Tarea 4: Grafica

Daniel Estuardo, Suy Fuentes, 202000443

Proyectos de computación aplicados a I.E

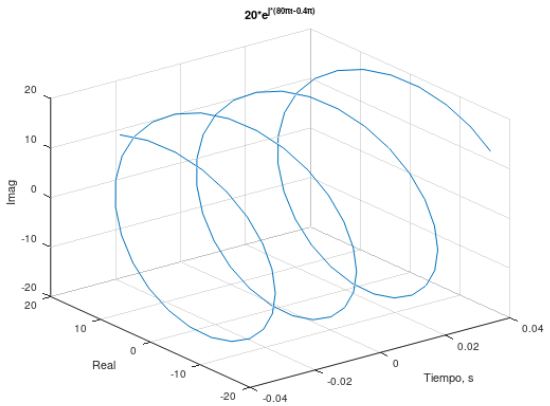
Escuela de Mecánica Eléctrica, Facultad de Ingeniería, Universidad de San Carlos de Guatemala

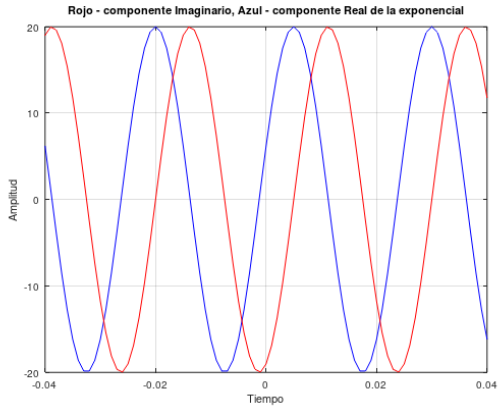
Programa 1

Este código de Octave genera dos figuras separadas. La primera figura es un gráfico 3D que representa la función compleja 20*e^j(80πt−0.4π) en función del tiempo, la parte real y la parte imaginaria.

La segunda figura contiene dos gráficos 2D. En uno, se muestra la parte real de la función compleja en azul, y en el otro se muestra la parte imaginaria en rojo. Ambos gráficos comparten el mismo eje de tiempo.

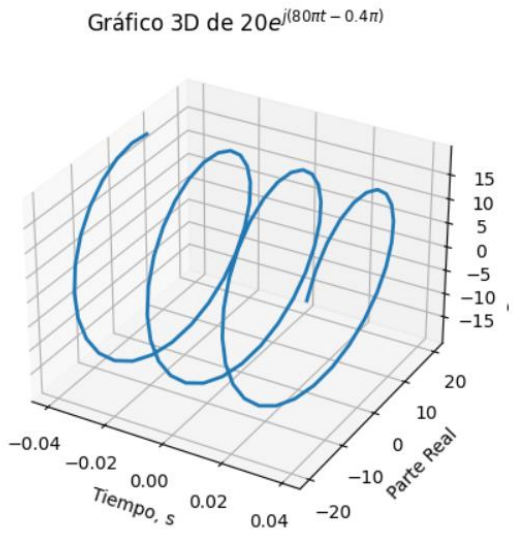
```
t=-0.04:0.001:0.04;
x=20*exp(j*(80*pi*t-0.4*pi));

%Grafica 3D
figure;
plot3(t, real(x), imag(x)); grid on;
title('20*e^{j*(80\pit-0.4\pi)}')
xlabel('Tiempo, s'); ylabel('Real'); zlabel('Imag')

%Grafica 2D
figure;
plot(t, real(x), 'b'); hold on
plot(t, imag(x), 'r'); grid
title('Rojo - componente Imaginario, Azul - componente Real de la exponencial')
xlabel('Tiempo'); ylabel('Amplitud')
```

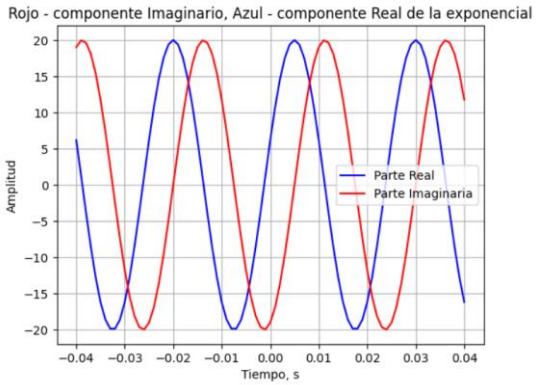




Código en Python utilizando la biblioteca numpy para cálculos y matplotlib para la visualización:

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

t = np.arange(-0.04, 0.041, 0.001)
x = 20 * np.exp(1j * (80 * np.pi * t - 0.4 * np.pi))

# Gráfico 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(t, np.real(x), np.imag(x), linewidth=2)
ax.grid(True)
ax.set_title('Gráfico 3D de $20e^{j(80\pi t - 0.4\pi)}$')
ax.set_xlabel('Tiempo, s')
ax.set_ylabel('Parte Real')
ax.set_zlabel('Parte Imaginaria')

# Gráfico 2D
plt.figure()
plt.plot(t, np.real(x), 'b', label='Parte Real')
plt.plot(t, np.imag(x), 'r', label='Parte Imaginaria')
plt.grid(True)
plt.title('Rojo - componente Imaginario, Azul - componente Real de la exponencial')
plt.xlabel('Tiempo, s')
plt.ylabel('Amplitud')
plt.legend()

plt.show()
```

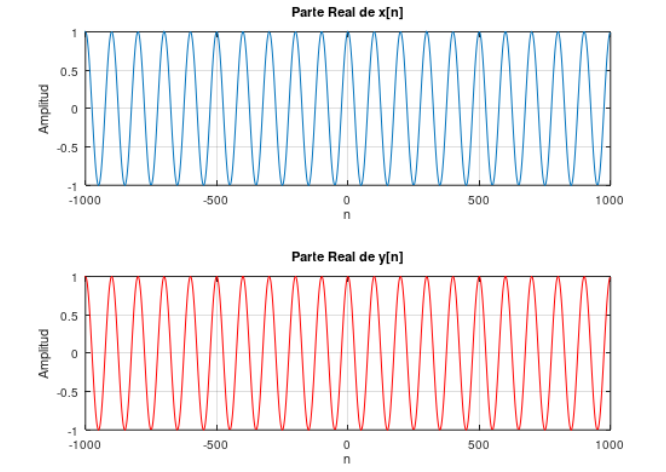




Problema 2

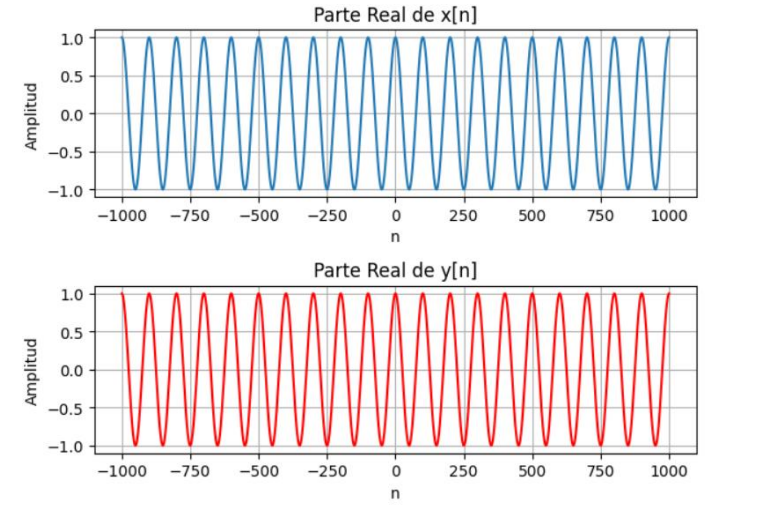
Este código crea dos subgráficas en una figura. La primera subgráfica muestra la parte real de x[n] en azul, y la segunda subgráfica muestra la parte real de y[n] en rojo

```
n = -1000:1000;
x = exp(1j * 2 * pi * 0.01 * n);
y = exp(1j * 2 * pi * 2.01 * n);

% Crear dos subgráficas en una figura
figure;

% Subgráfica 1
subplot(2, 1, 1);
plot(n, real(x));
title('Parte Real de x[n]');
xlabel('n');
ylabel('Amplitud');
grid on;

% Subgráfica 2
subplot(2, 1, 2);
plot(n, real(y), 'r');
title('Parte Real de y[n]');
xlabel('n');
ylabel('Amplitud');
grid on;
```



Código en Python utilizando numpy para cálculos y matplotlib para la visualización:

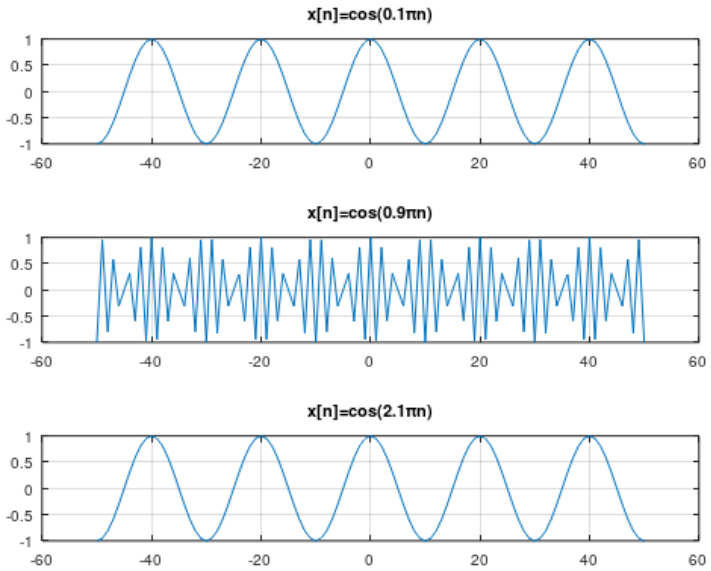
```python
import numpy as np
import matplotlib.pyplot as plt

n = np.arange(-1000, 1001)
x = np.exp(1j * 2 * np.pi * 0.01 * n)
y = np.exp(1j * 2 * np.pi * 2.01 * n)

# Crear dos subgráficas en una figura
plt.figure()

# Subgráfica 1
plt.subplot(2, 1, 1)
plt.plot(n, np.real(x))
plt.title('Parte Real de x[n]')
plt.xlabel('n')
plt.ylabel('Amplitud')
plt.grid(True)

# Subgráfica 2
plt.subplot(2, 1, 2)
plt.plot(n, np.real(y), 'r')
plt.title('Parte Real de y[n]')
plt.xlabel('n')
plt.ylabel('Amplitud')
plt.grid(True)

plt.tight_layout()  # Ajusta el diseño para evitar solapamientos
plt.show()
```



Problema 3

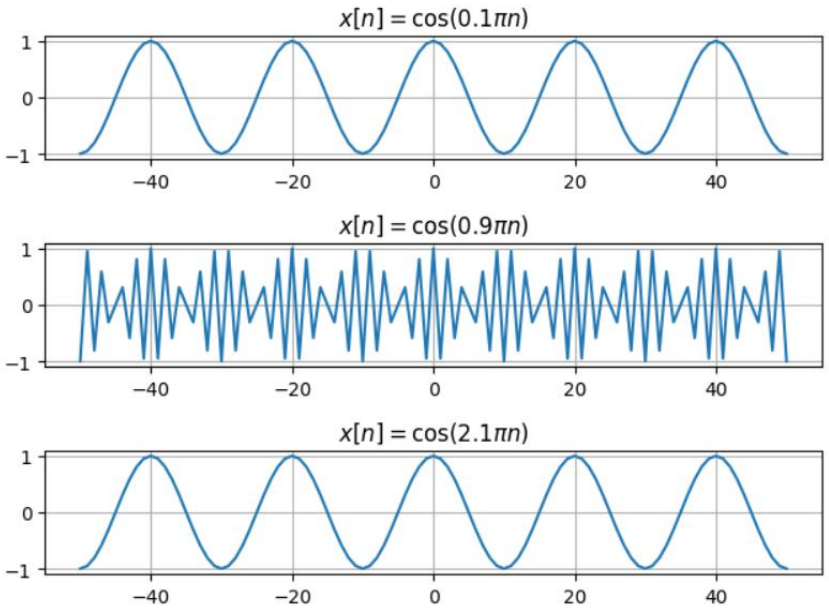
El código crea una figura con tres subgráficas apiladas verticalmente, cada una mostrando una señal diferente en función de la variable discreta n. Las señales x[n], y[n], y z[n] son representaciones discretas de las funciones coseno con diferentes frecuencias angulares.

```
n=-50:50;
x=cos(pi*0.1*n);
y=cos(pi*0.9*n);
z=cos(pi*2.1*n);

subplot(3,1,1);
plot(n,x);
title('x[n]=cos(0.1\pin)');
grid;

subplot(3,1,2);
plot(n,y);
title('x[n]=cos(0.9\pin)');
grid;

subplot(3,1,3);
plot(n,z);
title('x[n]=cos(2.1\pin)');
grid;
```



Código en Python utilizando numpy para cálculos y matplotlib para la visualización:

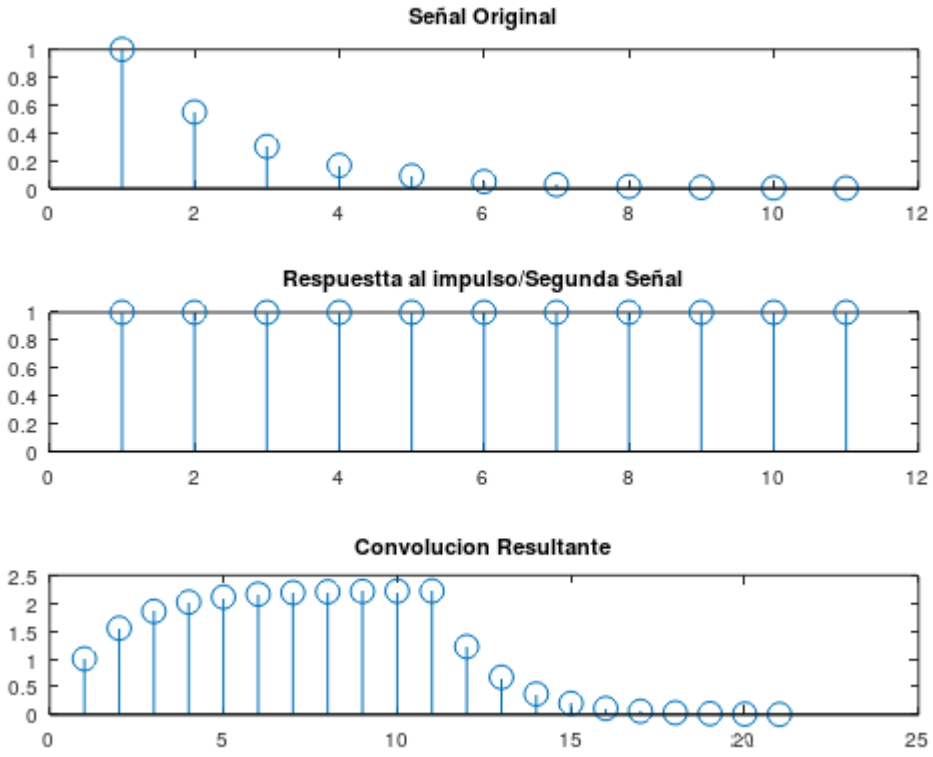
```python
import numpy as np
import matplotlib.pyplot as plt

n = np.arange(-50, 51)
x = np.cos(np.pi * 0.1 * n)
y = np.cos(np.pi * 0.9 * n)
z = np.cos(np.pi * 2.1 * n)

# Crear tres subgráficas en una figura
plt.figure()

# Subgráfica 1
plt.subplot(3, 1, 1)
plt.plot(n, x)
plt.title(r'$x[n]=\cos(0.1\pi n)$')  # Usar r para manejar caracteres especiales
plt.grid(True)

# Subgráfica 2
plt.subplot(3, 1, 2)
plt.plot(n, y)
plt.title(r'$x[n]=\cos(0.9\pi n)$')
plt.grid(True)

# Subgráfica 3
plt.subplot(3, 1, 3)
plt.plot(n, z)
plt.title(r'$x[n]=\cos(2.1\pi n)$')
plt.grid(True)

plt.tight_layout()  # Ajusta el diseño para evitar solapamientos
plt.show()
```



Problema 4

Código muestra la señal original x[n], la respuesta al impulso (que parece ser un impulso discreto), y la convolución resultante y[n] de x[n] y h[n] en tres subgráficas separadas. La función *stem* se utiliza para graficar secuencias discretas.

```
n=-3:7;
x=0.55.^(n+3);
h=[1,1,1,1,1,1,1,1,1,1,1];
y=conv(x,h);

subplot(3,1,1);
stem(x)
title('Señal Original');

subplot(3,1,2);
stem(h) %usa stem para secuencias discretas
title('Respuestta al impulso/Segunda Señal');

subplot(3,1,3);
stem(y)
title('Convolucion Resultante');
```



Código en Python utilizando numpy para cálculos y matplotlib para la visualización:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import convolve

n = np.arange(-3, 8)
x = 0.55**(n + 3)
h = np.ones(11)  # Se define una secuencia de impulso discreto
y = convolve(x, h, mode='full')

# Crear tres subgráficas en una figura
plt.figure(figsize=(10, 8))

# Subgráfica 1
plt.subplot(3, 1, 1)
plt.stem(n, x)
plt.title('Señal Original')
plt.grid(True)

# Subgráfica 2
plt.subplot(3, 1, 2)
plt.stem(n, h)
plt.title('Respuesta al impulso/Segunda Señal')
plt.grid(True)

# Subgráfica 3
plt.subplot(3, 1, 3)
plt.stem(np.arange(len(y)), y)
plt.title('Convolución Resultante')
plt.grid(True)

plt.tight_layout()  # Ajusta el diseño para evitar solapamientos
plt.show()
```