos/p14-g3.png', dpi=120)  # guarda figura en un archivo

plt.show()
```

- Desde la terminal ejecute:

  python3 p14-g3.py

  e interprete el resultado.

Modifique el programa y pruebe otras cosas.