Tabla de Contenidos

ESCUELA POLITÉCNICA NACIONAL	Tarea No.
Metodos Numericos – Computación	**5**
NOMBRE: Ivonne Carolina Ayala	

[Tarea 05] Unidad 03-A

Conjunto de Ejercicios

Ejercicio 1

1. Sea $f(x) = -x^3 - \cos x$ y $p_0 = -1$. Use el método de Newton y de la Secante para encontrar p_2. ¿Se podría usar $p_0 = 0$?

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize

f = lambda x: -x**3 - np.cos(x)

root = optimize.newton(f, -1, fprime=lambda x:-3*x**2 + np.sin(x))
print('La raiz de la función con el método de Newton es: {}'.format(root))
root = optimize.newton(f, -1)
print('La raiz de la función con el método de Secante es: {}'.format(root))
```

```
La raiz de la función con el método de Newton es: -0.8654740331016144
La raiz de la función con el método de Secante es: -0.8654740331016144
```

¿Se puede usar $p_0 = 0$?
No, no se puede usar $p_0 = 0$ para el método de Newton porque en ese punto la derivada de $f(x)$ es:

$$f'(0) = -3(0)^2 + \sin(0) = 0$$

Esto hace que el denominador en la fórmula de Newton se anule, y el método no puede avanzar. En cambio, el método de la secante podría usarse con $p_0 = 0$ y $p_1 \neq 0$.

Ejercicio 2

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

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

```
f = lambda x: x**3 - 2*x**2 - 5
tol = 10**(-4)
root = optimize.newton(f, 1, fprime=lambda x:3*x**2 - 4*x, tol = tol)
print('La raiz de la función es: {}'.format(root))
```

```
La raiz de la función es: 2.6906474480286287
```

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

```
f = lambda x: x**3 + 3*x**2 - 1
root = optimize.newton(f, -3, fprime=lambda x:3*x**2 + 2*x, tol = tol)
print('La raiz de la función es: {}'.format(root))
```

```
La raiz de la función es: -2.879486729132456
```

c. $x - cos(x) = 0, [0, \frac{\pi}{2}]$

```
f = lambda x: x - np.cos(x)
root = optimize.newton(f, 0, tol = tol)
print('La raiz de la función es: {}'.format(root))
```

```
La raiz de la función es: 0.7390851121452099
```

d. $x - 0.8 - 0.2sen(x) = 0, [0, \frac{\pi}{2}]$

```
f = lambda x: x - 0.8 - 0.2* np.sin(x)
root = optimize.newton(f, 0, tol = tol)
print('La raiz de la función es: {}'.format(root))
```

```
La raiz de la función es: 0.9643338895520454
```

Ejercicio 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 - np.e**x
root = optimize.newton(f, 2, tol = tol)
print('La raiz de la función con el metodo de la secante es: {}'.format(root))
root = optimize.newton(f, 2, fprime = lambda x: 3 - np.e**x, tol = tol)
print('La raiz de la función con el metodo de Newton es: {}'.format(root))
```

```
La raiz de la función con el metodo de la secante es: 1.512134607515724
La raiz de la función con el metodo de Newton es: 1.5121345516685922
```

b. $2x + 3cos(x) - e^x = 0$ para $1 \leq x \leq 2$

```
f = lambda x: 2*x + 3*np.cos(x) - np.e**x
root = optimize.newton(f, 2)
print('La raiz de la función con el metodo de la secante es: {}'.format(root))
root = optimize.newton(f, 2, fprime = lambda x: 2 - 3*np.sin(x) - np.e**x)
print('La raiz de la función con el metodo de Newton es: {}'.format(root))
```

```
La raiz de la función con el metodo de la secante es: 1.2397146979752174
La raiz de la función con el metodo de Newton es: 1.2397146979752174
```

Ejercicio 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 $[0,]$. Intente aproximar estos ceros dentro de 10^{-6} con

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

```
f = lambda x: 230*x**4 + 18*x**3 +9*x**2 - 221*x - 9
tol = 10**-6
root = optimize.newton(f, -1)
print(f'La raiz de la funcion es: {root}')
root = optimize.newton(f, 1)
print(f'La raiz de la funcion es: {root}')
```

```
La raiz de la funcion es: -0.040659288315758865
La raiz de la funcion es: 0.9623984187505422
```

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
tol = 10**-6
root = optimize.newton(f, -0.5, fprime= lambda x: 920*x**3 + 54*x**2 + 18*x - 221)
print(f'La raiz de la funcion es: {root}')
root = optimize.newton(f, 2, fprime= lambda x: 920*x**3 + 54*x**2 + 18*x - 221)
print(f'La raiz de la funcion es: {root}')
```

```
La raiz de la funcion es: -0.040659288315758865
La raiz de la funcion es: 0.9623984187505414
```

Ejercicio 5

La función $f(x) = tan(\pi x) - 6$ tiene cero en $(1/\pi)$ arcotangente 6 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: np.tan(np.pi*x) - 6

try:
    root = optimize.bisect(f, 0, 0.48, maxiter = 10)
    print(root)
except RuntimeError as e:
    print(e)
```

```
Failed to converge after 10 iterations.
```

b. método de newton

```
f = lambda x: np.tan(np.pi*x) - 6

try:
    root = optimize.newton(f, 0.48, fprime= lambda x:np.arctan(np.pi*x) * np.pi, maxiter=10)
    print(root)
except RuntimeError as e:
    print(e)
```

```
Failed to converge after 10 iterations, value is -12.778491310467388.
```

c. método de la secante

```
f = lambda x: np.tan(np.pi*x) - 6


try:
    root = optimize.newton(f, 0.48, maxiter=10)
    print(root)
except RuntimeError as e:
    print(e)
```

```
0.44743154328893403
```

El método más eficiente es el de la secante, porque es el único que converge.

Ejercicio 6

La función descrita por $f(x) = ln(x2 + 1) - e^{0.4x}cos(\pi x)$ tiene un número infinito de ceros.

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

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

root = optimize.newton(f, -0.5, tol=10**-6)
print(root)
```

```
-0.43414304730000164
```

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

```
p = [0.4, 1.6, 2, 3]

roots = []

for p0 in p:
    try:
        root = optimize.newton(f, p0, tol=1e-6)
        roots.append(root)
    except RuntimeError as e:
        print(e)

print("Raíces encontradas:", roots)
```

```
Failed to converge after 50 iterations, value is -2.062270253992078e+26.
Failed to converge after 50 iterations, value is -2.0014414689911985e+27.
Failed to converge after 50 iterations, value is 3.1714517236664837e+18.
Raíces encontradas: [0.44743154328871987]
```

c. Determine una aproximación inicial razonable para encontrar el enésimo cero positivo más pequeño de f.

```
root = optimize.newton(f, 9.4, tol=1e-6)
print(f'El enésimo cero positivo más pequeño es {root}')
```

```
El enésimo cero positivo más pequeño es 9.447431543755696
El enésimo cero positivo más pequeño es 9.447431543755696
```

d. Use la parte c) para determinar, dentro de 10−6, el vigesimoquinto cero positivo más pequeño de f.

```
root = optimize.newton(f, 24.4, tol=1e-6)
print(f'El vigesimoquinto cero positivo más pequeño es {root}')
```

```
El vigesimoquinto cero positivo más pequeño es 0.9623984187540163
```

7. La función $f(x) = x^{\frac{1}{3}}$ tiene raíz en $x = 0$.Usando el punto de inicio de $x = 1$ y $p_0 = 5$, $p_1 = 0.5$ para el método de secante, compare los resultados de los métodos de la secante y de Newton.

```python
f = lambda x: np.cbrt(x)

try:
    root = optimize.newton(f, x0=5, x1= 0.5)
    print(f'La raiz con el metodo de la secante es {root}')
except RuntimeError as e:
    print(e)

try:
    root = optimize.newton(f, x0=5, fprime=lambda x:(1/3)*x**(-(2/3)))
    print(f'La raiz con el metodo de Newton es {root}')
except RuntimeError as e:
    print(e)
```

```
Failed to converge after 50 iterations, value is 0.15125956067017968.
Failed to converge after 50 iterations, value is nan.
```

Ninguno de los dos métodos converge por el dominio de la función, sin embargo, el único método que devuelve un valor es la secante