Comenzado el	viernes, 29 de noviembre de 2024, 10:17
Estado	Finalizado
Finalizado en	viernes, 29 de noviembre de 2024, 12:59
Tiempo empleado	2 horas 42 minutos
Calificación	**32,00** de 60,00 (**53**%)

Pregunta **1**

Parcialmente correcta

Puntúa 2,00 sobre 4,00

Los primeros tres términos diferentes a cero de la serie de Maclaurin para la función arcotangente son: $x - 1/3 * x^3 + 1/5 * x^5$.

Calcule el error relativo en las siguientes aproximaciones de π mediante el polinomio (en lugar del arcotangente).
Asuma que $\pi = 3.14159$.
$4(arctan(1/2) + arctan(1/3))$

Redondee a 4 cifras significativas únicamente en la respuesta final de sus cálculos.
$\epsilon =$ ✔.

¿En qué orden de magnitud está este error? Es decir, $\epsilon < 10^n, n =$ ✖.

$16 * arctan(1/5) - 4 * arctan(1/239)$

$\epsilon =$ ✖.

¿En qué orden de magnitud está este error? Es decir, $\epsilon < 10^n, n =$ ✔.

Pregunta **2**

Correcta

Puntúa 6,00 sobre 6,00

Suponga que dos puntos (x_0, y_0) y (x_1, y_1) se encuentran en línea recta con $y_1 \neq y_0$.

Existen dos fórmulas para encontrar la intersección x de la línea:

Método A: $x = \frac{x_0 \times y_1 - x_1 \times y_0}{y_1 - y_0}$

y

Método B: $x = x_0 - \frac{(x_1 - x_0) \times y_0}{y_1 - y_0}$

Usando los datos $(x_0, y_0) = (1.31, 3.24)$ y $(x_1, y_1) = (1.93, 4.76)$, determine el valor real de la intersección x (asumiendo redondeo a 6 cifras significativas):

$x =$ ✔

Usando aritmética de computador con redondeo a 3 cifras significativas resuelva para ambos métodos.

Usando el método A:

$x =$ ✔.

El error relativo (redondee al final del cálculo a 3 cifras significativas) del método A:

$\epsilon =$ ✔.

Usando el método B:

$x =$.

El error relativo (redondee al final del cálculo a 3 cifras significativas) del método B:

$\epsilon =$.

¿Cuál método es mejor?

Método B

Pregunta **3**

Parcialmente correcta

Puntúa 1,00 sobre 5,00

El método de Newton para encontrar raíces se basa en la siguiente ecuación:

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

Cuál es la raíz de la ecuación:

$x^3 + x = 1 + 3x^2$

$x_{sol} =$ ✖

Qué sucede cuando:

$x_0 = 3$ x_sol ✔

$x_0 = 1$ x_sol ✖

$x_0 = 0$ x_sol ✖

$x_0 = 1 + \sqrt{6}/3$ x_sol ✖

Pregunta **4**

Parcialmente correcta

Puntúa 8,00 sobre 15,00

El método de la Secante se basa en la siguiente fórmula:

$$x_n = x_{n-1} - \frac{y_{n-1}(x_{n-1} - x_{n-2})}{y_{n-1} - y_{n-2}}$$

En base a esta fórmula, se ha generado el siguiente código.

```
def secant_method(f, x0, x1, tol=1e-6, max_iter=100):
    x_prev = x0
    x_curr = x1
    iter_count = 0

    while abs(f(x_curr)) > tol and iter_count < max_iter:
        # Calculate the next approximation using the secant method formula
        x_next = x_curr - f(x_curr) * (x_curr - x_prev) / (f(x_curr) - f(x_prev))

        # Update variables for the next iteration
        x_prev = x_curr
        x_curr = x_next
        iter_count += 1

    return x_curr, iter_count
```

El código funciona correctamente. Sin embargo, al depurarlo y profundizar en su ejecución, usted ha notado que el código realiza llamadas repetitivas e innecesarias.

Esto se evidencia en la siguiente Figura:

La variable i representa el número de invocaciones a la función. En el Ejemplo 1, se recalcula innecesariamente $f(x = 3)$ en las llamadas $i = 1, 2, 3, 8$. Lo mismo sucede en $i = 5, 6, 7, 12$ para $f(x = 2.6)$. Esto ocasiona que se realicen 25 llamadas a la función en el Ejemplo 1.

```python
def func(x):
    global i
    i += 1
    y = x**3 - 3 * x**2 + x - 1
    print(f"Llamada i={i}\t x={x:.5f}\t y={y:.2f}")
    return y


secant_method(func, x0=2, x1=3)
```

[3] ✓ 0.0s

```
Llamada i=1     x=3.00000       y=2.00
Llamada i=2     x=3.00000       y=2.00
Llamada i=3     x=3.00000       y=2.00
Llamada i=4     x=2.00000       y=-3.00
Llamada i=5     x=2.60000       y=-1.10
Llamada i=6     x=2.60000       y=-1.10
Llamada i=7     x=2.60000       y=-1.10
Llamada i=8     x=3.00000       y=2.00
Llamada i=9     x=2.74227       y=-0.20
Llamada i=10    x=2.74227       y=-0.20
Llamada i=11    x=2.74227       y=-0.20
Llamada i=12    x=2.60000       y=-1.10
Llamada i=13    x=2.77296       y=0.03
Llamada i=14    x=2.77296       y=0.03
Llamada i=15    x=2.77296       y=0.03
Llamada i=16    x=2.74227       y=-0.20
Llamada i=17    x=2.76922       y=-0.00
Llamada i=18    x=2.76922       y=-0.00
Llamada i=19    x=2.76922       y=-0.00
Llamada i=20    x=2.77296       y=0.03
```

```
    Llamada i=21     x=2.76929       y=-0.00
    Llamada i=22     x=2.76929       y=-0.00
    Llamada i=23     x=2.76929       y=-0.00
    Llamada i=24     x=2.76922       y=-0.00
    Llamada i=25     x=2.76929       y=0.00

... (2.7692923542484045, 6)
```

Modifique el código provisto para optimizar el número de llamadas a la función.

https://github.com/ztjona/MN-examen-01-2024-B/blob/main/secante_optimizar.ipynb

Cree un repositorio público en Github, ingrese el link del **notebook** donde resuelve el ejercicio.

✔

Luego de optimizar el código y utilizando \(x_0=2, x_1=3 \), conteste:
¿Cuál es el número mínimo de llamadas a la función para llegar a la raíz en el Ejemplo 1?

$i =$ ✔

¿Cuál es el número mínimo de llamadas a la función para llegar a la raíz en el Ejemplo 2?

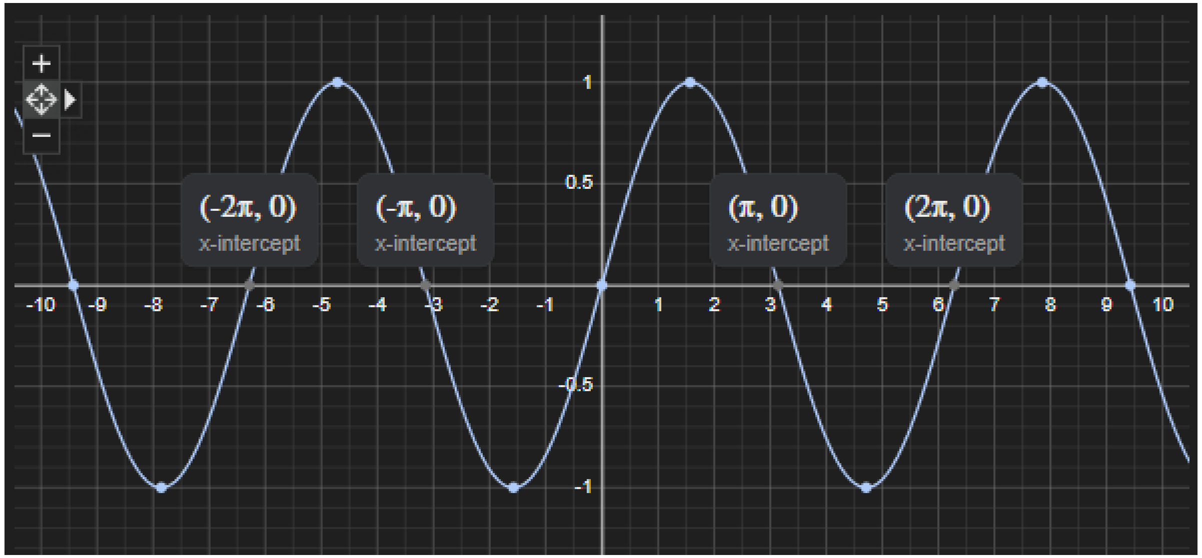
$i =$ ✖

Pregunta **5**

Parcialmente correcta

Puntúa 2,00 sobre 3,00

La función $sin(x)$ tiene infinitas soluciones $\{\ldots, -2\pi, -\pi, 0, \pi, 2\pi, \ldots\}$.



¿A cuál solución converge el método de la Bisección en los siguientes intervalos?

$a = -4, b = 5$	pi ✔
$a = -1, b = 2$	0 ✔
$a = 3, b = 5$	pi ✔

$a=-2.5, b=-1$	Ninguna de las anteriores	✖
$a=-5, b=4$	-pi	✔
$a=-3.5, b=3$	Ninguna de las anteriores	✖

Pregunta **6**

Correcta

Puntúa 5,00 sobre 5,00

Dados los puntos $(-1,1),(1,3)$. Determine el spline cúbico teniendo en cuenta que $f'(x_0)=1, f'(x_n)=2$.

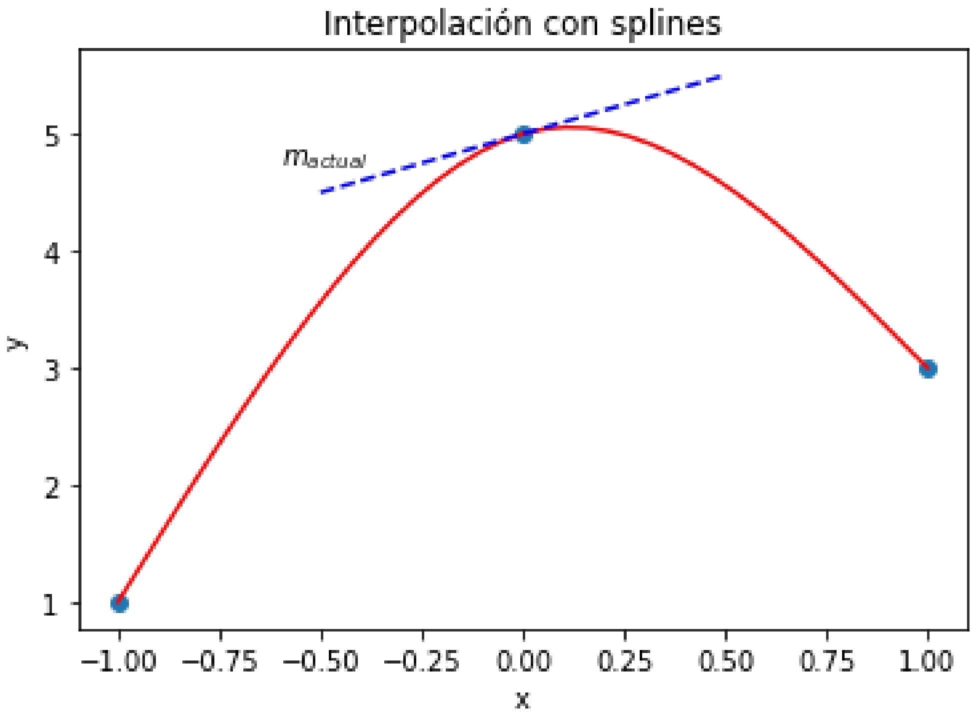
$S_0(x) =$ ✔ $\times(x$ - ✔ $)^3$+ ✔ $\times(x$ - ✔ $)^2$+ ✔ $\times(x$ - ✔ $)$ + ✔

Pregunta **7**

Parcialmente correcta

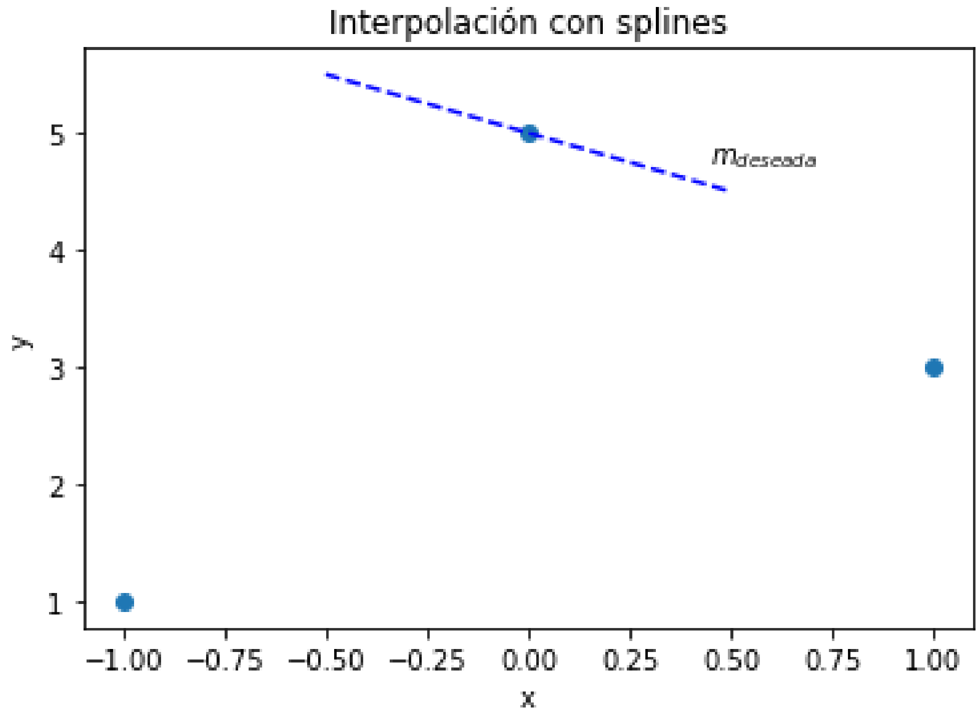
Puntúa 4,00 sobre 18,00

Dados los puntos $(-1, 1), (0, 5), (1, 3)$, se ha obtenido los splines cúbicos correspondientes.
Sin embargo, al observar la figura, usted no se siente satisfecho con la pendiente resultante en el punto (x_1, y_1). Y decide intentar una modificación a las ecuaciones, tal que los splines sean tangentes a una pendiente deseada m en el punto (x_1, y_1).



Recuerde que la expresión de un spline cúbico es la siguiente:
$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$

¿En caso de ser posible, y bajo qué condiciones se puede encontrar los splines cúbicos que cumplan con la condición de m?



Determine la ecuación que se debe modificar para poder cumplir con el requisito de m.

- $S_0(x_1) = y_1$
- $S_1''(x_2) = 0$
- $S_1(x_2) = y_2$
- $S_0''(x_0) = 0$

- $S_1(x_1) = y_1$
- Ninguna de las anteriores
- Sin solución única. Depende de los valores de m
- $S_0'(x_1) = S_1'(x_1)$
- $S_0(x_0) = y_0$
- $S_0''(x_1) = S_1''(x_1)$ ✔

Escriba la expresión del spline S_0. En caso de no existir solución, llene los casilleros con Nan.

$S_0(x) =$ ✖ $\times(x$ - ✖ $)^3$+ ✖ $\times(x$ - ✖ $)^2$+ ✖ $\times(x$ - ✖ $)$ + ✖

Escriba la expresión del spline S_1. En caso de no existir solución, llene los casilleros con Nan.

$S_1(x) =$ ✖ $\times(x$ - ✖ $)^3$+ ✖ $\times(x$ - ✖ $)^2$+ ✖ $\times(x$ - ✖ $)$ + ✖

Cree un repositorio público en Github y grafique su solución. Pegue aquí el enlace de su **notebook**.

✔

Para graficar su respuesta debe utilizar el código base del siguiente repositorio:
https://github.com/ztjona/MN-examen-01-2024-B/blob/main/splines_pendiente.ipynb

Pregunta **8**

Correcta

Puntúa 4,00 sobre 4,00

La interpolación de un conjunto de puntos usando polinomios de Lagrange $P(x)$ está dada por la fórmula:

$$P(x) = \sum_{k=0}^{n} f(x_k)L_k(x)$$

Donde:

$$L_k(x) = \prod_{i=0, i\neq k}^{n} \frac{x-x_i}{x_k-x_i}$$

Dados los puntos $(0,0), (1,1), (2,2), (3,3)$,

El polinomio resultante **simplificado** tiene orden =

Encuentre el polinomio de Lagrange respectivo, (simplifique la expresión para facilitar la evaluación)

$P(x) =$

Usando el polinomio $P(x)$ que obtuvo de respuesta, calcule:

$P(x = 3.78) =$

$P(x = 19.102) =$