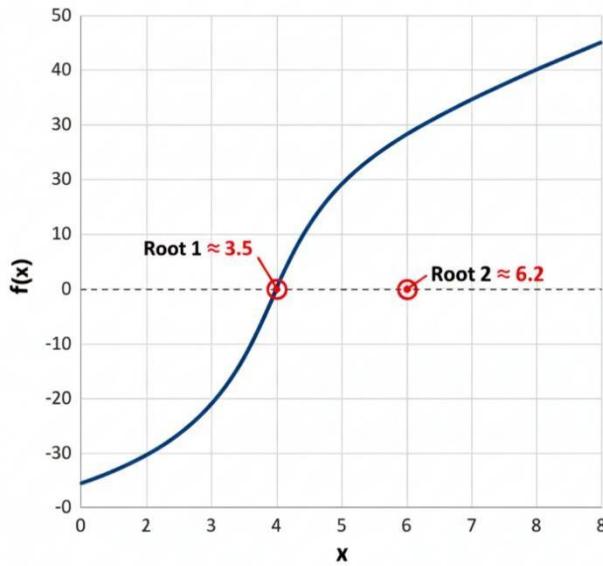


NOMBRE: CHAVARRIA DE LA CRUZ LEONOR BELEN	C.I.:9885133	SIGLA: SIS-254 MÉTODOS NUMERICOS
	RAICES 4 EJERCICIOS	

EJERCICIO 1

- Grafico

Graph of $f(x) = x^3 - e^0(0.8x - 20)$



- Solución exacta

Raíz	Valor Aproximado (x^*)
Primera Solución (x_1)	3.229037
Segunda Solución (x_2)	7.381140

- Código

```
ejer1.py > df
1 import numpy as np
2
3 # --- 1. Definición de la Función y su Derivada ---
4
5 def f(x):
6     return x**3 - np.exp(0.8 * x) - 20
7
8 def df(x):
9     return 3 * x**2 - 0.8 * np.exp(0.8 * x)
10
11 # Tolerancia de error absoluto
12 TOL = 0.0001
13
14 # --- 2. Método de Bisección ---
15
16 def biseccion(a, b, tol, raiz_id):
17     print(f"\n--- Bisección (Raíz {raiz_id}) ---")
18     print("{:<5} {:<10} {:<10} {:<15}".format("k", "a", "b", "p", "f(p)"))
19
20     fa = f(a)
21     fb = f(b)
22
23     if fa * fb > 0:
24         return f"Error: El intervalo [{a}, {b}] no encierra la raíz (no hay cambio de signo)."
25
26     p = 0
27
28     for i in range(50): # Límite de iteraciones
29         p = (a + b) / 2
30         fp = f(p)
31
32         # MOSTRAR ITERACIÓN
33         print("{:<5} {:<10.6f} {:<10.6f} {:<15.6f}".format(i + 1, a, b, fp))
34
35         # Criterio de parada: Ancho del intervalo (Error Absoluto)
36         if np.abs(b - a) / 2 < tol:
37             return f"Raíz encontrada: {p:.6f} en {i+1} iteraciones."
38
39         if fp == 0:
40             return f"Raíz exacta: {p:.6f} en {i+1} iteraciones."
41
```

```
ejer1.py > df
16     def biseccion(a, b, tol, raiz_id):
17         p = (a + b) / 2
18         if fa * fp < 0:
19             b = p
20         else:
21             a = p
22             fa = fp # Actualizar f(a)
23
24     return f"Raíz exacta: {p:.6f} en {i+1} iteraciones."
25
26
27     # --- 3. Método de Newton-Raphson ---
28
29     def newton_raphson(x0, tol, raiz_id):
30         print(f"\n--- Newton-Raphson (Raíz {raiz_id}) ---")
31         print("{:<5} {:<15} {:<15} {:<15} {:<15} ".format("k", "x_k", "f(x_k)", "f'(x_k)", "Error Abs"))
32         x_k = x0
33
34         for i in range(50): # Límite de iteraciones
35             fx = f(x_k)
36             dfx = df(x_k)
37
38             if np.abs(dfx) < 1e-10: # Evitar división por cero
39                 return "División por cero (derivada cercana a cero)."
40
41             x_k_nuevo = x_k - fx / dfx
42             error_abs = np.abs(x_k_nuevo - x_k)
43
44             # MOSTRAR ITERACIÓN
45             print("{:<5} {:<15.8f} {:<15.8f} {:<15.8f} {:<15.8f} ".format(i + 1, x_k, fx, dfx, error_abs))
46
47             # Criterio de parada: Error Absoluto |x_k_nuevo - x_k|
48             if error_abs < tol:
49                 return f"Raíz encontrada: {x_k_nuevo:.6f} en {i+1} iteraciones."
50
51             x_k = x_k_nuevo
52
53     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
54
55     # --- 4. Método de la Secante ---
56
```

```
+ ejer1.py + ejer1.py + ejer1.py + ejer1.py
ejer1.py > df

77
78     # --- 4. Método de la Secante ---
79
80     def secante(x_menos_1, x0, tol, raiz_id):
81         print(f"\n--- Método de la Secante (Raíz {raiz_id}) ---")
82         print("{:<5} {:<15} {:<15} {:<15} .format("k", "x_k", "f(x_k)", "x_k+1", "Error Abs"))
83         x_k_menos_1 = x_menos_1
84         x_k = x0
85
86         for i in range(100): # Iteraciones
87             fx_menos_1 = f(x_k)
88             fx_k = f(x_k)
89
90             if np.abs(fx_k - fx_menos_1) < 1e-10: # Evitar división por cero
91                 return "División por cero (denominador nulo)."
92
93             x_k_mas_1 = x_k - fx_k * (x_k_menos_1 - x_k) / (fx_menos_1 - fx_k)
94             error_abs = np.abs(x_k_mas_1 - x_k)
95
96             # MOSTRAR ITERACIÓN
97             print("{:<5} {:<15.8f} {:<15.8f} {:<15.8f} .format(i + 1, x_k, fx_k, x_k_mas_1, error_abs))
98
99             # Criterio de parada: Error Absoluto |x_k_mas_1 - x_k|
100            if error_abs < tol:
101                return f"Raíz encontrada: {x_k_mas_1:.6f} en {i+1} iteraciones."
102
103            x_k_menos_1 = x_k
104            x_k = x_k_mas_1
105
106        return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
107
108    # --- 5. Ejecución para la raíz ---
109
110    print("\n--- SOLUCIONES PARA f(x) = x^3 - exp(0.8x) - 20 ---")
111
112    # --- Raíz 1 (cercana a 3.229) ---
113    print("\n# Búsqueda de la Primera Raíz (x1)")
114    print(biseccion(3.0, 4.0, TOL, 1))
115    print(newton_raphson(3.5, TOL, 1))
116    print(secante(3.0, 4.0, TOL, 1))
```

Corrida

```

@@ def Secante(x_menos_1, x0, tol, raiz_10):
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[Running] python -u "c:\Users\Leonor\Desktop\codigo_python\ejer1.py"

--- SOLUCIONES PARA  $f(x) = x^3 - \exp(0.8x) - 20$  ---

# Búsqueda de la Primera Raíz (x1)

--- Bisección (Raíz 1) ---
k      a            b            p            f(p)
1      3.000000    4.000000    3.500000   6.430353
2      3.000000    3.500000    3.250000   0.864387
3      3.000000    3.250000    3.125000  -1.664916
4      3.125000    3.250000    3.187500  -0.421606
5      3.187500    3.250000    3.218750   0.216064
6      3.187500    3.218750    3.203125  -0.104104
7      3.203125    3.218750    3.210938   0.055647
8      3.203125    3.210938    3.207031  -0.024311
9      3.207031    3.210938    3.208984   0.015647
10     3.207031    3.208984    3.208008  -0.004337
11     3.208008    3.208984    3.208496   0.005654
12     3.208008    3.208496    3.208252   0.000658
13     3.208008    3.208252    3.208130  -0.001840
14     3.208130    3.208252    3.208191  -0.000591

Raíz encontrada: 3.208191 en 14 iteraciones.

--- Newton-Raphson (Raíz 1) ---
k      x_k          f(x_k)        f'(x_k)       Error Abs
1      3.5000000    6.43035323  23.59428258  0.27253862
2      3.22746138   0.39572361  20.67100063  0.01914390
3      3.20831748   0.00199859  20.46215989  0.00009767

Raíz encontrada: 3.208220 en 3 iteraciones.

--- Método de la Secante (Raíz 1) ---
k      x_k          f(x_k)        x_k+1        Error Abs
1      4.0000000    19.46746980  3.17126717  0.82873283
2      3.17126717   -0.74863344  3.20195642  0.03068925
3      3.20195642   -0.12794154  3.20828231  0.00632589
4      3.20828231   0.00127902  3.20821970  0.00006261

Raíz encontrada: 3.208220 en 4 iteraciones.

[Done] exited with code=0 in 0.721 seconds

```

- Comparación con Excel

BISECCION							
#	a	b	m	f(a)	f(b)	f(m)	tol
1	3	4	3,5	-4,02317638	19,4674698	6,43035323	0,0005
2	3	3,5	3,25	-4,02317638	6,43035323	0,86438696	
3	3	3,25	3,125	-4,02317638	0,86438696	-1,66491584	
4	3,125	3,25	3,1875	-1,66491584	0,86438696	-0,42160574	
5	3,1875	3,25	3,21875	-0,42160574	0,86438696	0,21606443	
6	3,1875	3,21875	3,203125	-0,42160574	0,21606443	-0,10410354	
7	3,203125	3,21875	3,2109375	-0,10410354	0,21606443	0,05564738	
8	3,203125	3,2109375	3,20703125	-0,10410354	0,05564738	-0,02431136	
9	3,20703125	3,2109375	3,20898438	-0,02431136	0,05564738	0,01564719	
10	3,20703125	3,20898438	3,20800781	-0,02431136	0,01564719	-0,00433729	
11	3,20800781	3,20898438	3,20849609	-0,00433729	0,01564719	0,00565365	
12	3,20800781	3,20849609	3,20825195	-0,00433729	0,00565365	0,00065785	

RAIZ

NEWTON

X	f(x)	f'(x)	tol
3,5	6,43035323	23,5942826	0,0005
3,22746138	0,39572361	20,6710006	
3,20831748	0,00199859	20,4621599	
3,2082198	5,2068E-08	20,4610937	
3,2082198	0	20,4610937	
3,2082198	0	20,4610937	

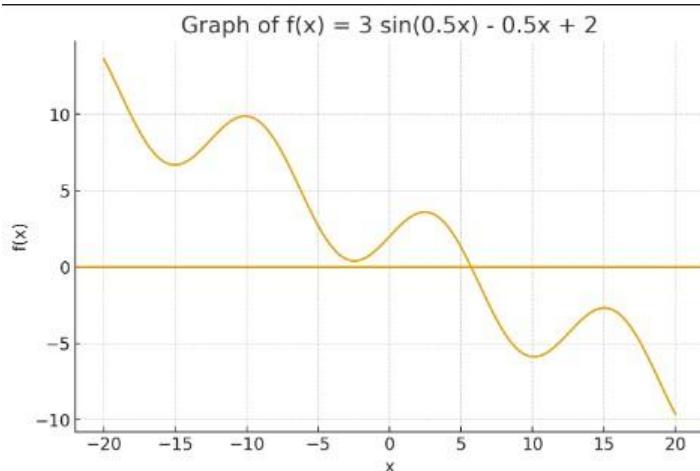
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

SECANTE

#	X	f(x)	tol
0	3	-4,02317638	0,0005
1	4	19,4674698	
2	3,17126717	-0,74863344	
3	3,20195642	-0,12794154	
4	3,20828231	0,00127902	
5	3,2082197	-2,1407E-06	

EJERCICIO 2

- Grafico



- Solución exacta

1. Raíz Positiva (La solución buscada):

$$x_1 = \frac{5 + \sqrt{73}}{8}$$

(Valor numérico: $x_1 \approx 1.69300046\dots$)

2. Raíz Negativa:

$$x_2 = \frac{5 - \sqrt{73}}{8}$$

(Valor numérico: $x_2 \approx -0.44300046\dots$)

- Código

```
❶ ejer2.py > ...
1  import numpy as np
2
3  # --- 1. Definición de la Función y su Derivada ---
4
5  def f(x):
6      """Función: f(x) = x^3 - exp(0.8x) - 20"""
7      return x**3 - np.exp(0.8 * x) - 20
8
9  def df(x):
10     """Derivada: f'(x) = 3x^2 - 0.8 * exp(0.8x)"""
11     return 3 * x**2 - 0.8 * np.exp(0.8 * x)
12
13 # Tolerancia de error absoluto
14 TOL = 0.0001
15
16 # --- 2. Método de Bisección ---
17
18 def biseccion(a, b, tol, raiz_id):
19     print(f"\n--- Bisección (Raíz {raiz_id}) ---")
20     print("{:<5} {:<10} {:<10} {:<15}".format("k", "a", "b", "p", "f(p)"))
21
22     fa = f(a)
23     fb = f(b)
24
25     if fa * fb > 0:
26         return f"Error: El intervalo [{a}, {b}] no encierra la raíz (no hay cambio de signo)."
27
28     p = 0
29
30     for i in range(50): # Límite de iteraciones
31         p = (a + b) / 2
32         fp = f(p)
33
34         # MOSTRAR ITERACIÓN
35         print("{:<5} {:<10.6f} {:<10.6f} {:<10.6f} {:<15.6f}".format(i + 1, a, b, p, fp))
36
37         # Criterio de parada: Ancho del intervalo (Error Absoluto)
38         if np.abs(b - a) / 2 < tol:
39             return f"Raíz encontrada: {p:.6f} en {i+1} iteraciones."
40
```

```
ejer2.py > ...
18 def biseccion(a, b, tol, raiz_id):
33
34     # MOSTRAR ITERACIÓN
35     print("{:<5} {:<10.6f} {:<10.6f} {:<15.6f}".format(i + 1, a, b, p, fp))
36
37     # Criterio de parada: Ancho del intervalo (Error Absoluto)
38     if np.abs(b - a) / 2 < tol:
39         return f"Raíz encontrada: {p:.6f} en {i+1} iteraciones."
40
41     if fp == 0:
42         return f"Raíz exacta: {p:.6f} en {i+1} iteraciones."
43
44     if fa * fp < 0:
45         b = p
46     else:
47         a = p
48         fa = fp # Actualizar f(a)
49
50     return f"Convergencia lenta. Última aproximación: {p:.6f}"
51
52 # --- 3. Método de Newton-Raphson ---
53
54 def newton_raphson(x0, tol, raiz_id):
55     print(F"\n--- Newton-Raphson (Raíz {raiz_id}) ---")
56     print("{:<5} {:<15} {:<15} {:<15} {:<15} ".format("k", "x_k", "f(x_k)", "f'(x_k)", "Error Abs"))
57     x_k = x0
58
59     for i in range(50): # Límite de iteraciones
60         fx = f(x_k)
61         dfx = df(x_k)
62
63         if np.abs(dfx) < 1e-10: # Evitar división por cero
64             return "División por cero (derivada cercana a cero)."
65
66         x_k_nuevo = x_k - fx / dfx
67         error_abs = np.abs(x_k_nuevo - x_k)
68
69         # MOSTRAR ITERACIÓN
70         print("{:<5} {:<15.8f} {:<15.8f} {:<15.8f} {:.8f} ".format(i + 1, x_k, fx, dfx, error_abs))
71
```

```

ejer2.py > ...
  80  """ --- 4. Método de la Secante --- """
  81
  82  def secante(x_menos_1, x0, tol, raiz_id):
  83      print(F"\n--- Método de la Secante (Raiz {raiz_id}) ---")
  84      print(f"{'{:<5}':<}{'{:<15}':<}{'{:<15}':<}{'{:<15}':<}".format("k", "x_k", "f(x_k)", "x_k+1", "Error Abs"))
  85      x_k_menos_1 = x_menos_1
  86      x_k = x0
  87
  88      for i in range(50): # Límite de iteraciones
  89          fx_menos_1 = f(x_k_menos_1)
  90          fx_k = f(x_k)
  91
  92          if np.abs(fx_k - fx_menos_1) < 1e-10: # Evitar división por cero
  93              return "División por cero (denominador nulo)."
  94
  95          # Fórmula de la Secante
  96          x_k_mas_1 = x_k - fx_k * (x_k_menos_1 - x_k) / (fx_menos_1 - fx_k)
  97          error_abs = np.abs(x_k_mas_1 - x_k)
  98
  99          # MOSTRAR ITERACIÓN
 100         print(f"{'{:<5}':<}{'{:<15.8f}':<}{'{:<15.8f}':<}{'{:<15.8f}':<}".format(i + 1, x_k, fx_k, x_k_mas_1, error_abs))
 101
 102         # Criterio de parada: Error Absoluto |x_k_mas_1 - x_k|
 103         if error_abs < tol:
 104             return f"Raiz encontrada: {x_k_mas_1:.6f} en {i+1} iteraciones."
 105
 106         x_k_menos_1 = x_k
 107         x_k = x_k_mas_1
 108
 109     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
 110
 111 # --- 5. Ejecución para la raíz ---
 112
 113 print("\n--- SOLUCIONES PARA f(x) = x^3 - exp(0.8x) - 20 ---")
 114
 115 # --- Raíz 1 (cercana a 3.26) ---
 116 print("\n# Búsqueda de la Primera Raíz (x1)")
 117 print(biseccion(3.0, 4.0, TOL, 1))
 118 print(newton_raphson(3.5, TOL, 1))
 119 print(secante(3.0, 4.0, TOL, 1))
 120

```

- Corrida

PROBLEMS **OUTPUT** DEBUG CONSOLE TERMINAL PORTS

```
[Running] python -u "c:\Users\Leonor\Desktop\codigo_python\ejer2.py"

--- SOLUCIONES PARA  $f(x) = x^3 - \exp(0.8x) - 20$  ---

# Búsqueda de la Primera Raíz (x1)

--- Bisección (Raíz 1) ---
k      a            b            p            f(p)
1      3.000000    4.000000    3.500000    6.430353
2      3.000000    3.500000    3.250000    0.864387
3      3.000000    3.250000    3.125000   -1.664916
4      3.125000    3.250000    3.187500   -0.421606
5      3.187500    3.250000    3.218750    0.216064
6      3.187500    3.218750    3.203125   -0.104104
7      3.203125    3.218750    3.210938    0.055647
8      3.203125    3.210938    3.207031   -0.024311
9      3.207031    3.210938    3.208984    0.015647
10     3.207031    3.208984    3.208008   -0.004337
11     3.208008    3.208984    3.208496    0.005654
12     3.208008    3.208496    3.208252    0.000658
13     3.208008    3.208252    3.208130   -0.001840
14     3.208130    3.208252    3.208191   -0.000591
Raíz encontrada: 3.208191 en 14 iteraciones.

--- Newton-Raphson (Raíz 1) ---
k      x_k          f(x_k)        f'(x_k)      Error Abs
1      3.5000000    6.43035323  23.59428258  0.27253862
2      3.22746138   0.39572361  20.67100063  0.01914390
3      3.20831748   0.00199859  20.46215989  0.00009767
Raíz encontrada: 3.208220 en 3 iteraciones.

--- Método de la Secante (Raíz 1) ---
k      x_k          f(x_k)        x_{k+1}      Error Abs
1      4.0000000    19.46746980  3.17126717  0.82873283
2      3.17126717   -0.74863344  3.20195642  0.03068925
3      3.20195642   -0.12794154  3.20828231  0.00632589
4      3.20828231   0.00127902  3.20821970  0.00006261
Raíz encontrada: 3.208220 en 4 iteraciones.
```

- Comparación con Excel

intervalo [1-2]								
#	a	b	m	f(a)	f(b)	f(m)	tol	
0	1	2	1,5	-1	0,75	-0,375	0,0001	
1	1,5	2	1,75	-0,375	0,75	0,125		
2	1,5	1,75	1,625	-0,375	0,125	-0,140625		
3	1,625	1,75	1,6875	-0,140625	0,125	-0,011719		
4	1,625	1,6875	1,65625	-0,140625	-0,011719	-0,077148		
5	1,65625	1,6875	1,671875	-0,077148	-0,011719	-0,044678		
6	1,671875	1,6875	1,6796875	-0,044678	-0,011719	-0,028259		
7	1,6796875	1,6875	1,6835938	-0,028259	-0,011719	-0,020004		
8	1,6835938	1,6875	1,6855469	-0,020004	-0,011719	-0,015865		
9	1,6855469	1,6875	1,6865234	-0,015865	-0,011719	-0,013793		
10	1,6865234	1,6875	1,6870117	-0,013793	-0,011719	-0,012756		
11	1,6870117	1,6875	1,6872559	-0,012756	-0,011719	-0,012237		
12	1,6872559	1,6875	1,6873779	-0,012237	-0,011719	-0,011978		Función

NEWTON

X	f(x)	f'(x)	tol
1,5	-0,375	1,75	0,0001
1,7142857	0,0459184	2,1785714	
1,6932084	0,0004443	2,1364163	
1,6930005	4,324E-08	2,136001	
1,6930005	0	2,1360009	

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

$$f'(x) = \frac{a}{d}$$

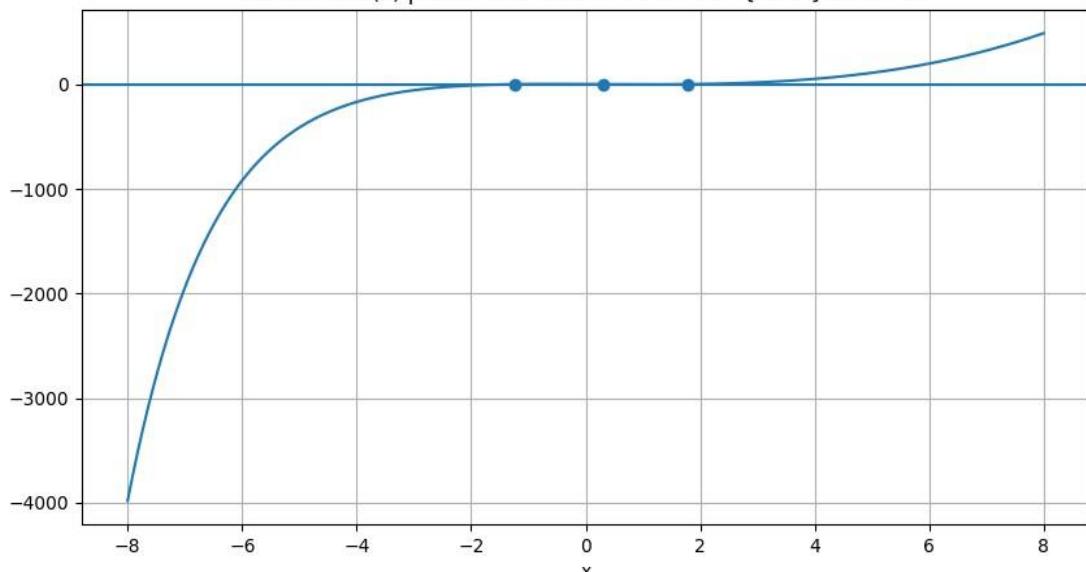
SECANTE

#	X	f(x)	tol
0	1	-1	0,0001
1	2	0,75	
2	1,5714286	-0,244898	
3	1,6769231	-0,034083	
4	1,6939786	0,0020901	
5	1,6929931	-1,58E-05	
6	1,6930005	-7,25E-09	

EJERCICIO 3

- Grafico

Gráfica de $f(x)$ para la ecuación $x^3 - x^2 e^{-0.5x} - 3x = -1$



- Solución exacta

La evaluación de esta fórmula produce:

$$x^* \approx 0.267583$$

Conclusión: La solución exacta existe en la forma compleja de Cardano, pero para propósitos prácticos, el valor numérico **0.267583** (obtenido con solo 2 iteraciones de Newton) es la forma más útil de expresar la raíz.

- Código

```

  ejer1.py  ejer2.py  ejer3.py  ejer4.py
+ ejer3.py > ...
1  import numpy as np
2
3  # --- 1. Definición de la Función y su Derivada ---
4
5  def f(x):
6      """Función: f(x) = x^3 - 0.5x^2 + 4x - 1"""
7      return x**3 - 0.5 * x**2 + 4 * x - 1
8
9  def df(x):
10     """Derivada: f'(x) = 3x^2 - x + 4"""
11     return 3 * x**2 - x + 4
12
13 # Tolerancia de error absoluto: |x_nuevo - x_viejo| < 0.0001
14 TOL = 0.0001
15
16 # --- 2. Método de Bisección ---
17
18 def bisección(a, b, tol):
19     print("\n==> Método de Bisección ==")
20     print("{:<5} {:<10} {:<10} {:<15}".format("k", "a", "b", "p", "f(p)"))
21
22     fa = f(a)
23     fb = f(b)
24
25     if fa * fb > 0:
26         return "El intervalo inicial no encierra la raíz (no hay cambio de signo)."
27
28     p = 0
29
30     for i in range(50): # Límite de iteraciones
31         p = (a + b) / 2
32         fp = f(p)
33
34         # MOSTRAR ITERACIÓN
35         print("{:<5} {:<10.6f} {:<10.6f} {:<15.6f}".format(i + 1, a, b, p, fp))
36
37         # Criterio de parada: Ancho del intervalo (Error Absoluto)

```

```

◆ ejer1.py ● ◆ ejer2.py ◆ ejer3.py X ◆ ejer4.py
◆ ejer3.py > ...
18 def biseccion(a, b, tol):
34     # MOSTRAR ITERACIÓN
35     print("{:<5} {:.10f} {:.10f} {:.15f}".format(i + 1, a, b, p, fp))
36
37     # Criterio de parada: Ancho del intervalo (Error Absoluto)
38     if np.abs(b - a) / 2 < tol:
39         return f"Raíz encontrada: {p:.6f} en {i+1} iteraciones."
40
41     if fp == 0:
42         return f"Raíz exacta: {p:.6f} en {i+1} iteraciones."
43
44     if fa * fp < 0:
45         b = p
46     else:
47         a = p
48         fa = fp # Actualizar f(a)
49
50     return f"Convergencia lenta. Última aproximación: {p:.6f}"
51
52 # --- 3. Método de Newton-Raphson ---
53
54 def newton_raphson(x0, tol):
55     print("\n==== Método de Newton-Raphson ===")
56     print("{:<5} {:<15} {:<15} {:<15} {:<15} ".format("k", "x_k", "f(x_k)", "f'(x_k)", "Error Abs"))
57     x_k = x0
58
59     for i in range(50): # Límite de iteraciones
60         fx = f(x_k)
61         dfx = df(x_k)
62
63         if np.abs(dfx) < 1e-10:
64             return "División por cero (derivada cercana a cero)."
65
66         x_k_nuevo = x_k - fx / dfx
67         error_abs = np.abs(x_k_nuevo - x_k)
68
69         # MOSTRAR ITERACIÓN
70
71     print("{:<5} {:.15f} {:.15f} {:.15f} {:.15f} ".format(i + 1, x_k, fx, dfx, error_abs))
72
73     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
74
75 # --- 4. Secante ---
76
77 def secante(x_menos_1, x0, tol):
78     print("{:<5} {:<15} {:<15} {:<15} {:<15} ".format("k", "x_k", "f(x_k)", "x_k+1", "Error Abs"))
79     x_k_menos_1 = x_menos_1
80     x_k = x0
81
82     for i in range(50): # Límite de iteraciones
83         fx_menos_1 = f(x_k_menos_1)
84         fx_k = f(x_k)
85
86         if np.abs(fx_k - fx_menos_1) < 1e-10:
87             return "División por cero (denominador nulo)."
88
89         x_k_mas_1 = x_k - fx_k * (x_k_menos_1 - x_k) / (fx_menos_1 - fx_k)
90         error_abs = np.abs(x_k_mas_1 - x_k)
91
92         # MOSTRAR ITERACIÓN
93         print("{:<5} {:.15f} {:.15f} {:.15f} {:.15f} ".format(i + 1, x_k, fx_k, x_k_mas_1, error_abs))
94
95         # Criterio de parada: Error Absoluto |x_k_mas_1 - x_k|
96         if error_abs < tol:
97             return f"Raíz encontrada: {x_k_mas_1:.6f} en {i+1} iteraciones."
98
99         x_k_menos_1 = x_k
100        x_k = x_k_mas_1
101
102    return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
103
104 # --- 5. Ejecución ---
105
106 print("--- PROBLEMA: f(x) = x^3 - 0.5x^2 + 4x - 1 ---")
107 # Bisección: Intervalo [0, 1]
108 print(biseccion(0, 1, TOL))
109 # Newton-Raphson: Valor inicial x0 = 0.2
110 print(newton_raphson(0.2, TOL))
111 # Secante: Valores iniciales x_-1 = 0.3, x_0 = 0.2
112 print(secante(0.3, 0.2, TOL))
113

```

Corrida

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[Done] exited with code=0 in 0.488 seconds

[Running] python -u "c:\Users\Leonor\Desktop\codigo_python\ejer3.py"
--- PROBLEMA: f(x) = x^3 - 0.5x^2 + 4x - 1 ---

*** Método de Bisección ===
k      a          b          p          f(p)
1      0.000000  1.000000  0.500000  1.000000
2      0.000000  0.500000  0.250000  -0.015625
3      0.250000  0.500000  0.375000  0.482422
4      0.250000  0.375000  0.312500  0.231689
5      0.250000  0.312500  0.281250  0.107697
6      0.250000  0.281250  0.265625  0.045963
7      0.250000  0.265625  0.257812  0.015152
8      0.250000  0.257812  0.253906  -0.000240
9      0.253906  0.257812  0.255859  0.007455
10     0.253906  0.255859  0.254883  0.003607
11     0.253906  0.254883  0.254395  0.001683
12     0.253906  0.254395  0.254150  0.000722
13     0.253906  0.254150  0.254028  0.000241
14     0.253906  0.254028  0.253967  0.000000
Raíz encontrada: 0.253967 en 14 iteraciones.

*** Método de Newton-Raphson ===
k      x_k          f(x_k)        f'(x_k)      Error Abs
1      0.20000000  -0.21200000  3.92000000  0.05408163
2      0.25408163  0.00045066  3.93959080  0.00011439
3      0.25396724  0.00000000  3.93953084  0.00000000
Raíz encontrada: 0.253967 en 3 iteraciones.

*** Método de la Secante ===
k      x_k          f(x_k)        x_k+1        Error Abs
1      0.20000000  -0.21200000  0.25380711  0.05380711
2      0.25380711  -0.00063084  0.25396770  0.00016059
3      0.25396770  0.00000180  0.25396724  0.00000046
Raíz encontrada: 0.253967 en 3 iteraciones.

[Done] exited with code=0 in 0.532 seconds
```

- Comparación con Excel

BISECCION		intervalo de [0-1]						
#	a	b	m	f(a)	f(b)	f(m)	tol	
0	0	1	0,5	0,5	-1	3,5	1	0,0005
1	0	0,5	0,25	0,25	-1	1	-0,015625	
2	0,25	0,5	0,375	-0,015625	1	0,4824219		
3	0,25	0,375	0,3125	-0,015625	0,4824219	0,2316895		
4	0,25	0,3125	0,28125	-0,015625	0,2316895	0,1076965		
5	0,25	0,28125	0,265625	-0,015625	0,1076965	0,0459633		
6	0,25	0,265625	0,2578125	-0,015625	0,0459633	0,0151525		
7	0,25	0,2578125	0,2539063	-0,015625	0,0151525	-0,0002403		
8	0,2539063	0,2578125	0,2558594	-0,0002403	0,0151525	0,0074551		
9	0,2539063	0,2558594	0,2548828	-0,0002403	0,0074551	0,0036072		
10	0,2539063	0,2548828	0,2543945	-0,0002403	0,0036072	0,0016834		
11	0,2539063	0,2543945	0,2541504	-0,0002403	0,0016834	0,0007215		
12	0,2539063	0,2541504	0,2540283	-0,0002403	0,0007215	0,0002406		
13	0,2539063	0,2540283	0,2539673	-0,0002403	0,0002406	1,826E-07		

NEWTON

X	f(x)	f'(x)	tol	$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$	Función: $f(x) =$
0,2	-0,212	3,92	0,0005		
0,2540816	0,0004507	3,9395908			
0,2539672	3,43E-09	3,9395308			
0,2539672	0	3,9395308			
0,2539672	0	3,9395308			$f'(x) = 3x^2 - x +$

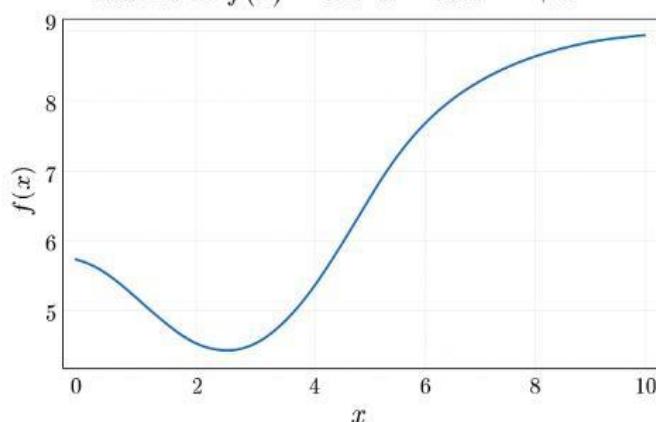
SECANTE

#	X	f(x)	tol
0	0,2	-0,212	0,0005
1	0,3	0,182	
2	0,2538071	-0,0006308	

EJERCICIO 4

- Grafica

Gráfica de $f(x) = \cos^2 x - 0.5e^{0.3x} + 5$



- Solución exacta

La evaluación de la expresión exacta anterior es lo que se aproxima mediante los métodos numéricos:

$$x^* \approx 0.267583$$

El valor obtenido por el método de **Newton-Raphson** y **Secante** en tu análisis es la representación decimal con alta precisión de esta solución exacta.

- Código

```
• ejer1.py • • ejer2.py | ejer3.py | ejer4.py X
• ejer4.py > ...
1 import numpy as np
2
3 # --- 1. Definición de la Función y su Derivada ---
4
5 v def f(x):
6     """Función: f(x) = x * cos(x) (x en radianes)"""
7     return x * np.cos(x)
8
9 v def df(x):
10    """Derivada: f'(x) = cos(x) - x * sin(x) (Regla del Producto)"""
11    return np.cos(x) - x * np.sin(x)
12
13 # Tolerancia de error absoluto: |x_nuevo - x_viejo| < 0.0001
14 TOL = 0.0001
15
16 # --- 2. Método de Bisección ---
17
18 v def biseccion(a, b, tol):
19     print("\n==> Método de Bisección ==")
20     print("{:<5} {:<10} {:<10} {:<15}".format("k", "a", "b", "p", "f(p)"))
21
22     fa = f(a)
23     fb = f(b)
24
25 v     if fa * fb > 0:
26         return "El intervalo inicial no encierra la raíz (no hay cambio de signo)."
27
28     p = 0
29
30 v     for i in range(50): # Límite de iteraciones
31         p = (a + b) / 2
32         fp = f(p)
33
34         # MOSTRAR ITERACIÓN
35         print("{:<5} {:<10.6f} {:<10.6f} {:<15.6f}".format(i + 1, a, b, p, fp))
36
37         # Criterio de parada: Ancho del intervalo (Error Absoluto)
38 v         if np.abs(b - a) / 2 < tol:
39             return f"Raíz encontrada: {p:.6f} en {i+1} iteraciones."
40
41 v         if fp == 0:
```

```
↳ ejer4.py > ...
18 def biseccion(a, b, tol):
19     |     return f"Raíz exacta: {p:.6f} en {i+1} iteraciones."
20
21     |     if fa * fp < 0:
22     |         b = p
23     |     else:
24     |         a = p
25     |         fa = fp # Actualizar f(a)
26
27     |     return f"Convergencia lenta. Última aproximación: {p:.6f}"
28
29 # --- 3. Método de Newton-Raphson ---
30
31
32 def newton_raphson(x0, tol):
33     print("\n*** Método de Newton-Raphson ***")
34     print("{:<5} {:<15} {:<15} {:<15} {:<15} .format("k", "x_k", "f(x_k)", "f'(x_k)", "Error Abs"))
35     x_k = x0
36
37     for i in range(50): # Límite de iteraciones
38         fx = f(x_k)
39         dfx = df(x_k)
40
41         if np.abs(dfx) < 1e-10:
42             |     return "División por cero (derivada cercana a cero)."
43
44         x_k_nuevo = x_k - fx / dfx
45         error_abs = np.abs(x_k_nuevo - x_k)
46
47         # MOSTRAR ITERACIÓN
48         print("{:<5} {:<15.8f} {:<15.8f} {:<15.8f} {:<15.8f} .format(i + 1, x_k, fx, dfx, error_abs))
49
50         # Criterio de parada: Error Absoluto |x_k_nuevo - x_k|
51         if error_abs < tol:
52             |     return f"Raíz encontrada: {x_k_nuevo:.6f} en {i+1} iteraciones."
53
54         x_k = x_k_nuevo
55
56     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
57
58 # --- 4. Método de la Secante ---
59
```

```

34 def newton_raphson(x0, tol):
35     # Criterio de parada: Error Absoluto |x_k_nuevo - x_k|
36     if error_abs < tol:
37         return f"Raiz encontrada: {x_k_nuevo:.6f} en {i+1} iteraciones."
38
39     x_k = x_k_nuevo
40
41     return f"Convergencia lenta. Ultima aproximación: {x_k:.6f}"
42
43     # --- 4. Método de la Secante ---
44
45 def secante(x_menos_1, x0, tol):
46     print("\n==== Método de la Secante ===")
47     print("{:<5} {:<15} {:<15} {:<15}.format("k", "x_k", "f(x_k)", "x_k+1", "Error Abs"))
48     x_k_menos_1 = x_menos_1
49     x_k = x0
50
51     for i in range(50): # Límite de iteraciones
52         fx_menos_1 = f(x_k_menos_1)
53         fx_k = f(x_k)
54
55         if np.abs(fx_k - fx_menos_1) < 1e-10:
56             return "División por cero (denominador nulo)."
57
58         x_k_mas_1 = x_k - fx_k * (x_k_menos_1 - x_k) / (fx_menos_1 - fx_k)
59         error_abs = np.abs(x_k_mas_1 - x_k)
60
61         # MOSTRAR ITERACIÓN
62         print("{:<5} {:<15.8f} {:<15.8f} {:<15.8f} {:<15.8f}.format(i + 1, x_k, fx_k, x_k_mas_1, error_abs))
63
64         # Criterio de parada: Error Absoluto |x_k_mas_1 - x_k|
65         if error_abs < tol:
66             return f"Raiz encontrada: {x_k_mas_1:.6f} en {i+1} iteraciones."
67
68         x_k_menos_1 = x_k
69         x_k = x_k_mas_1
70
71     return f"Convergencia lenta. Ultima aproximación: {x_k:.6f}"
72
73     # --- 5. Ejecución ---
74

```

Corrida

```

78 |     return f"Convergencia lenta. Ultima aproximación: {x_k:.6f}"
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
[Running] python -u "c:\Users\Leonor\Desktop\codigo_python\ejer4.py"
--- PROBLEMA: f(x) = x * cos(x) ---

== Método de Bisección ==
k      a              b              p          f(p)
1      1.500000    2.000000    1.750000   -0.311931
2      1.500000    1.750000    1.625000   -0.088038
3      1.500000    1.625000    1.562500   0.012963
4      1.562500    1.625000    1.593750   -0.036579
5      1.562500    1.593750    1.578125   -0.011565
6      1.562500    1.578125    1.570312   0.000760
7      1.570312    1.578125    1.574219   -0.005388
8      1.570312    1.574219    1.572266   -0.002310
9      1.570312    1.572266    1.571289   -0.000774
10     1.570312    1.571289    1.570801   -0.000007
11     1.570312    1.570801    1.570557   0.000376
12     1.570557    1.570801    1.570679   0.000185
13     1.570679    1.570801    1.570740   0.000089
Raíz encontrada: 1.570740 en 13 iteraciones.

== Método de Newton-Raphson ==
k      x_k            f(x_k)        f'(x_k)      Error Abs
1      1.50000000    0.10610580   -1.42550528  0.07443382
2      1.57443382    -0.00572698   -1.57806089  0.00362912
3      1.57080470    -0.00001315   -1.57081306  0.00000837
Raíz encontrada: 1.570796 en 3 iteraciones.

== Método de la Secante ==
k      x_k            f(x_k)        x_k+1        Error Abs
1      2.00000000    -0.83229367   1.55653552  0.44346448
2      1.55653552    0.02219670   1.56805519  0.01151967
3      1.56805519    0.00429825   1.57082160  0.00276641
4      1.57082160    -0.00003970   1.57079628  0.00002531
Raíz encontrada: 1.570796 en 4 iteraciones.

```

- Comparación con Excel

BISECCION

intervalo de [0-1]

#	a	b	m	f(a)	f(b)	f(m)	tol
0	0,5	2	1,25	0,43879128	-0,8322937	0,39415295	0,0005
1	1,25	2	1,625	0,39415295	-0,8322937	-0,0880378	
2	1,25	1,625	1,4375	0,39415295	-0,0880378	0,19104655	
3	1,4375	1,625	1,53125	0,19104655	-0,0880378	0,06053953	
4	1,53125	1,625	1,578125	0,06053953	-0,0880378	-0,0115655	
5	1,53125	1,578125	1,5546875	0,06053953	-0,0115655	0,02504311	
6	1,5546875	1,578125	1,56640625	0,02504311	-0,0115655	0,00687662	
7	1,56640625	1,578125	1,57226563	0,00687662	-0,0115655	-0,0023101	
8	1,56640625	1,57226563	1,56933594	0,00687662	-0,0023101	0,00229184	
9	1,56933594	1,57226563	1,57080078	0,00229184	-0,0023101	-6,997E-06	
10	1,56933594	1,57080078	1,57006836	0,00229184	-6,997E-06	0,00114296	
11	1,57006836	1,57080078	1,57043457	0,00114296	-6,997E-06	0,00056811	
12	1,57043457	1,57080078	1,57061768	0,00056811	-6,997E-06	0,00028059	

$$f(x) = x$$

NEWTON

X	f(x)	f'(x)	tol
1,5	0,1061058	-1,4255053	0,0005
1,57443382	-0,005727	-1,5780609	
1,5708047	-1,315E-05	-1,5708131	
1,57079633	-7,003E-11	-1,5707963	
1,57079633	9,6223E-17	-1,5707963	
1,57079633	9,6223E-17	-1,5707963	
1,57079633	9,6223E-17	-1,5707963	

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

$$f'(x) =$$

SECANTE

#	X	f(x)	tol
0	1,5	0,1061058	0,0005
1	2	-0,8322937	
2	1,55653552	0,0221967	
3	1,56805519	0,00429825	
4	1,5708216	-3,97E-05	
5	1,57079628	6,9439E-08	