

The background of the slide is a dark blue color with a complex, light blue circuit board pattern. The pattern consists of numerous lines, dots, and small circles, resembling a printed circuit board (PCB) layout. The lines are of varying thicknesses and connect various points across the slide. Some points are highlighted with small white circles.

Proyecto A: Ahorro semanal


MÉTODOS NUMÉRICOS

START

The background is a dark blue field with a light blue circuit board pattern. The pattern consists of various lines, dots, and small circles, resembling a printed circuit board (PCB) layout. The lines are of different thicknesses and connect various points, some of which are marked with small circles. The overall effect is a technical, digital aesthetic.

Primer paso

OBTENCIÓN DE LA FÓRMULA



Entendiendo el comportamiento del interés compuesto podemos completar la fórmula a través de un sumatorio.

$$v_f = v_o * (1 + i)^n + \sum_{k=1}^{n-1} A(1 + i)^k$$

Tabla de ejemplo



Semana	Aporte (\$)	Capital (\$)	Ganancia (\$)	Total (\$)
1	100	100	0.15	100.15
2	5	105.15	0.16	105.31
3	5	110.31	0.17	110.48
4	5	115.48	0.18	115.66
5	5	120.66	0.19	120.85
...				
51	5	367.65	0.57	368.22
52	5	373.22	0.57	373.79

USO DE SERIE GEOMÉTRICA

Se puede simplificar la expresión del sumatorio usando la fórmula para la suma de la progresión geométrica.

$$v_f = v_0(1+i)^n + A * \left(\frac{(1+i)^n - (1+i)}{i} \right)$$

The background is a dark blue gradient with a light blue circuit board pattern. The pattern consists of various lines, dots, and geometric shapes that resemble electronic components and wiring, distributed across the entire frame.

Paso 2

USO DEL MÉTODO DE LA SECANTE



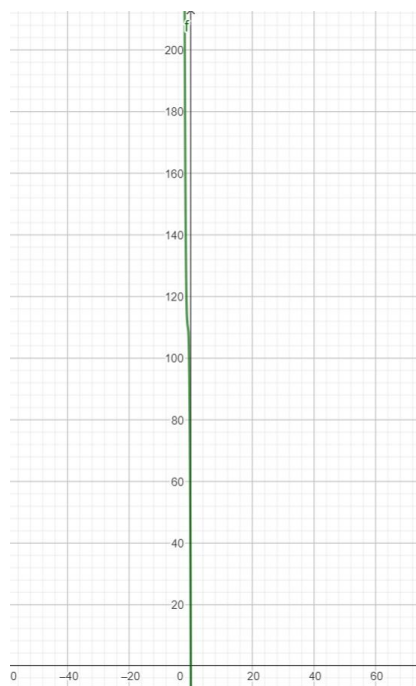
CÁLCULO DE INTERÉS COMPUESTO

Se puede utilizar método de la secante, en la gráfica interés compuesto vs valor final , para obtener el valor de interés compuesto

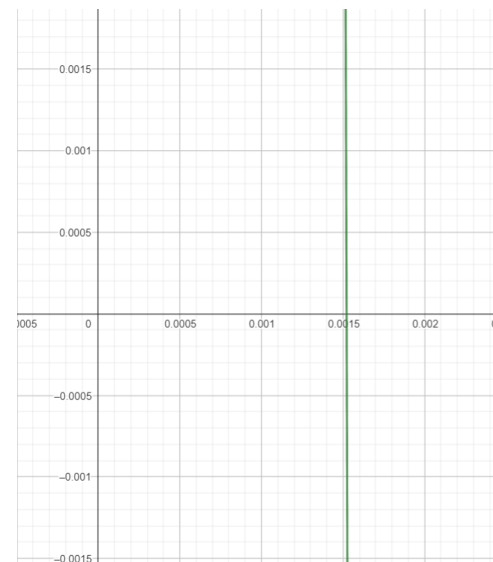
Ejemplo



Ejemplo en funcion de los datos del profesor



- **Capital Inicial:** 100
- **Aporte Periódico:** 5
- **Número de Períodos:** 3
- **Capital Final:** 110.48



The background is a dark blue gradient with a light blue circuit board pattern. The pattern consists of various lines, dots, and geometric shapes that resemble electronic components and wiring, distributed across the entire frame.

Parte 3

PROGRAMACIÓN

CÁLCULO DEL INTERÉS COMPUESTO

```
def f(v0, vf, n, a):  
    def funcion(x):  
        if abs(x) < 1e-10:  
            return vf - v0 * (1 + x) ** n  
        else:  
            return vf - v0 * (1 + x) ** n - a * (((1 + x) ** n -  
(1 + x)) / x)  
    return funcion  
def obtener_interes(capital_inicial, capital_final,  
numero_periodos, aporte):  
    return optimize.newton(f(capital_inicial,  
capital_final, numero_periodos, aporte), 1e-5)
```

pseudocódigo

FUNCIÓN $f(v_0, v_f, n, a)$:

SI x es aproximadamente 0:

Devuelve $v_f - v_0 * (1 + x)^n$

SINO:

Devuelve $v_f - v_0 * (1 + x)^n - a * ((1 + x)^n - (1 + x)) / x$

FIN FUNCIÓN

FUNCIÓN $\text{obtener_interes}(\text{capital_inicial}, \text{capital_final},$
 $\text{numero_periodos}, \text{aporte})$:

Definir función objetivo usando $f(\text{capital_inicial}, \text{capital_final},$
 $\text{numero_periodos}, \text{aporte})$

Usar el método de Newton para resolver la función objetivo y
encontrar la tasa de interés

Retornar la tasa de interés encontrada

FIN FUNCIÓN



GENERACIÓN DE LA GRÁFICA

```
def graficar(capitalInicial, aporte, interes, frequency):  
    funcion_reemplazada = f(capitalInicial, interes, aporte)  
    x = np.linspace(1, periodo(frequency), 60)  
    y = funcion_reemplazada(x)  
    plt.figure(figsize=(10, 6))  
    plt.plot(x, y, color='blue', linewidth=2)  
    plt.title('Gráfica interés vs Ganancia', fontsize=16)  
    plt.xlabel('Interés', fontsize=14)  
    plt.ylabel('Ganancia', fontsize=14)  
    plt.grid(True, linestyle='--', alpha=0.7)  
    image_path = os.path.join(os.getcwd(), 'static', 'grafico_funcion.png')  
    plt.savefig(image_path, format='png', dpi=300, bbox_inches='tight')  
    plt.close()  
    if os.path.exists(image_path):  
        print(f"Archivo generado en: {image_path}, Tamaño: {os.path.getsize(image_path)} bytes")  
        image_url = url_for('static', filename='grafico_funcion.png', _external=True)  
        print('La url es: {}'.format(image_url))  
    else:  
        print("No se pudo generar el archivo.")
```

Pseudocódigo



FUNCIÓN graficar(capitalInicial, aporte, interes, frecuencia):

Crear la función del interés compuesto usando $f(\text{capitalInicial}, \text{interes}, \text{aporte})$

Calcular el rango de x (tiempos) usando la función $\text{periodo}(\text{frecuencia})$

Calcular los valores de y aplicando la función para cada valor de x

Crear una figura para la gráfica

Dibujar la gráfica de x vs y usando un color y tamaño de línea adecuados

Añadir título y etiquetas a los ejes

Guardar la imagen de la gráfica en la carpeta 'static' como 'grafico_funcion.png'

Cerrar la gráfica

SI el archivo fue guardado correctamente:

Imprimir la URL de la imagen generada

SINO:

Imprimir un error

FIN FUNCIÓN

OBTENER URL DEL GRÁFICO

```
async function fetchChartImage() {  
  const url = 'https://backend-calculadora.onrender.com/api/chart';  
  try {  
    const response = await fetch(url, { method: 'POST', headers: { 'Content-Type':  
'application/json' } });  
    const data = await response.json();  
    const chartImage = document.getElementById('resultChartImage');  
    chartImage.src = data.chartUrl;  
  } catch (error) {  
    console.error('Error al obtener el gráfico:', error);  
    alert('Ocurrió un error al obtener la imagen del gráfico.');  } finally {  
    loadingScreen.style.display = 'none';  
  }  
}
```

Pseudocódigo



Función fetchChartImage()

- Definir URL del gráfico

- Intentar obtener la URL del gráfico desde el servidor backend usando una solicitud POST

- Si la respuesta es exitosa:

 - Actualizar la imagen del gráfico en el frontend con la URL recibida

- Si hay un error:

 - Mostrar un mensaje de error

- Finalmente:

 - Ocultar pantalla de carga

Fin

GENERAR TABLA DE DATOS

```
def table_data(initial_capital, num_periods, periodic_contribution, interest, start_period=1):
    table_data = []
    function = f(initial_capital, interest, periodic_contribution)
    for period in range(start_period, start_period + num_periods):
        total = function(period)
        if period == start_period:
            gain = total - initial_capital
            capital = initial_capital
            contribution = 0
        else:
            capital = function(period - 1)
            gain = total - capital - periodic_contribution
            contribution = periodic_contribution
        table_data.append({
            'period': period,
            'contribution': contribution,
            'capital': round(capital, 2),
            'gain': round(gain, 2),
            'total': round(total, 2)
        })
    return table_data
```

Pseudocódigo

FUNCIÓN table_data(capital_inicial, num_periodos, aporte_periodico, interes, periodo_inicial=1):

Crear una lista vacía llamada 'table_data'

Crear la función del interés compuesto usando $f(\text{capital_inicial}, \text{interes}, \text{aporte_periodico})$

PARA cada periodo desde 'periodo_inicial' hasta 'periodo_inicial + num_periodos':

Calcular el total acumulado para ese periodo usando la función del interés compuesto

SI el periodo es el primero:

Calcular ganancia como la diferencia entre el total y el capital inicial

Establecer el capital como el capital inicial

No hay aporte en el primer periodo

SINO:

Calcular el capital en el período anterior usando la función

Calcular la ganancia como la diferencia entre el total y el capital (ajustado por el aporte)

Establecer el aporte como el aporte periódico

Crear un diccionario con los valores del periodo, aporte, capital, ganancia y total

Añadir el diccionario a la lista 'table_data'

Retornar 'table_data' (la lista con los datos de la tabla)

FIN FUNCIÓN



CREAR TABLA EN FRONT END

```
async function fetchTableData(requiredRows = 5) {
  const url = 'https://backend-calculadora.onrender.com/api/table';
  try {
    const response = await fetch(url, {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ requiredRows })
    });
    const data = await response.json();
    if (data.dataTable) {
      createTable(data.dataTable);
    } else {
      alert('No hay datos para mostrar');
    }
  } catch (error) {
    console.error('Error al obtener los datos:', error);
    alert('Ocurrió un error al obtener los datos de la tabla.');
```

Pseudocódigo

```
  } finally {
    loadingScreen.style.display = 'none';
  }
}
```



Función fetchTableData(requiredRows)

- Definir URL para la tabla de datos

- Intentar obtener los datos de la tabla desde el servidor backend usando una solicitud POST

- Si la respuesta es exitosa:

- Si hay datos:

- Llamar a la función para crear la tabla con los datos obtenidos

- Si no hay datos:

- Mostrar advertencia

- Si hay un error:

- Mostrar mensaje de error

- Finalmente:

- Ocultar pantalla de carga

Fin

GRACIAS POR SU ATENCIÓN!

