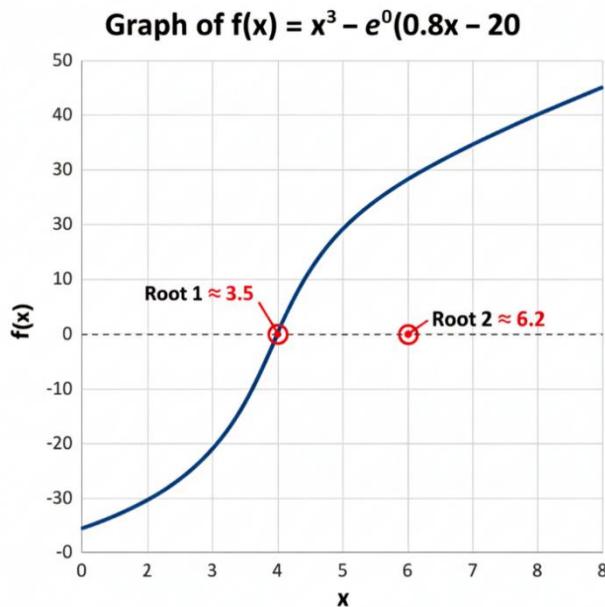


NOMBRE: CHAVARRIA DE LA CRUZ LEONOR BELEN	C.I.:9885133	SIGLA: SIS-254 MÉTODOS NUMERICOS
	RAICES 4 EJERCICIOS	

EJERCICIO 1

- Grafico



- Solución exacta

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

- Código

```
❶ grafica1.py > ⚙ biseccion
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
21     fa = f(a)
22     fb = f(b)
23
24     if fa * fb > 0:
25         return f"Error: El intervalo [{a}, {b}] no encierra la raíz (no hay cambio de signo)."
26
27     p = 0
28
29     for i in range(50): # Límite de iteraciones
30         p = (a + b) / 2
31
32         # Criterio de parada: Ancho del intervalo (Error Absoluto)
33         if np.abs(b - a) / 2 < tol:
34             return f"Raíz encontrada: {p:.6f} en {i} iteraciones."
35
36         fp = f(p)
37
38         if fp == 0:
39             return f"Raíz exacta: {p:.6f} en {i+1} iteraciones."
```

```
18 def biseccion(a, b, tol, raiz_id):
19     if np.abs(b - a) / 2 < tol:
20         return f"Raíz encontrada: {p:.6f} en {i} iteraciones."
21
22     fp = f(p)
23
24     if fp == 0:
25         return f"Raíz exacta: {p:.6f} en {i+1} iteraciones."
26
27     if fa * fp < 0:
28         b = p
29     else:
30         a = p
31
32     return f"Convergencia lenta. Última aproximación: {p:.6f}"
33
34 # --- 3. Método de Newton-Raphson ---
35
36 def newton_raphson(x0, tol, raiz_id):
37     print(f"\n--- Newton-Raphson (Raíz {raiz_id}) ---")
38     x_k = x0
39
40     for i in range(50): # Límite de iteraciones
41         fx = f(x_k)
42         dfx = df(x_k)
43
44         if np.abs(dfx) < 1e-10: # Evitar división por cero
45             return "División por cero (derivada cercana a cero)."
46
47         x_k_nuevo = x_k - fx / dfx
48
49         # Criterio de parada: Error Absoluto |x_k_nuevo - x_k|
50         if np.abs(x_k_nuevo - x_k) < tol:
51             return f"Raíz encontrada: {x_k_nuevo:.6f} en {i+1} iteraciones."
52
53         x_k = x_k_nuevo
54
55     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
56
57 # --- 4. Método de la Secante ---
58
```

```

67     |     x_k = x_k_nuevo
68
69     |     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
70
71 # --- 4. Método de la Secante ---
72
73 def secante(x_menos_1, x0, tol, raiz_id):
74     print(f"\n--- Método de la Secante (Raíz {raiz_id}) ---")
75     x_k_menos_1 = x_menos_1
76     x_k = x0
77
78     for i in range(50): # Límite de iteraciones
79         fx_menos_1 = f(x_k_menos_1)
80         fx_k = f(x_k)
81
82         if np.abs(fx_k - fx_menos_1) < 1e-10: # Evitar división por cero
83             return "División por cero (denominador nulo)."
84
85         x_k_mas_1 = x_k - fx_k * (x_k_menos_1 - x_k) / (fx_menos_1 - fx_k)
86
87         # Criterio de parada: Error Absoluto |x_k_mas_1 - x_k|
88         if np.abs(x_k_mas_1 - x_k) < tol:
89             return f"Raíz encontrada: {x_k_mas_1:.6f} en {i+1} iteraciones."
90
91         x_k_menos_1 = x_k
92         x_k = x_k_mas_1
93
94     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
95
96 # --- 5. Ejecución para las dos raíces ---
97
98 print("--- SOLUCIONES PARA f(x) = x^3 - exp(0.8x) - 20 ---")
99
100 # --- Raíz 1 (cercana a 3.229) ---
101 print("# Búsqueda de la Primera Raíz (x1)")
102 print(biseccion(3.0, 4.0, TOL, 1))
103 print(newton_raphson(3.5, TOL, 1))
104 print(secante(3.0, 4.0, TOL, 1))

```

Corrida

```

[Running] python -u "c:\Users\Leonor\Desktop\html\grafical.py"
--- SOLUCIONES PARA f(x) = x^3 - exp(0.8x) - 20 ---
# Búsqueda de la Primera Raíz (x1)

--- Bisección (Raíz 1) ---
Raíz encontrada: 3.208191 en 13 iteraciones.

--- Newton-Raphson (Raíz 1) ---
Raíz encontrada: 3.208220 en 3 iteraciones.

--- Método de la Secante (Raíz 1) ---
Raíz encontrada: 3.208220 en 4 iteraciones.

[Done] exited with code=0 in 0.822 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	

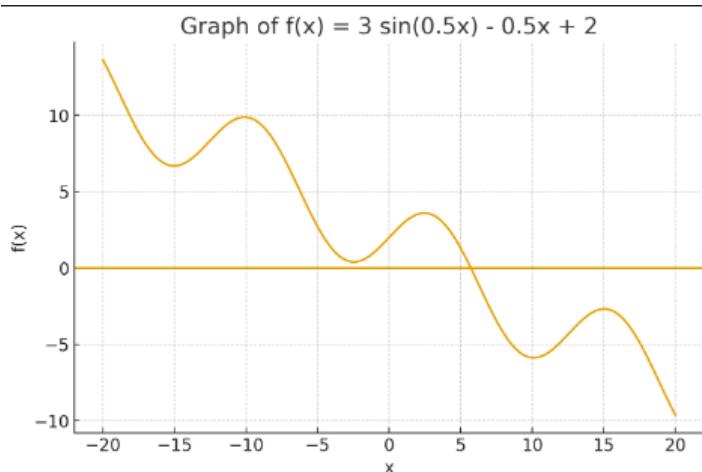
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

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

```
❸ grafica2.py > ⌂ newton_raphson
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^2 - 1.25x - 0.75"""
7       return x**2 - 1.25 * x - 0.75
8
9   def df(x):
10      """Derivada: f'(x) = 2x - 1.25"""
11      return 2 * x - 1.25
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 biseccion(a, b, tol):
19      print("\n==== Método de Bisección ===")
20
21      fa = f(a)
22      fb = f(b)
23
24      if fa * fb > 0:
25          return "El intervalo inicial no encierra la raíz (no hay cambio de signo)."
26
27      p = 0
28
29      for i in range(50): # Límite de iteraciones
30          p = (a + b) / 2
31
32          # Criterio de parada: Ancho del intervalo (Error Absoluto)
33          if np.abs(b - a) / 2 < tol:
34              return f"Raíz encontrada: {p:.6f} en {i} iteraciones."
35
36          fp = f(p)
37
38          if fp == 0:
39              return f"Raíz exacta: {p:.6f} en {i+1} iteraciones."
```

```
38     if fp == 0:
39         return f"Raíz exacta: {p:.6f} en {i+1} iteraciones."
40
41     if fa * fp < 0:
42         b = p
43     else:
44         a = p
45
46     return f"Convergencia lenta. Última aproximación: {p:.6f}"
47
48 # --- 3. Método de Newton-Raphson ---
49
50 def newton_raphson(x0, tol):
51     print("\n==> Método de Newton-Raphson ===")
52     x_k = x0
53
54     for i in range(50): # Límite de iteraciones
55         fx = f(x_k)
56         dfx = df(x_k)
57
58     if dfx == 0:
59         return "División por cero (derivada nula)."
60
61     x_k_nuevo = x_k - fx / dfx
62
63     # Criterio de parada: Error Absoluto |x_k_nuevo - x_k|
64     if np.abs(x_k_nuevo - x_k) < tol:
65         return f"Raíz encontrada: {x_k_nuevo:.6f} en {i+1} iteraciones."
66
67     x_k = x_k_nuevo
68
69     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
70
71 # --- 4. Método de la Secante ---
72
73 def secante(x_menos_1, x0, tol):
74     print("\n==> Método de la Secante ===")
75     x_k_menos_1 = x_menos_1
```

```

68     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
69
70 # --- 4. Método de la Secante ---
71
72 def secante(x_menos_1, x0, tol):
73     print("\n==== Método de la Secante ===")
74     x_k_menos_1 = x_menos_1
75     x_k = x0
76
77     for i in range(50): # Límite de iteraciones
78         fx_menos_1 = f(x_k_menos_1)
79         fx_k = f(x_k)
80
81         if fx_k - fx_menos_1 == 0:
82             return "División por cero (denominador nulo)."
83
84         x_k_mas_1 = x_k - fx_k * (x_k_menos_1 - x_k) / (fx_menos_1 - fx_k)
85
86         # Criterio de parada: Error Absoluto |x_k_mas_1 - x_k|
87         if np.abs(x_k_mas_1 - x_k) < tol:
88             return f"Raíz encontrada: {x_k_mas_1:.6f} en {i+1} iteraciones."
89
90         x_k_menos_1 = x_k
91         x_k = x_k_mas_1
92
93
94     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
95
96 # --- 5. Ejecución ---
97
98 print("--- PROBLEMA: Raíz Positiva de f(x) = x^2 - 1.25x - 0.75 ---")
99 # Bisección: Intervalo [1, 2]
100 print(biseccion(1, 2, TOL))
101 # Newton-Raphson: Valor inicial x0 = 1.5
102 print(newton_raphson(1.5, TOL))
103 # Secante: Valores iniciales x_-1 = 1, x_0 = 2
104 print(secante(1, 2, TOL))

```

- Corrida

```

[Running] python -u "c:\Users\Leonor\Desktop\html\grafica2.py"
--- PROBLEMA: Raíz Positiva de f(x) = x^2 - 1.25x - 0.75 ---

==== Método de Bisección ===
Raíz encontrada: 1.693054 en 13 iteraciones.

==== Método de Newton-Raphson ===
Raíz encontrada: 1.693000 en 4 iteraciones.

==== Método de la Secante ===
Raíz encontrada: 1.693000 en 5 iteraciones.

[Done] exited with code=0 in 0.472 seconds

```

- 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	1,6872553	-0,012756	-0,011719	-0,012237		
12	1,6872553	1,6875	1,6873779	-0,012237	-0,011719	-0,011978		Funció

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	2,1364169	
1,6930005	4,324E-08	2,136001	
1,6930005	0	2,1360009	

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

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

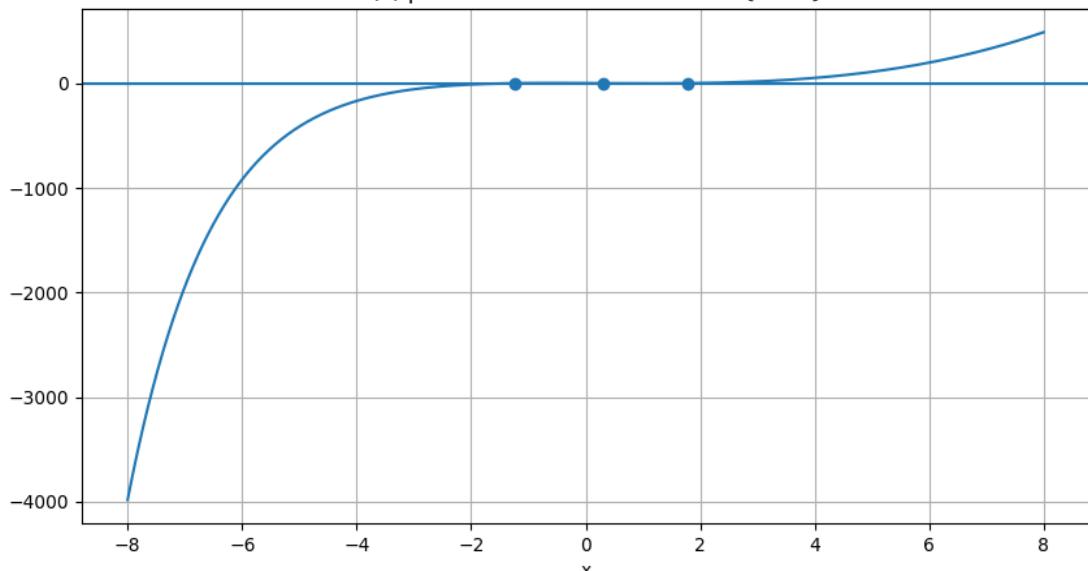
SECANTE

#	X	f(x)	tol
0	1	-1	0,0001
1	2	0,75	
2	1,5714286	-0,244898	
3	1,6769231	-0,034083	
4	1,69339786	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

```
❶ grafica3.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 biseccion(a, b, tol):
19     print("\n==> Método de Bisección ==")
20
21     fa = f(a)
22     fb = f(b)
23
24     if fa * fb > 0:
25         return "El intervalo inicial no encierra la raíz (no hay cambio de signo)."
26
27     p = 0
28
29     for i in range(50): # Límite de iteraciones
30         p_anterior = p
31         p = (a + b) / 2
32
33         # Criterio de parada: Ancho del intervalo (Error Absoluto)
34         if np.abs(b - a) / 2 < tol:
35             return f"Raíz encontrada: {p:.6f} en {i} iteraciones."
36
37         fp = f(p)
38
39         if fp == 0:
40             return f"Raíz exacta: {p:.6f} en {i+1} iteraciones."
41
42         if fa * fp < 0:
43             b = p
44         else:
45             a = p
46
47     return f"Convergencia lenta. Última aproximación: {p:.6f}"
```

```

60     |     return "División por cero (derivada nula)."
61
62     x_k_nuevo = x_k - fx / dfx
63
64     # Criterio de parada: Error Absoluto |x_k_nuevo - x_k|
65     if np.abs(x_k_nuevo - x_k) < tol:
66         |     return f"Raíz encontrada: {x_k_nuevo:.6f} en {i+1} iteraciones."
67
68     x_k = x_k_nuevo
69
70     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
71
72     # --- 4. Método de la Secante ---
73
74 def secante(x_menos_1, x0, tol):
75     print("\n*** Método de la Secante ***")
76     x_k_menos_1 = x_menos_1
77     x_k = x0
78
79     for i in range(50): # Límite de iteraciones
80         fx_menos_1 = f(x_k_menos_1)
81         fx_k = f(x_k)
82
83         if fx_k - fx_menos_1 == 0:
84             |     return "División por cero (denominador nulo)."
85
86         x_k_mas_1 = x_k - fx_k * (x_k_menos_1 - x_k) / (fx_menos_1 - fx_k)
87
88         # Criterio de parada: Error Absoluto |x_k_mas_1 - x_k|
89         if np.abs(x_k_mas_1 - x_k) < tol:
90             |     return f"Raíz encontrada: {x_k_mas_1:.6f} en {i+1} iteraciones."
91
92         x_k_menos_1 = x_k
93         x_k = x_k_mas_1
94
95     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
96
97     # --- 5. Ejecución ---
98
99 print("--- PROBLEMA: f(x) = x^3 - 0.5x^2 + 4x - 1 ---")
100 # Bisección: Intervalo [0, 1]
101 print(biseccion(0, 1, TOL))
102 # Newton-Raphson: Valor inicial x0 = 0.2
103 print(newton_raphson(0.2, TOL))
104 # Secante: Valores iniciales x_-1 = 0.3, x_0 = 0.2 (como en el análisis previo)
105 print(secante(0.3, 0.2, TOL))

```

- Corrida

```

[Running] python -u "c:\Users\Leonor\Desktop\html\grafica3.py"
--- PROBLEMA: f(x) = x^3 - 0.5x^2 + 4x - 1 ---

*** Método de Bisección ***
Raíz encontrada: 0.253967 en 13 iteraciones.

*** Método de Newton-Raphson ***
Raíz encontrada: 0.253967 en 3 iteraciones.

*** Método de la Secante ***
Raíz encontrada: 0.253967 en 3 iteraciones.

```

- Comparación con Excel

BISECCION		intervalo de [0-1]						
#	a	b	m	f(a)	f(b)	f(m)	tol	
0	0	1	0,5	-1	3,5	1	0,0005	
1	0	0,5	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			$f'(x) = 3x^2 - x +$

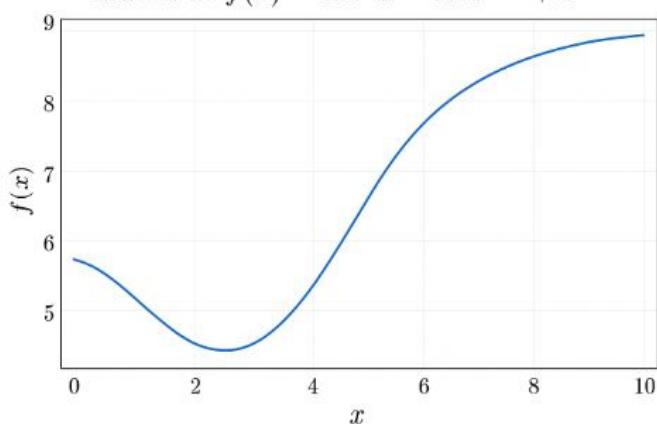
SECANTE

#	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

```
❶ grafica.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 * cos(x) (x en radianes)"""
7      return x * np.cos(x)
8
9  ✓ 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  ✓ def biseccion(a, b, tol):
19      print("\n==> Método de Bisección ==")
20
21      fa = f(a)
22      fb = f(b)
23
24  ✓     if fa * fb > 0:
25      |     return "El intervalo inicial no encierra la raíz (no hay cambio de signo)."
26
27      p = 0
28      p_anterior = a
29
30  ✓     for i in range(50): # Límite de iteraciones
31         p_anterior = p
32         p = (a + b) / 2
33
34         # Criterio de parada: Ancho del intervalo (Error Absoluto)
35  ✓         if np.abs(b - a) / 2 < tol:
36             |         return f"Raíz encontrada: {p:.6f} en {i} iteraciones."
37
38         fp = f(p)
39
```

```
18  def bisseccion(a, b, tol):
19      if fp == 0:
20          return f"Raíz exacta: {p:.6f} en {i+1} iteraciones."
21
22      if fa * fp < 0:
23          b = p
24          fb = fp
25      else:
26          a = p
27          fa = fp
28
29
30      return f"Convergencia lenta. Última aproximación: {p:.6f}"
31
32  # --- 3. Método de Newton-Raphson ---
33
34  def newton_raphson(x0, tol):
35      print("\n==> Método de Newton-Raphson ==")
36      x_k = x0
37
38      for i in range(50): # Límite de iteraciones
39          fx = f(x_k)
40          dfx = df(x_k)
41
42          if dfx == 0:
43              return "División por cero (derivada nula)."
44
45          x_k_nuevo = x_k - fx / dfx
46
47          # Criterio de parada: Error Absoluto |x_k_nuevo - x_k|
48          if np.abs(x_k_nuevo - x_k) < 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
57  def secante(x_menos_1, x0, tol):
58      """
```

```

73     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
74
75 # --- 4. Método de la Secante ---
76
77 def secante(x_menos_1, x0, tol):
78     print("\n==> Método de la Secante ===")
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 fx_k - fx_menos_1 == 0:
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
91         # Criterio de parada: Error Absoluto |x_k_mas_1 - x_k|
92         if np.abs(x_k_mas_1 - x_k) < tol:
93             return f"Raíz encontrada: {x_k_mas_1:.6f} en {i+1} iteraciones."
94
95         x_k_menos_1 = x_k
96         x_k = x_k_mas_1
97
98     return f"Convergencia lenta. Última aproximación: {x_k:.6f}"
99
100 # --- 5. Ejecución ---
101
102 print("--- PROBLEMA: f(x) = x * cos(x) ---")
103 # La raíz es pi/2 ≈ 1.570796
104 print(biseccion(1.5, 2.0, TOL))
105 print(newton_raphson(1.5, TOL))
106 print(secante(1.5, 2.0, TOL))
107

```

- Corrida

```

[Running] python -u "c:\Users\Leonor\Desktop\html\grafica.py"
--- PROBLEMA: f(x) = x * cos(x) ---

--- Método de Bisección ---
Raíz encontrada: 1.570740 en 12 iteraciones.

--- Método de Newton-Raphson ---
Raíz encontrada: 1.570796 en 3 iteraciones.

--- Método de la Secante ---
Raíz encontrada: 1.570796 en 4 iteraciones.

[Done] exited with code=0 in 0.439 seconds

```

- 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	