

Tarea 5

Danny Iñaguazo

Tabla de Contenidos

Conjunto de ejercicios	1
Ejercicio 1	2
Ejercicio 2	3
Literal a)	3
Literal b)	3
Literal c)	3
Literal d)	4
Ejercicio 3	4
Literal a)	4
Literal b)	5
Ejercicio 4	5
Método de la secante	5
Método de newton	6
Ejercicio 5	7
Método de la Bisección	7
Método de Newton	8
Método de la Secante	8
Ejercicio 6	9
Literal a)	9
Literal b)	9
Literal c)	10
Literal d)	12
Ejercicio 7	13

Conjunto de ejercicios

Para los siguientes ejercicios, utilizaremos la librería scipy, que incluye el método de la secante y newton.

```
from scipy.optimize import newton
import math
```

En los próximos ejercicios, entenderemos cómo solo este módulo implementa los dos métodos.

Ejercicio 1

Sea $f(x) = -x^3 - \cos x$ y $p_0 = -1$. Use el método de Newton y de la Secante para encontrar p_1 . ¿Se podría usar $p_0 = 0$?

Para el método de newton, necesitamos la derivada de la función anterior. Como resultado:
 $f'(x) = -3x^2 + \sin x$

Llamamos a la función Newton con los parámetros que tenemos:

```
def fprime(x):
    return -3*x**2 + math.sin(x)

p1 = newton(func = lambda x : -x**3 - math.cos(x), x0 = -1, fprime = fprime)
print(p1)
```

-0.8654740331016144

Para el método de la secante, no necesitamos la derivada de la función.

```
p1 = newton(func = lambda x : -x**3 - math.cos(x), x0 = -1)
print(p1)
```

-0.8654740331016144

Finalmente, el cero como punto aproximado es una mala elección. Para el método de newton, esto implica que en $p_0 = 0$ la derivada sea cero, algo no permitido. En la secante, este punto produce complicaciones por la presencia de multiplicaciones para cero.

```
p1 = newton(func = lambda x : -x**3 - math.cos(x), x0 = 0)
print(p1)
```

-4.998000183473029e-09

Ejercicio 2

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

Literal a)

Para el intervalo $[1, 4]$ en la ecuación $x^3 - 2x^2 - 5 = 0$.

Usamos el método de la secante:

```
p1 = newton(func = lambda x : x**3 - 2*x**2 - 5, x0 = 1, tol = 10e-4, x1 = 4)
print(p1)
```

2.6906484961992585

Literal b)

Para el intervalo $[-3, -2]$ en la ecuación $x^3 + 3x^2 - 1 = 0$.

Usamos el método de la secante:

```
p1 = newton(func = lambda x : x**3 + 3*x**2 - 1, x0 = -3, tol = 10e-4, x1 = -2)
print(p1)
```

-2.879385194736809

Literal c)

Para el intervalo $[0, \frac{\pi}{2}]$ en la ecuación $x - \cos x = 0$.

Usamos el método de la secante:

```
p1 = newton(func = lambda x : x - math.cos(x), x0 = 0, tol = 10e-4, x1 = math.pi/2)
print(p1)
```

0.7390834365030763

Literal d)

Para el intervalo $[0, \frac{\pi}{2}]$ en la ecuación $x - 0.8 - 0.2 \sin x = 0$.

Usamos el método de la secante:

```
p1 = newton(func = lambda x : x - 0.8 - 0.2*math.sin(x), x0 = 0, tol = 10e-4, x1 = math.pi/2)
print(p1)
```

0.9643338835706312

Ejercicio 3

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

Literal a)

Para el intervalo $1 \leq x \leq 2$ en la ecuación $3x - e^x = 0$.

Calculamos la derivada de la función, tal que: $3 - e^x$. Utilizamos el método de newton:

```
p1 = newton(func = lambda x : 3*x - math.exp(x), x0 = 1,
            fprime = lambda x : 3 - math.exp(x), tol = 10e-5, x1 = 2)
print(p1)
```

0.6190612833553127

Para el método de la secante:

```
p1 = newton(func = lambda x : 3*x - math.exp(x), x0 = 1, tol = 10e-5, x1 = 2)
print(p1)
```

1.5121345517620621

Los dos encontraron raíces diferentes, pero el método de newton encontró una fuera del intervalo dado, debido a que en uno de los puntos dado, la pendiente convergía hacia otra dirección (izquierda).

Literal b)

Para el intervalo $1 \leq x \leq 2$ en la ecuación $2x + 3 \cos x - e^x = 0$.

Calculamos la derivada de la función, tal que: $2 - 3 \sin x - e^x$. Utilizamos el método de newton:

```
p1 = newton(func = lambda x : 2*x + 3*math.cos(x) - math.exp(x),
            x0 = 1,
            fprime = lambda x : 2 - 3*math.sin(x) - math.exp(x),
            tol = 10e-5, x1 = 2)
print(p1)
```

1.2397146979752596

Para el método de la secante:

```
p1 = newton(func = lambda x : 2*x + 3*math.cos(x) - math.exp(x),
            x0 = 1, tol = 10e-5, x1 = 2)
print(p1)
```

1.2397146920815107

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 $[0, 1]$. Intente aproximar estos ceros dentro de 10^{-6} con:

Método de la secante

El ejercicio nos pide usar los dos extremos como aproximaciones iniciales. Para el primer intervalo $[-1, 0]$:

```
p1 = newton(func = lambda x : 230*x**4 + 18*x**3 + 9*x**2 - 221*x - 9,
            x0 = -1, tol = 10e-6, x1 = 0)
print(p1)
```

-0.04065928497591696

Para el segundo intervalo (El intervalo en la hoja original solo dispone de la aproximación 0, así que usamos 0.9 como aproximación):

```
p1 = newton(func = lambda x : 230*x**4 + 18*x**3 + 9*x**2 - 221*x - 9,
            x0 = 0.9, tol = 10e-6)
print(p1)
```

0.9623984187477349

Método de newton

Primero calculamos la derivada:

$$920x^3 + 54x^2 + 18x - 221$$

El ejercicio nos pide usar la media de los extremos como aproximación inicial. Para el primer intervalo $[-1, 0]$ con su mediana -0.5 :

```
p1 = newton(func = lambda x : 230*x**4 + 18*x**3 + 9*x**2 - 221*x - 9,
            x0 = -0.5,
            fprime = lambda x : 920*x**3 + 54*x**2 + 18*x - 221,
            tol = 10e-6)
print(p1)
```

-0.04065928831575899

Para el segundo intervalo (El intervalo en la hoja original solo dispone de la aproximación 0, así que usamos 0.9 como aproximación):

```
p1 = newton(func = lambda x : 230*x**4 + 18*x**3 + 9*x**2 - 221*x - 9,
            x0 = 0.9,
            fprime = lambda x : 920*x**3 + 54*x**2 + 18*x - 221,
            tol = 10e-6)
print(p1)
```

0.9623984187505487

Ejercicio 5

La función $f(x) = \tan \pi x - 6$ tiene cero en $(\frac{1}{\pi})$ arcotangente $6 \approx 0.447431543$. 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?

Método de la Bisección

Necesitamos el código de la asignación anterior:

```
def sign(x: float) -> int:
    if x > 0:
        return 1
    elif x < 0:
        return -1
    else:
        return 0

from typing import Callable

def bisection(
    a: float, b: float, *, equation: Callable[[float], float], N: int
) -> tuple[float, int] | None:
    i = 1

    assert a < b, "a not lower than b, the interval is not valid."

    assert (
        equation(a) * equation(b) < 0
    ), "The function does not change sign over the interval."

    Fa = equation(a)
    p = a
    for i in range(N + 1):
        p = a + (b - a) / 2
        FP = equation(p)

        if sign(Fa) * sign(FP) > 0:
            a = p
            Fa = FP

    else:
```

```

        b = p

    return p, i

```

Usamos el método con los valores propuestos:

```

p, i = bisection(a = 0, b = 0.48,
                equation = lambda x : math.tan(math.pi*x) - 6,
                N = 10)
print(f"Raíz resultante: {p} en {i} iteraciones")

```

Raíz resultante: 0.44742187499999997 en 10 iteraciones

Método de Newton

Para este caso, necesitamos la derivada

$$\pi \sec^2 \pi x$$

y usamos el algoritmo:

```

p1 = newton(func = lambda x : math.tan(math.pi*x) - 6,
            x0 = 0,
            fprime = lambda x : math.pi*(1/math.cos(math.pi*x))**2,
            maxiter = 10, x1 = 0.48)
print(p1)

```

RuntimeError: Failed to converge after 10 iterations, value is 13.655012218324751.

Uno de los errores es la pendiente que tiende a converger hacia los valores mínimos.

Método de la Secante

Usamos el algoritmo con los parámetros:

```

p1 = newton(func = lambda x : math.tan(math.pi*x) - 6,
            x0 = 0, maxiter = 10, x1 = 0.48)
print(p1)

```


RuntimeError: Failed to converge after 10 iterations, value is -3694.358600967476.

Como conclusión, el método de la bisección es más efectivo en cuanto a hallar una raíz en la función. Las otras dos funciones tienden a manejarse en cuanto a pendiente o a una línea que entre corta el eje x, así que estos cálculos se salen del intervalo dado. Otra razón es la propia función, ya que no es continua respecto a los reales.

Ejercicio 6

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

Literal a)

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

Usamos el método de secante:

```
p1 = newton(func = lambda x : math.log(x**2 + 1) - math.exp(0.4*x)*math.cos(math.pi*x),
            x0 = -0.4, tol = 10e-6)
print(p1)
```

-0.43414304724770203

Literal b)

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

Usamos el método de secante:

```
p1 = newton(func = lambda x : math.log(x**2 + 1) - math.exp(0.4*x)*math.cos(math.pi*x),
            x0 = 0.5, tol = 10e-6)
print(p1)
```

0.4506567478906115

```
p1 = newton(func = lambda x : math.log(x**2 + 1) - math.exp(0.4*x)*math.cos(math.pi*x),
            x0 = 1.5, tol = 10e-6)
print(p1)
```

1.7447380533760186

```
p1 = newton(func = lambda x : math.log(x**2 + 1) - math.exp(0.4*x)*math.cos(math.pi*x),  
            x0 = 2.5, tol = 10e-6)  
print(p1)
```

2.238319795077607

```
p1 = newton(func = lambda x : math.log(x**2 + 1) - math.exp(0.4*x)*math.cos(math.pi*x),  
            x0 = 3.5, tol = 10e-6)  
print(p1)
```

3.709041201416693

Literal c)

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

Para este literal, se necesitó de unan gráfica, hecha en Geogebra.

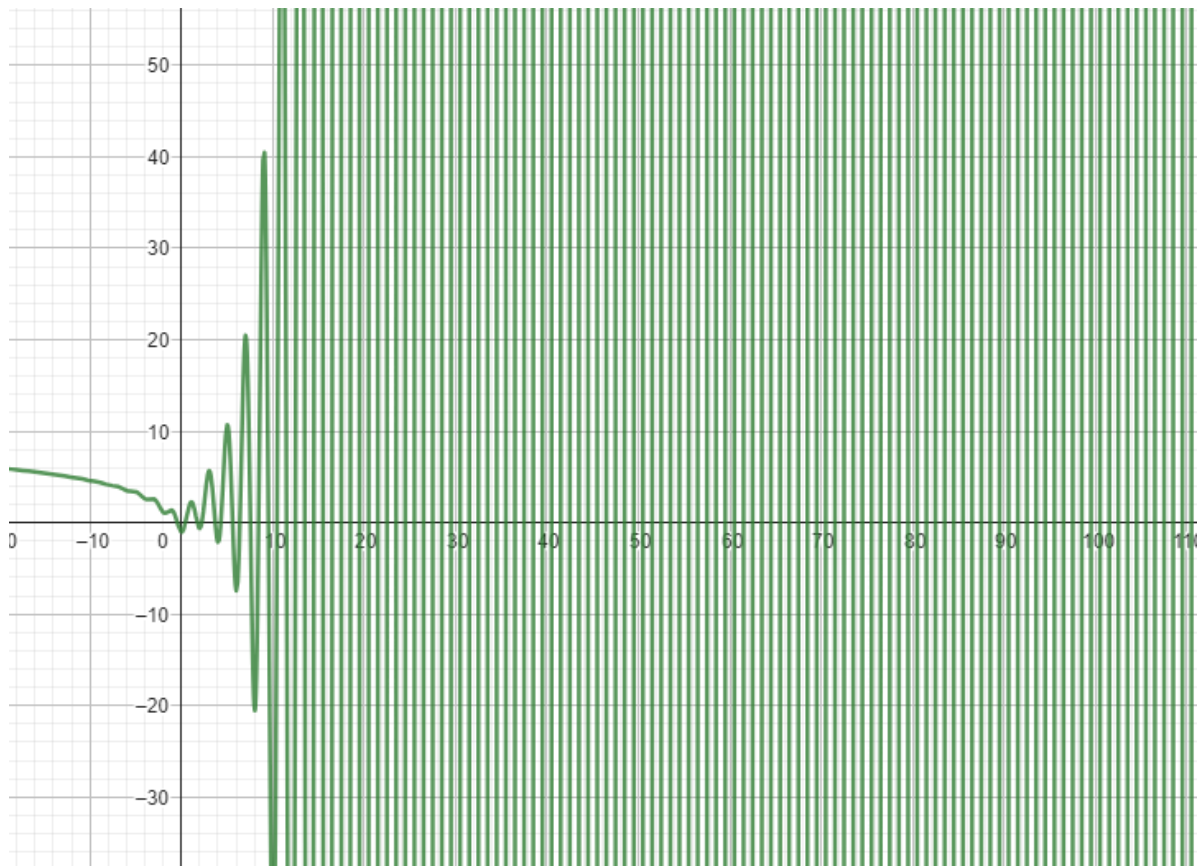


Figura 1: Gráfica estimada de la función

Existen infinitas raíces positivas según el gráfico, la función tiene una forma de onda que va creciendo mientras x avanza. Necesitaríamos verlo más cerca.

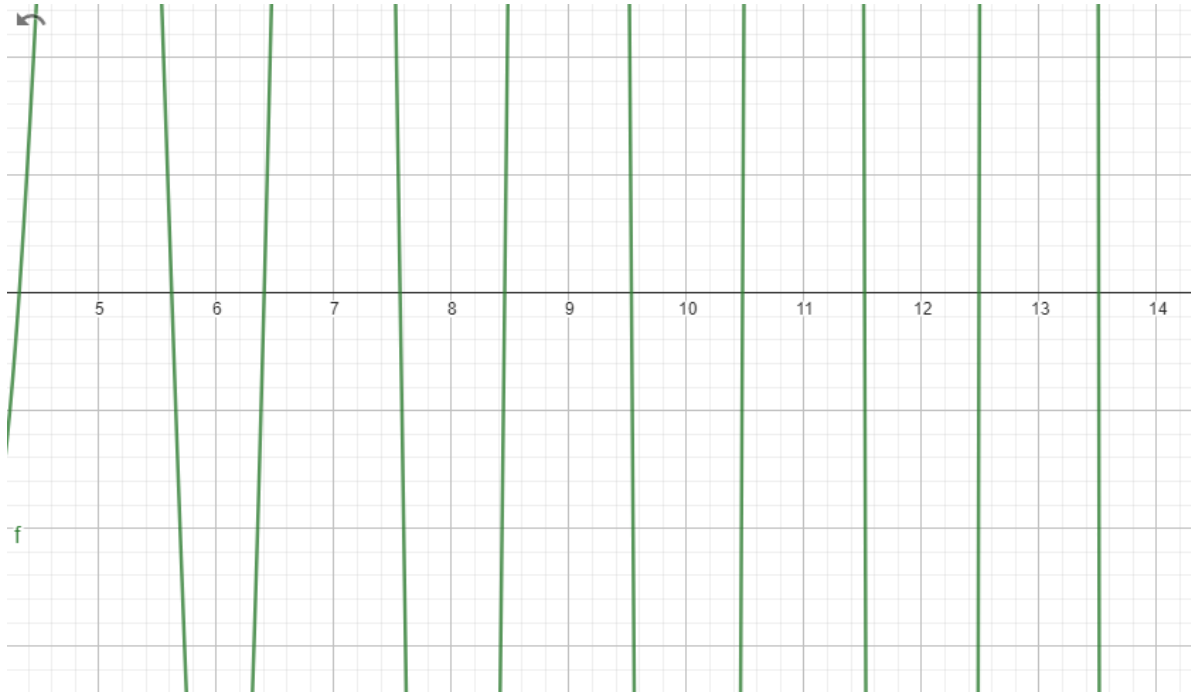


Figura 2: Parte pequeña de la gráfica de la función

En esta imagen, podemos observar que las raíces tienden a estar cerca de un número entero $+ 0.5$ infinitamente, por lo que podemos usar esto como punto de aproximación para encontrar la n ésima raíz de esta función. La fórmula propuesta para encontrar la n ésima raíz de la función es:

$$n - 0.5$$

Tal que n representa una aproximación a la n ésima raíz positiva, para $n \in \mathbb{N} - \{0\}$.

Literal d)

Use la parte c) para determinar, dentro de 10^{-6} , el vigesimoquinto (25) cero positivo más pequeño de f .

Usamos la fórmula $n - 0.5$ con $n = 25$, tal que $(25) - 0.5 = 24.5$.

El resultado lo empleamos como parámetro en el método de la secante:

```
p1 = newton(func = lambda x : math.log(x**2 + 1) - math.exp(0.4*x)*math.cos(math.pi*x),
            x0 = 24.5, tol = 10e-6)
print(p1)
```

24.49988704757148

Ejercicio 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.

Para el método de newton, su derivada es: $\frac{1}{3x^{\frac{2}{3}}}$. Junto con los parámetros dados, llamados a la función de Newton

```
p1 = newton(func = lambda x : x**(1/3), x0 = 1,
            fprime = lambda x : 1 / (3*x**(2/3)))
print(p1)
```

RuntimeError: Failed to converge after 50 iterations, value is nan.

Para el método de la secante, solo usamos los parámetros dados:

```
p1 = newton(func = lambda x : x**(1/3) + 1, x0 = 1)
print(p1)
```

RuntimeError: Failed to converge after 50 iterations, value is nan.

En el caso de los dos, existe el error de no convergencia, básicamente dado por el cero. Además, hay que esclarecer que la única raíz existente para esa función, también es un punto de inflexión. Por último, la raíz tiene una pendiente igual a cero 0.