Métodos Numéricos - DEBER 04 - Método de Bisección

Alicia Pereira

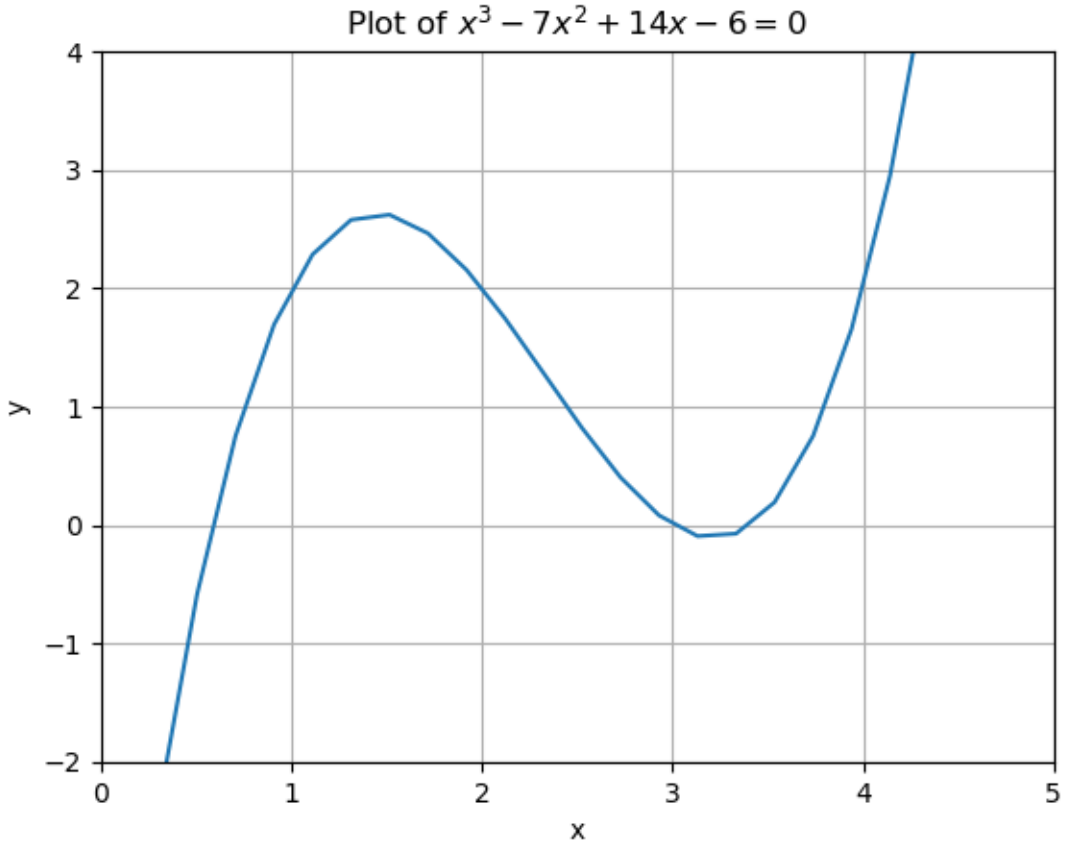
Tabla de Contenidos

1. Use el método de bisección para encontrar soluciones precisas dentro de 10^{-2} para $x^3 - 7x^2 + 14x - 6 = 0$ en cada intervalo.

Para empezar, se dibuja la gráfica de la función requerida para de este modo tener una mejor visualización del ejercicio y la bisección a realizar.

```
import numpy as np

import matplotlib.pyplot as plt

def equation(x:float)->float:
    return (x**3 - 7*x**2 + 14*x - 6)

x = np.linspace(-10, 10, 100)

y = equation(x)

plt.plot(x, y)

plt.xlabel('x')
plt.ylabel('y')
plt.title('Plot of $ x^{3} - 7x^{2} + 14x - 6 = 0$')
ax = plt.gca()
ax.set_ylim([-2, 4])
```

```
ax.set_xlim([0, 5])
plt.grid(True)
plt.show()
```



El método de la bisección es aplicado mediante la función definida en Python a continuación:

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


def bisection(
```

```
    a: float, b: float, *, equation: Callable[[float], float], tol: float, N: int
) -> tuple[float, float, 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):
        p = a + (b - a) / 2
        FP = equation(p)
        if FP == 0 or (b - a) / 2 < tol:
            return p, a, b, i

        if sign(Fa) * sign(FP) > 0:
            a = p
            Fa = FP
        else:
            b = p
    return p, a, b, i
```

a. $[0, 1]$

```
a = 0
b = 1
tol = 10**(-2)

result= bisection(a = a, b=b,equation=equation,tol = tol, N = 20)
print("Al utilizar el rango ["+str(a)+","+str(b)+"], en la iteración n°: "+str(result[3])+" s
      " dentro de la precisión de "+format(tol, ".0e")+ " es: "+str(result[0]))
```

```
Al utilizar el rango [0,1], en la iteración n°: 6 se encontró que la raíz de f(x) dentro de l
```

b. $[1, 3.2]$

```
a = 1
b = 3.2
tol = 10**(-2)
```

```
result= bisection(a = a, b=b,equation=equation,tol = tol, N = 20)
print("Al utilizar el rango ["+str(a)+","+str(b)+"], en la iteración n°: "+str(result[3])+" s
      " dentro de la precisión de "+format(tol, ".0e")+ " es: "+str(result[0]))
```

```
Al utilizar el rango [1,3.2], en la iteración n°: 7 se encontró que la raíz de f(x) dentro de
```

c. $[3.2, 4]$

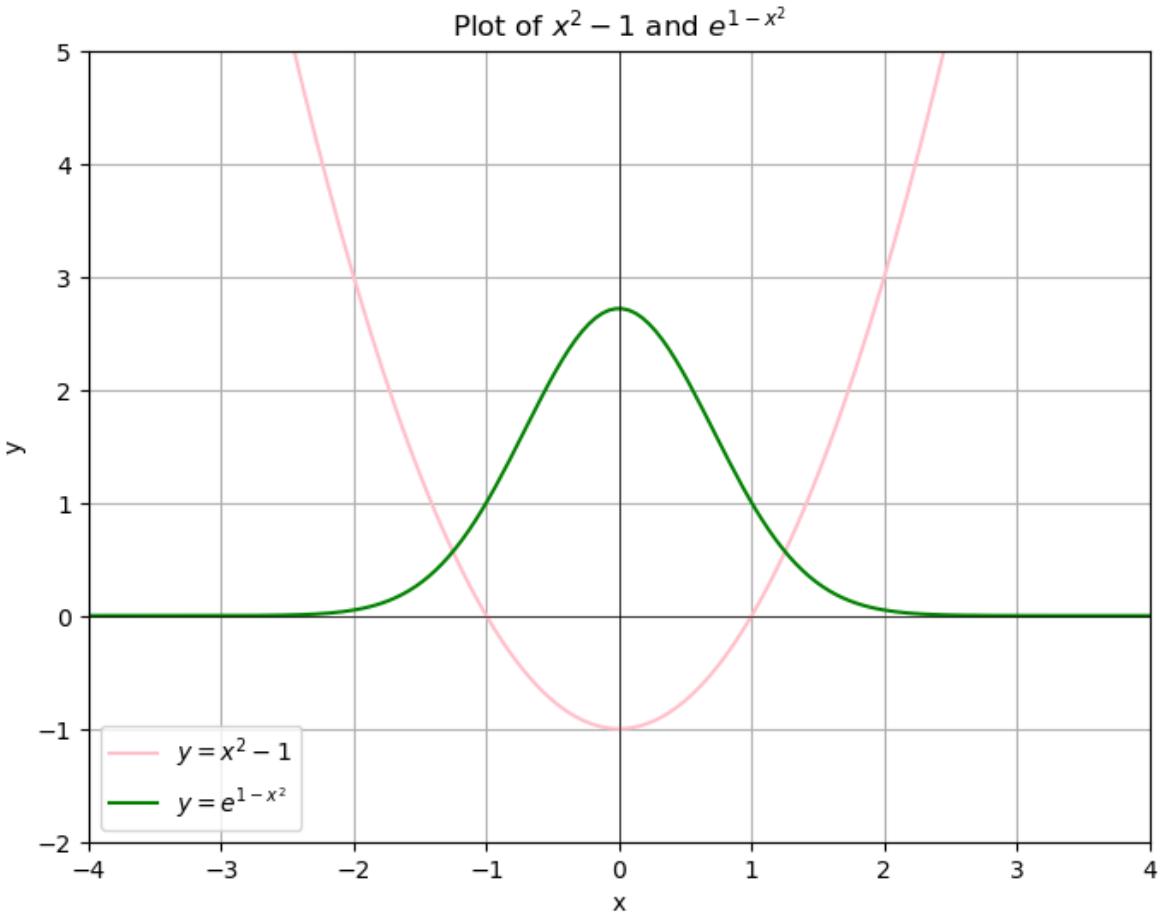
```
a = 3.2
b = 4
tol = 10**(-2)

result= bisection(a = a, b=b,equation=equation,tol = tol, N = 20)
print("Al utilizar el rango ["+str(a)+","+str(b)+"], en la iteración n°: "+str(result[3])+" s
      " dentro de la precisión de "+format(tol, ".0e")+ " es: "+str(result[0]))
```

```
Al utilizar el rango [3.2,4], en la iteración n°: 6 se encontró que la raíz de f(x) dentro de
```

4.a. Dibuje las gráficas para $y = x^2 - 1$ y $y = e^{1-x^2}$

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

def equation1(x:float)->float:
    return ((x**2) - 1)

def equation2(x:float)->float:
    return np.exp(1-x**2)

x = np.linspace(-5, 5, 200)
y1 = equation1(x)
y2 = equation2(x)
plt.figure(figsize=(8, 6))
plt.plot(x, y1, label=r'$y = x^{2} - 1$', color='pink')
plt.plot(x, y2, label=r'$y = e^{1-x^{2}}$', color='green')

plt.xlabel('x')
plt.ylabel('y')
plt.title('Plot of $x^{2}-1$ and $e^{1-x^{2}}$')
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
```

```
ax = plt.gca()
ax.set_ylim([-2, 5])
ax.set_xlim([-4, 4])

plt.grid(True)
plt.legend()
plt.show()
```

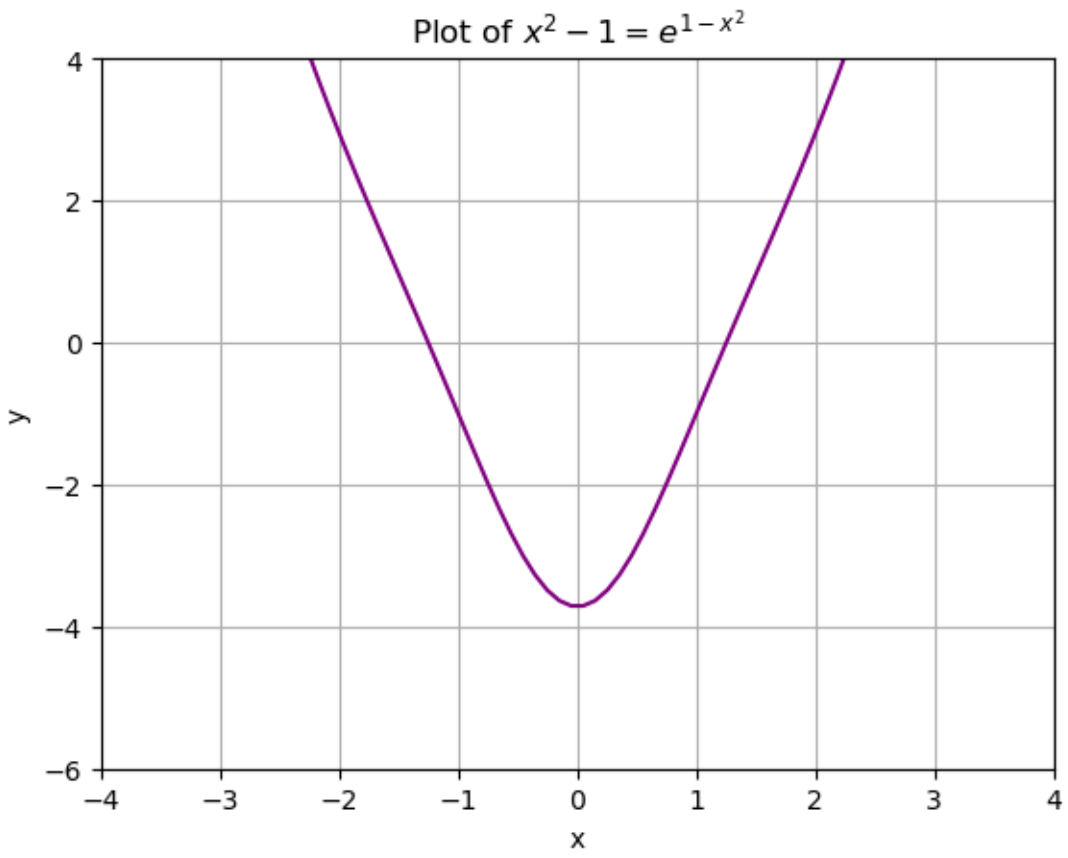


4.b. Use el método de bisección para encontrar una aproximación dentro de 10^{-3} para un valor de $[-2, 0]$ con $x^2 - 1 = e^{1-x^2}$

Para visualizar mejor el problema, se proporciona la gráfica correspondiente a la función.

```
def eq(x):
    return ((x**2) - 1)-(np.exp(1-x**2))
```

```
x = np.linspace(-5, 5, 100)

y = eq(x)

plt.plot(x, y, label = '$x^{2} - 1 = e^{1-x^{2}}$', color = 'purple')

plt.xlabel('x')
plt.ylabel('y')
plt.title('Plot of $x^{2} - 1 = e^{1-x^{2}}$')
ax = plt.gca()
ax.set_ylim([-6, 4])
ax.set_xlim([-4, 4])
plt.grid(True)
plt.show()
```



```
a = -2
b = 0
tol = 10**(-3)

result= bisection(a = a, b=b,equation=eq,tol = tol, N = 20)
print("Al utilizar el rango ["+str(a)+","+str(b)+"], en la iteración n°: "+str(result[3])+" s
      " dentro de la precisión de "+format(tol, ".0e")+ " es: "+str(result[0]))
```

```
Al utilizar el rango [-2,0], en la iteración n°: 10 se encontró que la raíz de f(x) dentro de
```

1 Ejercicios Aplicados

1. Un abrevadero de longitud tiene una sección transversal en forma de semicírculo con radio . (Consulte la figura adjunta.) Cuando se llena con agua hasta una distancia a partir de la parte superior, el volumen de agua es:

$$V = L\left(0.5\pi r^2 - r^2 \arcsin\left(\frac{h}{r}\right) - h\sqrt{r^2 - h^2}\right)$$

Suponga que = 10 , = 1 y =12.4 . Encuentre la profundidad del agua en el abrevadero dentro de 0.01 *cm*

Datos del Ejercicio:

- Tolerancia: $0.01cm$
- Intervalo: $[h_{min}, h_{max}]$, es decir: $[0, 1]$

```
import math

L = 10
r = 1
V_dado = 12.4
tol = 0.01


def f_h(h):
    V_calculado = L * (0.5 * math.pi * r**2 - r**2 * math.asin(h / r) - h * math.sqrt(r**2 -
    return V_calculado - V_dado

result = bisection(a=0,b=r,equation=f_h,tol=tol,N=20)

print("Al utilizar el rango ["+str(0)+","+str(r)+"], en la iteración n°: "+str(result[3])+" s
```

```
      " dentro de la precisión de "+format(tol, ".0e")+ " es: "+str(result[0]))
```

```
Al utilizar el rango [0,1], en la iteración n°: 6 se encontró que la raíz de f(h) dentro de
```

2. Un objeto que cae verticalmente a través del aire está sujeto a una resistencia viscosa, así como a la fuerza de gravedad. Suponga que un objeto con masa cae desde una altura y que la altura del objeto después de segundos es

$s(t) = s_0 - \frac{mg}{k}t + \frac{m^2 g}{k^2}\left(1 - e^{-\frac{kt}{m}}\right),$

donde $(g = 9.81, \mathbf{m/s^2})$ y (k) representa el coeficiente de la resistencia del aire en $(\mathbf{Ns/m})$. Suponga $(s_0 = 300, \mathbf{m})$, $(m = 0.25\,\mathbf{kg})$ y $(k = 0.1, \mathbf{Ns/m})$. Encuentre, dentro de $(0.01\,\mathbf{segundos})$, el tiempo que tarda un cuarto de kg en golpear el piso.

```
s0 = 300
m = 0.25
k = 0.1
g = 9.81
tol = 0.01

def f_t(t):
    s_t = s0 - (m * g / k) * t + (m**2 * g / k**2) * (1 - math.exp(-k * t / m))
    return s_t

result = bisection(a=0,b=s0,equation=f_t,tol=tol,N=20)

print("Al utilizar el rango ["+str(0)+","+str(s0)+"], en la iteración n°: "+str(result[3])+"
      " dentro de la precisión de "+format(tol, ".0e")+ " es: "+str(result[0]) + " seg.")
```

```
Al utilizar el rango [0,300], en la iteración n°: 14 se encontró que la raíz de f(t) dentro
```

2 Ejercicios Teóricos

1. Use el teorema 2.1. para encontrar una cota para el número de iteraciones necesarias para lograr una aproximación con precisión de 10^{-4} para la solución de $x^3 - x - 1 = 0$ que se encuentra dentro del intervalo $[1, 2]$. Encuentre una aproximación para la raíz con este grado de precisión.

```
a = 1
b = 2
tol = 10**(-4)
def equation3(x):
    return (x**(3)-x-1)

result = bisection(a=a,b=b,equation=equation3,tol=tol,N=20)

print("Después de " + str(result[3]+1) + " iteraciones la solución aproximada en la precisió
```

```
Después de 14 iteraciones la solución aproximada en la precisión de 1e-04 es: 1.3247680664062
```

GitHub: git@github.com: dayapt04

GitHub Métodos Númericos - Repositorio