Tarea N°5 Método de Newton y de la Secante

Tamara Benavidez

Tabla de Contenidos

ESCUELA POLITÉCNICA NACIONAL

TAREA N°5 Ejercicios Unidad 02 B

1. Sea ( ) = $ -x^3 -cos(x) $ y $p_0 = -1$. Use el método de Newton y de la Secante para encontrar 2. ¿Se podría usar $p_0 = 0$?

```
import math

# Función y derivada
def f(x): return -x**3 - math.cos(x)
def df(x): return -3*x**2 + math.sin(x)

# Newton p0=-1 hasta p2
p = -1
for i in range(3):
    p = p - f(p)/df(p)
print("Método de Newton p2  ", p)
```

```
Método de Newton p2   -0.865474075952977
```

El método de Newton falla porque no existe división para cero

```
# Secante p0=-1, p1=-0.5 hasta p2
p0, p1 = -1, -0.5
for i in range(2):
    p2 = p1 - f(p1)*(p1-p0)/(f(p1)-f(p0))
    p0, p1 = p1, p2
print("Método de la Secante p2  ", p2)
```

```
Método de la Secante p2   -0.8922107665609997
```

2. Encuentre soluciones precisas dentro de 10^{-4} para los siguientes problemas.

```
import math

# Función para Newton
def newton(f, df, x0, tol=1e-4):
    x = x0
    while True:
        x_next = x - f(x)/df(x)
        if abs(x_next - x) < tol: break
        x = x_next
    return x

# Función para Secante
def secante(f, p0, p1, tol=1e-4):
    while True:
        p2 = p1 - f(p1)*(p1-p0)/(f(p1)-f(p0))
        if abs(p2 - p1) < tol: break
        p0, p1 = p1, p2
    return p2
```

a. $x^3 - 2x^2 - 5 = 0$, **[1,4]**

```
f = lambda x: x**3 - 2*x**2 - 5
df = lambda x: 3*x**2 - 4*x
r2a = newton(f, df, 2)

print("Raíz 2a  ", r2a)
```

```
Raíz 2a   2.6906771528603617
```

b. $x^3 - 3x^2 - 1 = 0$**, [-3,-2]**

```
f = lambda x: x**3 + 3*x**2 -1
df = lambda x: 3*x**2 + 6*x
r2b = newton(f, df, -2.5)

print("Raíz 2b  ", r2b)
```

```
Raíz 2b   -2.87938532466927
```

c. $ x-cosx=0 $, [0,$ **$/2]**

```
f = lambda x: x - math.cos(x)
r2c = secante(f, 0, math.pi/2)

print("Raíz 2c  ", r2c)
```

```
Raíz 2c   0.739085133034638
```

d. $ x-0.8-0.2senx =0 $, [0,$ **$/2]**

```
f = lambda x: x - 0.8 - 0.2*math.sin(x)
r2d = secante(f, 0, math.pi/2)

print("Raíz 2d  ", r2d)
```

```
Raíz 2d   0.964333884548886
```

3 Use los 2 métodos en esta sección para encontrar las soluciones dentro de 10^{-5} **para los siguientes problemas.**

a. $3x - e^x = 0$ **para** $1 \leq x \leq 2$

```
f = lambda x: 3*x - math.exp(x)
df = lambda x: 3 - math.exp(x)

# Newton
x = 1.5
while True:
    x_next = x - f(x)/df(x)
    if abs(x_next - x) < 1e-5: break
    x = x_next
print("Eje del método de Newton:", x)

# Secante
p0, p1 = 1,2
while True:
    p2 = p1 - f(p1)*(p1-p0)/(f(p1)-f(p0))
    if abs(p2 - p1) < 1e-5: break
    p0, p1 = p1, p2
print("Eje del método de la Secante:", p2)
```

```
Eje del método de Newton: 1.512134625427124
Eje del método de la Secante: 1.5121345517620621
```

b. $ 2x +3cosx - e^{x} =0 $ para $ 1 x 2 $

```
f = lambda x: 2*x + 3*math.cos(x) - math.exp(x)
df = lambda x: 2 - math.exp(x) -3*math.sin(x)

# Newton
x = 1.5
while True:
    x_next = x - f(x)/df(x)
    if abs(x_next - x) < 1e-5: break
    x = x_next
print("Eje del método de Newton:", x)

# Secante
p0, p1 = 1,2
while True:
    p2 = p1 - f(p1)*(p1-p0)/(f(p1)-f(p0))
```

```
    if abs(p2 - p1) < 1e-5: break
    p0, p1 = p1, p2
print("Eje del método de la Secante:", p2)
```

```
Eje del método de Newton: 1.2397147825931407
Eje del método de la Secante: 1.2397146979752531
```

4 El polinomio de cuarto grado $f(x) = 230x^4 + 18x^3 + 9x^2 - 221x - 9$ tiene dos ceros reales, uno en [−1,0] y el otro en [0,1]. Intente aproximar estos ceros dentro de 10^{-6} con

a. El método de la secante (use los extremos como las estimaciones iniciales)

b. El método de Newton (use el punto medio como estimación inicial)

```
f = lambda x: 230*x**4 + 18*x**3 + 9*x**2 -221*x -9
df = lambda x: 920*x**3 + 54*x**2 + 18*x -221

# Secante [-1,0]
p0,p1 = -1,0
for _ in range(50):
    p2 = p1 - f(p1)*(p1-p0)/(f(p1)-f(p0))
    if abs(p2-p1)<1e-6: break
    p0,p1 = p1,p2
sec_neg = p2

# Newton [-1,0] punto medio
x = -0.5
while True:
    x_next = x - f(x)/df(x)
    if abs(x_next - x)<1e-6: break
    x = x_next
newt_neg = x

# Secante [0,1]
p0,p1 = 0,1
for _ in range(50):
    p2 = p1 - f(p1)*(p1-p0)/(f(p1)-f(p0))
    if abs(p2-p1)<1e-6: break
    p0,p1 = p1,p2
```

```
sec_pos = p2

# Newton [0,1]
x = 0.5
while True:
    x_next = x - f(x)/df(x)
    if abs(x_next - x)<1e-6: break
    x = x_next
newt_pos = x

print("Secante [-1,0]:",sec_neg,"Newton [-1,0]:",newt_neg)
print("Secante [0,1]:",sec_pos,"Newton [0,1]:",newt_pos)
```

```
Secante [-1,0]: -0.040659288315725135 Newton [-1,0]: -0.04065934349732934
Secante [0,1]: -0.04065928831557162 Newton [0,1]: -0.04065928834533494
```

5. La funcion $f(x)=tan(x)-6 $ tiene cero en $1/\pi arcontangente 6 \approx 0.447431543$. Sea $p_0 = 0$ y $p_1 = 0.48$ y use 10 iteraciones en cada uno de los siguientes métodos para aproximar esta raíz. ¿Cuál método es más eficaz y por qué?

a. método de bisección

```
f = lambda x: math.tan(math.pi*x)-6
df = lambda x: math.pi/(math.cos(math.pi*x)**2)

a,b = 0,0.48
for _ in range(10):
    c = (a+b)/2
    if f(a)*f(c)<=0: b=c
    else: a=c
bisec = (a+b)/2

print("Bisección  ", bisec)
```

```
Bisección   0.44742187499999997
```

b. método de Newton

```
x=0
for _ in range(10):
    x = x - f(x)/df(x)

print("Newton  ", x)
```

```
Newton   13.655012218663435
```

c. método de la secante

```
p0,p1 = 0,0.48
for _ in range(10):
    p2 = p1 - f(p1)*(p1-p0)/(f(p1)-f(p0))
    p0,p1 = p1,p2

print("Secante  ", p2)
```

```
Secante   -2989.9400375314453
```

6. La función descrita por $f(x) = ln(x^2 + 1) - e^{0.4x}cos\pi x$ tiene un número infinito de ceros.

```
f = lambda x: math.log(x**2+1)-math.exp(0.4*x)*math.cos(math.pi*x)

def bisection(f,a,b,tol=1e-6):
    while b-a>tol:
        c=(a+b)/2
        if f(a)*f(c)<=0: b=c
        else: a=c
    return (a+b)/2
```

a. Determine, dentro de 10^{-6}, el único cero negativo.

```
neg = bisection(f,-1,0)
print("Cero negativo  ", neg)
```

```
Cero negativo   -0.4341425895690918
```

b. Determine, dentro de 10^{-6}, los cuatro ceros positivos más pequeños

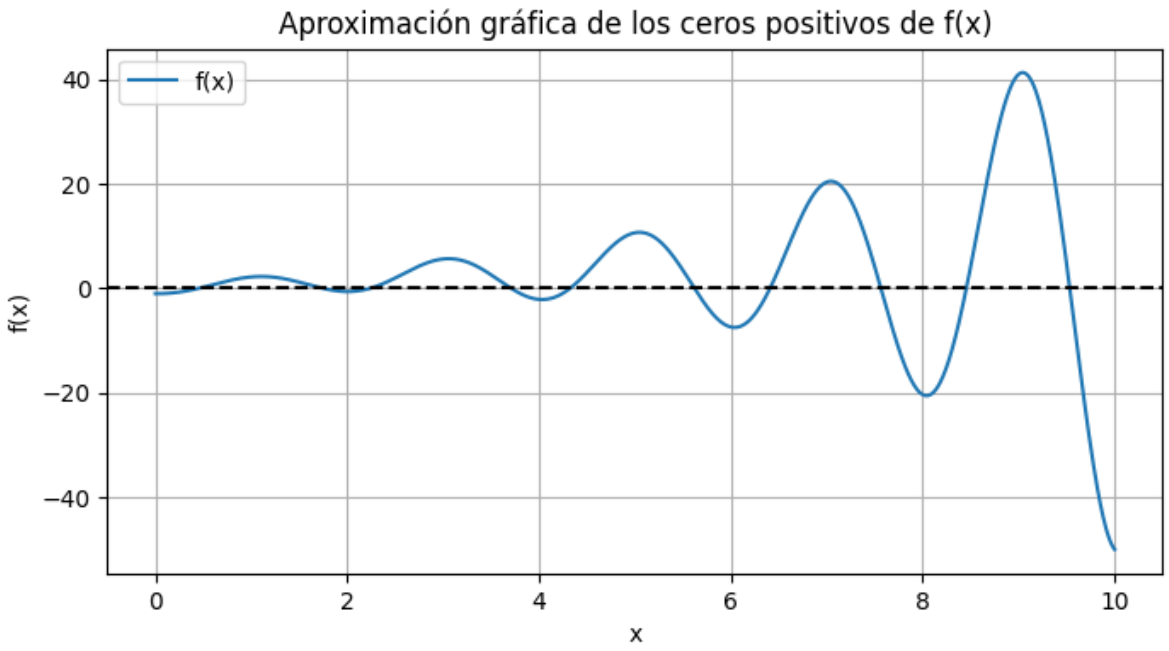
```
pos = [bisection(f,i,i+1) for i in range(5)]
print("Ceros positivos  ", pos)
```

```
Ceros positivos   [0.4506564140319824, 1.7447381019592285, 2.2383198738098145, 3.709041118621
```

c. Determine una aproximación inicial razonable para encontrar el enésimo cero positivo más pequeño de . [Sugerencia: Dibuje una gráfica aproximada de .]

```
import numpy as np
import matplotlib.pyplot as plt
import math

# Definir la función
f = lambda x: math.log(x**2+1) - math.exp(0.4*x)*math.cos(math.pi*x)

# Graficar f(x) para x entre 0 y 10 para ver ceros
x_vals = np.linspace(0, 10, 1000)
y_vals = [f(x) for x in x_vals]

plt.figure(figsize=(8,4))
plt.plot(x_vals, y_vals, label='f(x)')
plt.axhline(0, color='black', linestyle='--')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Aproximación gráfica de los ceros positivos de f(x)')
plt.grid(True)
plt.legend()
plt.show()

# Aproximación inicial para el enésimo cero positivo
```

```
# Observando el gráfico, los ceros positivos parecen casi 1 unidad separados
# Entonces, una aproximación inicial para el n-ésimo cero positivo:
n = 25
x_init = n  # como primera estimación para el n-ésimo cero positivo
print("Aproximación inicial razonable para el 25º cero positivo:", x_init)
```




```
Aproximación inicial razonable para el 25º cero positivo: 25
```

d. Use la parte c. para determinar, dentro de 10^{-6}, el vigesimoquinto cero positivo más pequeño de .

```
pos25 = bisection(f,24,25)
print("Cero cerca de 25  ", pos25)
```

```
Cero cerca de 25   24.499886989593506
```

La función $f(x) = x^{1/3}$, raíz en $x = 0$.Usando el punto de inicio de =1 y 0=5, 1=0.5 para el método de secante, compare los resultados de los métodos de la secante y de Newt

```
f = lambda x: x**(1/3)
df = lambda x: (1/3)/x**(2/3) if x!=0 else float('inf')

# Newton
x=1
for _ in range(20):
    x_next = x - f(x)/df(x)
    x = x_next
newton_res = x

# Secante
p0,p1 = 5,0.5
for _ in range(20):
    p2 = p1 - f(p1)*(p1-p0)/(f(p1)-f(p0))
    p0,p1 = p1,p2

print("Método de Newton:",newton_res)
print("Método de la Secante:",p2)
```

```
Método de Newton: (1048575.9999999895-2.7063199583615062e-09j)
Método de la Secante: (543.326975583032+3141.131427360435j)
```

En este ejercicio con los valores iniciales usados, los dos métodos fallan porque no logran acercarse a la raíz. Para que funcionen, es importante elegir los puntos iniciales más cercanos a la raíz

Link del repositorio GitHub

github_TamyBenavidez, Tarea N°5