**UNIVERSIDAD DE GUADALAJARA**

**CUCEI**

# MATERIA: COMPUTACIÓN TOLERANTE A FALLAS

**Clave:** I7036 **Sección:** D09

**NRC:** 201689 2024 B

**PROGRAMA 1:** Señales continuas en Tiempo

**Integrantes:** Alejandro Aecio Galindo Rivera - 222308963

Miguel Angel Velazquez Gonzalez - 218450135

Ricardo Aaron Hernandez Macias - 218588048

Diego Emilio Enciso Mora - 218452847

## 1. Importación de librerías

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

- **`numpy`** se importa como `np` y se usa para realizar operaciones matemáticas avanzadas, como la creación de arreglos, generación de funciones trigonométricas, y cálculo de valores estadísticos.
- **`matplotlib.pyplot`** se importa como `plt` y se usa para crear gráficos de las señales y sus propiedades.

## 2. Definición de parámetros

```python
# Definición de parámetros
A = 1  # Amplitud
omega0 = 2 * np.pi  # Frecuencia angular base
t_f = np.linspace(0, 2 * 2 * np.pi / omega0, 1000)  # Tiempo para dos periodos de f(t)
t_g = np.linspace(0, 4 * 2 * np.pi / (2 * omega0), 1000)  # Tiempo para cuatro periodos de g(t)
t_h = np.linspace(0, 6 * 2 * np.pi / (3 * omega0), 1000)  # Tiempo para seis periodos de h(t)
```

- **A**: Se define la amplitud de la señal como 1. La amplitud es el valor máximo de la señal sinusoidal.
- **omega0**: Se calcula la frecuencia angular base como $2\pi$, lo que representa una oscilación completa (un ciclo) en radianes por segundo.
- **t_f**, **t_g**, **t_h**: Utilizando np.linspace, se crean arrays de tiempo para los diferentes periodos de las señales:
  - t_f: Dos periodos de la señal f(t).
  - t_g: Cuatro periodos de la señal g(t).
  - t_h: Seis periodos de la señal h(t).

## 3. Definición de las funciones sinusoidales

```python
# Definición de las funciones
f_t = A * np.sin(omega0 * t_f)
g_t = A * np.sin(2 * omega0 * t_g)
h_t = A * np.sin(3 * omega0 * t_h)
```

- **`f_t`**: Se calcula como la señal sinusoidal $A \sin(\omega_0 t)$ usando la frecuencia angular base.
- **`g_t`**: Se define como $A \sin(2\omega_0 t)$ una señal con el doble de la frecuencia angular, es decir, oscila más rápido que f(t).

- **h_t**: Es A sin(3ω0t), una señal con el triple de la frecuencia angular, que oscila aún más rápido.

### 4. Gráfica de las señales sinusoidales

```
# Gráfica de dos periodos de f(t), cuatro de g(t) y seis de h(t)
plt.figure(figsize=(10, 6))
plt.plot(t_f, f_t, label='f(t) = A sin(w0t)', color='blue')
plt.plot(t_g, g_t, label='g(t) = A sin(2w0t)', color='green')
plt.plot(t_h, h_t, label='h(t) = A sin(3w0t)', color='red')

# Configuración de la gráfica
plt.title('Señales f(t), g(t), h(t) en varios periodos')
plt.xlabel('Tiempo (t)')
plt.ylabel('Amplitud')
plt.grid(True)
plt.legend()
plt.show()
```

- Se configura la figura con un tamaño de 10x6 pulgadas.
- Se grafica cada señal en diferentes colores:
  - f(t) en azul.
  - g(t) en verde.
  - h(t) en rojo.
- La gráfica incluye leyendas para identificar cada señal, etiquetas para los ejes de tiempo y amplitud, una cuadrícula para facilitar la lectura, y se muestra al usuario.

### 5. Cálculo del valor eficaz (RMS) de f(t)

```
# Cálculo del valor eficaz (RMS) de f(t)
rms_f = np.sqrt(np.mean(f_t**2))  # Valor eficaz de f(t)
```

- **f_t**2**: El valor cuadrado de cada punto de la señal f(t).
- **np.mean(f_t**2)**: El promedio del valor cuadrado a lo largo de todo el tiempo de la señal.
- **np.sqrt(np.mean(f_t**2))**: El valor eficaz o RMS (Root Mean Square) se calcula tomando la raíz cuadrada del promedio del valor cuadrático de la señal. Este valor representa una medida de la magnitud promedio de la señal.

### 6. Gráfica de un periodo de f(t) y su valor RMS

```
# Gráfica de un periodo de f(t) junto con su valor eficaz
t_f_1 = np.linspace(0, 2 * np.pi / omega0, 1000)  # Un periodo de f(t)
f_t_1 = A * np.sin(omega0 * t_f_1)
```

- **t_f_1**: Un array de tiempo que cubre solo un periodo de la señal `f(t)`.
- **f_t_1**: La señal sinusoidal `f(t)` en ese único periodo.

```python
plt.figure(figsize=(10, 6))
plt.plot(t_f_1, f_t_1, label='f(t) = A sin(w0t)', color='blue')
plt.axhline(y=rms_f, color='orange', linestyle='--', label=f'Valor eficaz (RMS) = {rms_f:.2f}')

# Configuración de la gráfica
plt.title('Señal f(t) y su valor eficaz')
plt.xlabel('Tiempo (t)')
plt.ylabel('Amplitud')
plt.grid(True)
plt.legend()
plt.show()
```

- Se grafica la señal `f(t)` para un solo periodo.
- Se añade una línea horizontal (`axhline`) para representar el valor RMS de la señal.
- El valor RMS se indica con una leyenda para que sea fácilmente identificable.

### 7. Gráfica del valor absoluto de `f(t)` para dos periodos

```python
# Gráfica del valor absoluto de f(t) para dos periodos
t_f_2 = np.linspace(0, 2 * 2 * np.pi / omega0, 1000)  # Dos periodos de f(t)
f_abs_t = np.abs(A * np.sin(omega0 * t_f_2))
```

- **`np.abs`**: Calcula el valor absoluto de la señal `f(t)` en cada punto del tiempo. El valor absoluto elimina los signos negativos y muestra solo la magnitud de la señal.

```python
plt.figure(figsize=(10, 6))
plt.plot(t_f_2, f_abs_t, label='|f(t)|', color='purple')

# Configuración de la gráfica
plt.title('Valor absoluto de f(t)')
plt.xlabel('Tiempo (t)')
plt.ylabel('Amplitud')
plt.grid(True)
plt.legend()
plt.show()
```

- Se grafica el valor absoluto de la señal `f(t)` durante dos periodos.

## 8. Gráfica del valor cuadrático de `f(t)` para dos periodos

```
# Gráfica del valor cuadrático de f(t) para dos periodos
f_squared_t = (A * np.sin(omega0 * t_f_2))**2
```

- **`f_squared_t`**: Se calcula el valor cuadrado de la señal `f(t)`. Esto es útil en el análisis de energía y potencia de señales, ya que la potencia está relacionada con el cuadrado de la amplitud.

```
plt.figure(figsize=(10, 6))
plt.plot(t_f_2, f_squared_t, label='$f^2(t)$', color='brown')

# Configuración de la gráfica
plt.title('Valor cuadrático de f(t)')
plt.xlabel('Tiempo (t)')
plt.ylabel('Amplitud')
plt.grid(True)
plt.legend()
plt.show()
```

- Se grafica el valor cuadrático de la señal para dos periodos.

## 9. Gráfica de `f(t)` en decibeles

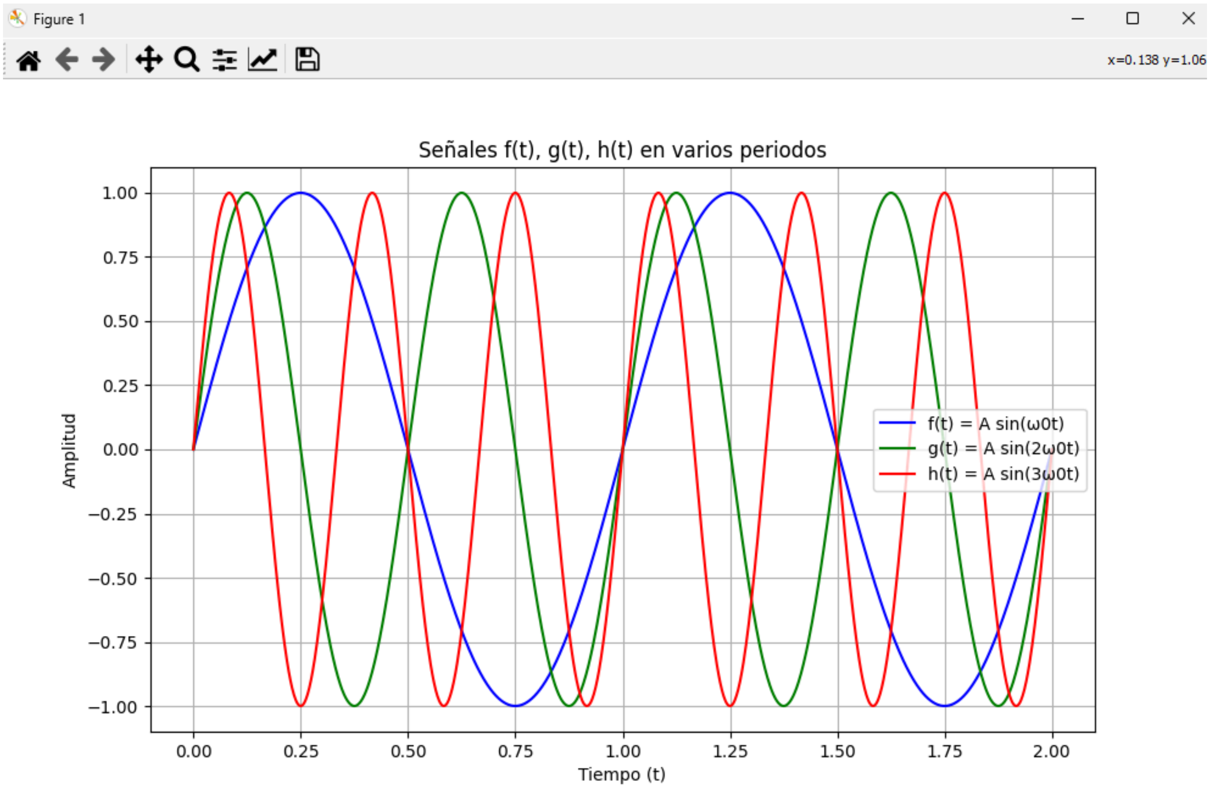
```
# Gráfica de f(t) en decibeles
f_db_t = 20 * np.log10(np.abs(f_t_1))
```

- **`np.log10(np.abs(f_t_1))`**: Calcula el logaritmo base 10 del valor absoluto de la señal `f(t)` en un periodo.
- **`20 *`**: La conversión a decibelios (dB) se hace multiplicando el logaritmo por 20. Los decibelios son una medida relativa usada en acústica y telecomunicaciones para representar la relación entre magnitudes.

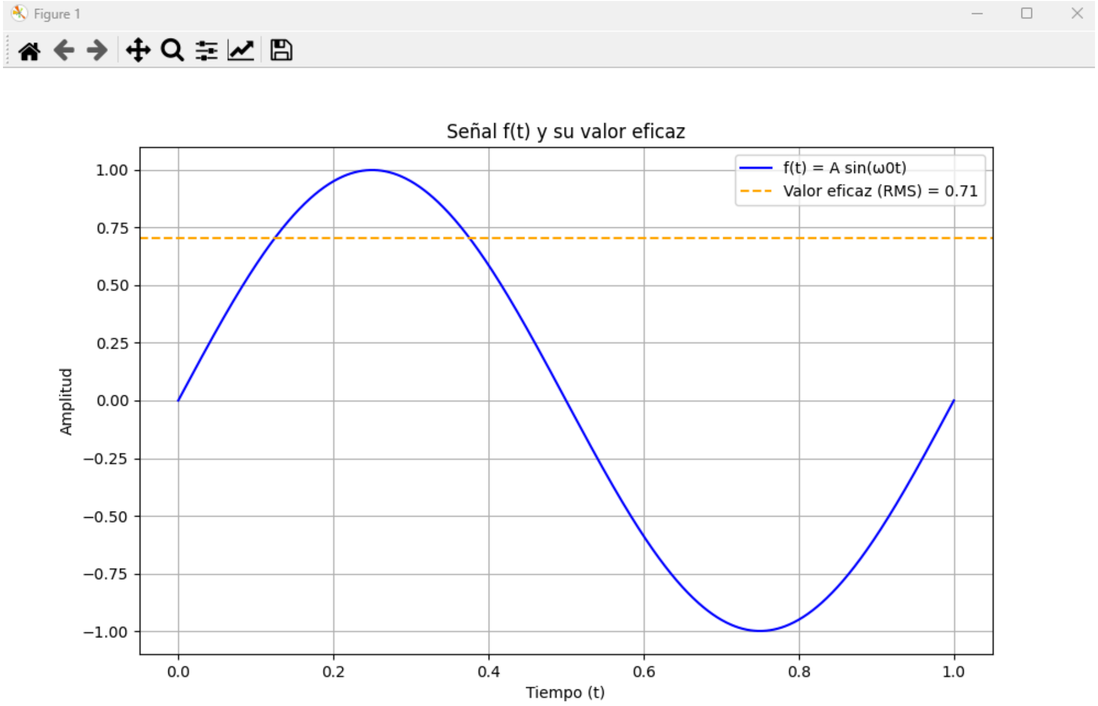
```python
plt.figure(figsize=(10, 6))
plt.plot(t_f_1, f_db_t, label='f(t) en dB', color='magenta')

# Configuración de la gráfica
plt.title('Señal f(t) en decibeles')
plt.xlabel('Tiempo (t)')
plt.ylabel('Amplitud (dB)')
plt.grid(True)
plt.legend()
plt.show()
```

Se grafica la señal `f(t)` en escala de decibelios. Esto convierte los valores de amplitud en una escala logarítmica, útil para analizar señales en el dominio de la frecuencia.

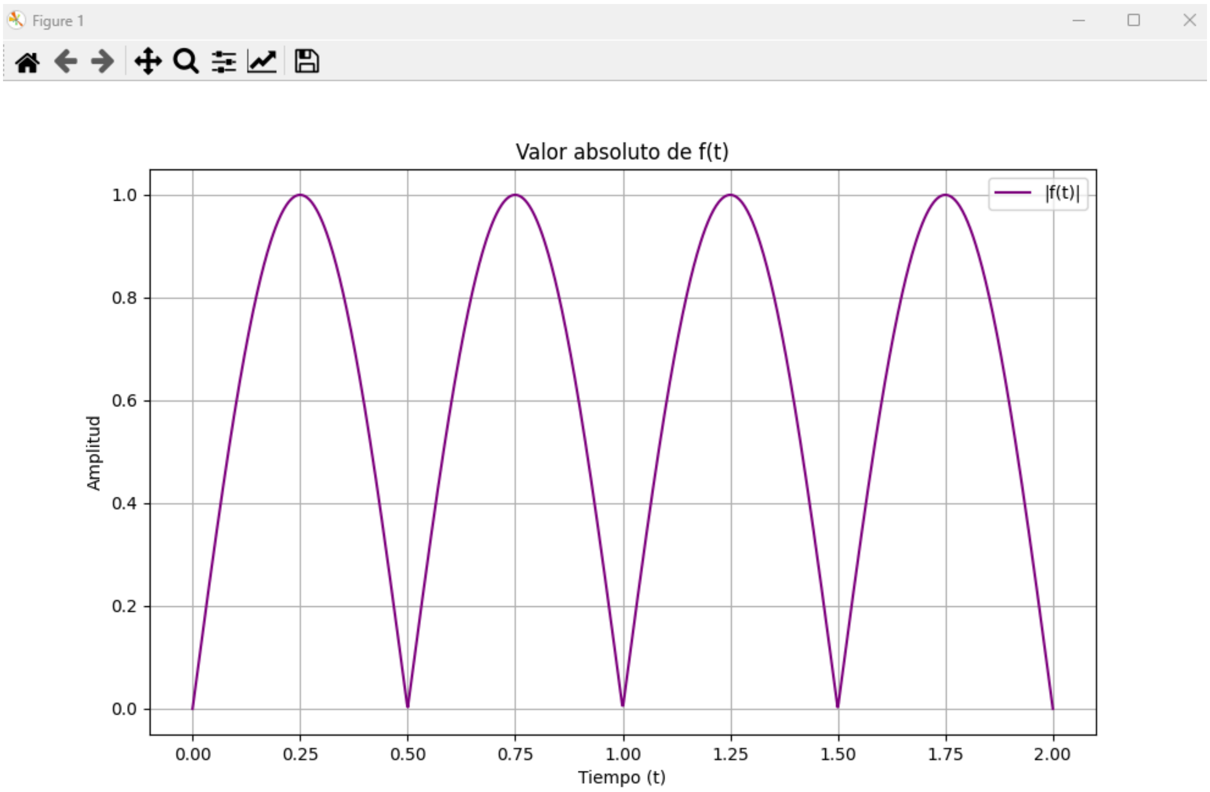
Gráficas para cada punto del documento del programa 01.
1. Dos periodos de la señal f (t), cuatro periodos de la señal g (t) y seis periodos de la señal h (t).




2. Valor eficaz.




3. Dos periodos de la señal |f(t)|.




4. Dos periodos de la señal f^2(t)

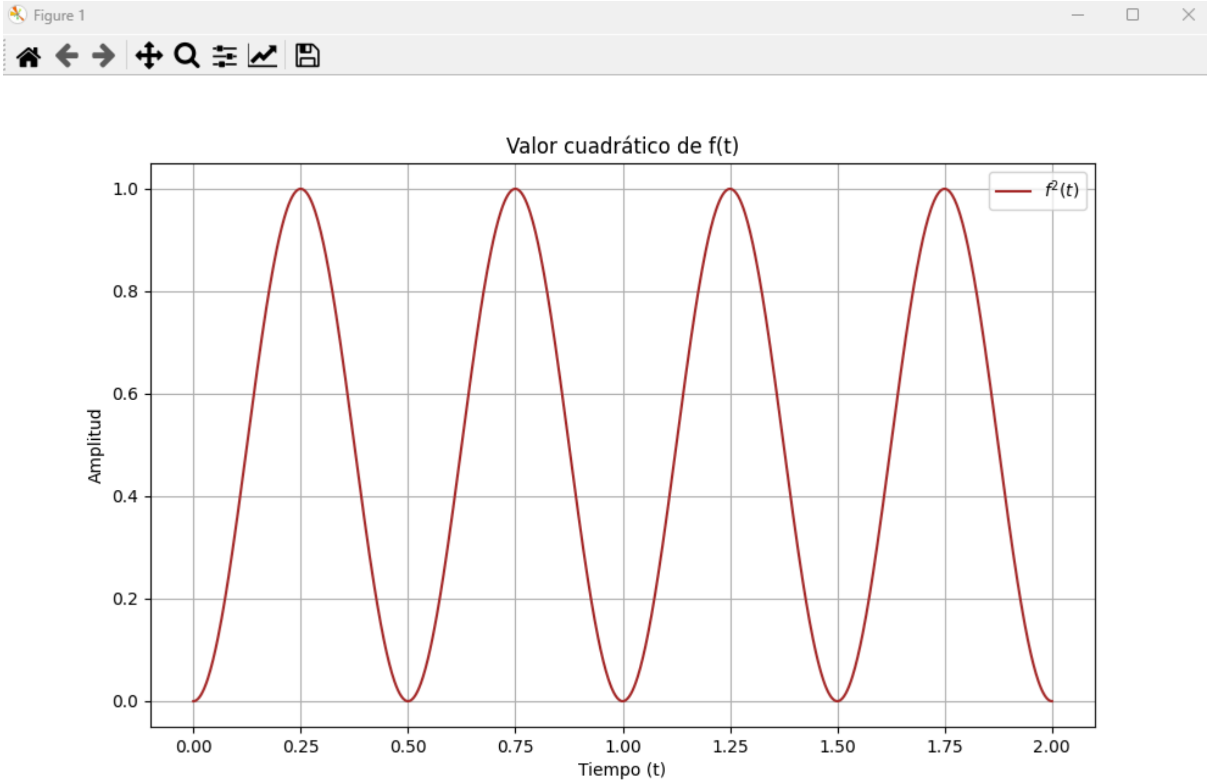
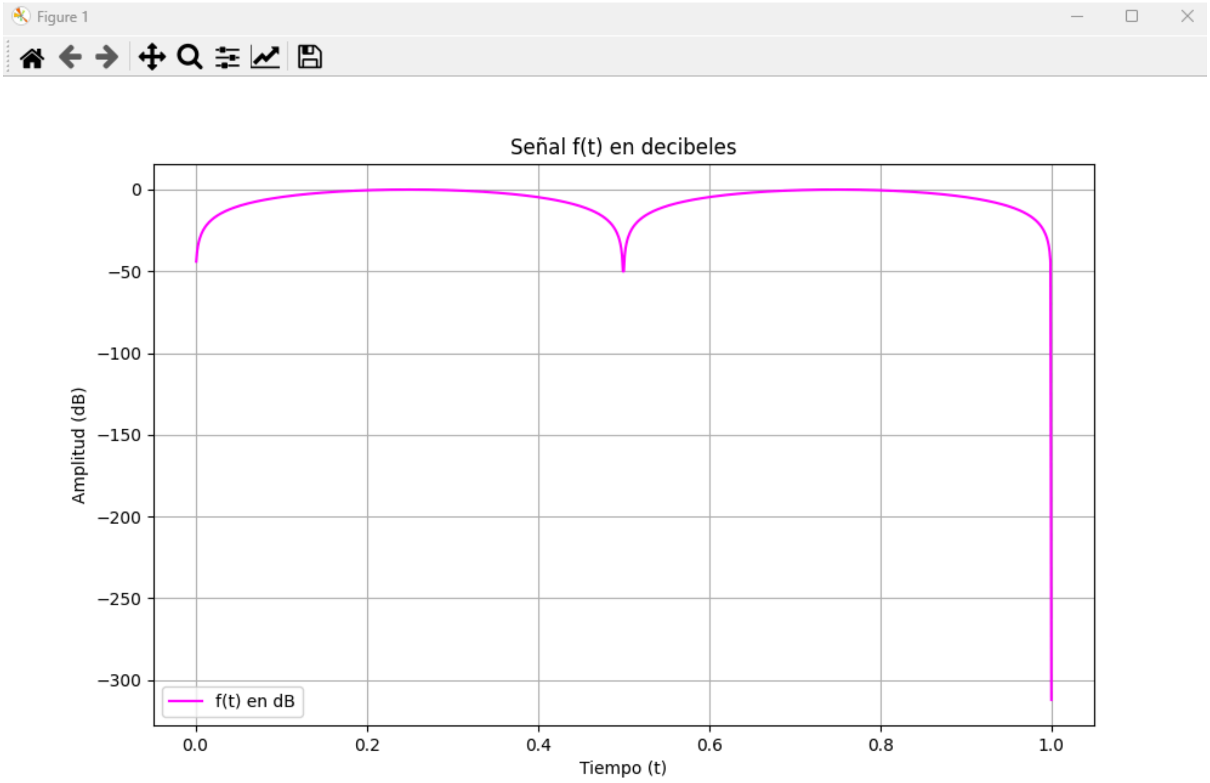


Figure 1
Valor cuadrático de f(t)
$f^2(t)$
Amplitud
Tiempo (t)
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

5. Dos periodos de la señal f(t) en decibeles.




Conclusión

Cada operación en este código nos muestra una forma distinta de visualizar y analizar una señal sinusoidal. Desde la representación directa de la señal en función del tiempo hasta su análisis en términos de valor eficaz, valor absoluto, y en decibelios, estas operaciones permiten una comprensión profunda de la magnitud, energía, y las variaciones de las señales en diferentes contextos.