Práctica 3: Muestreo, Cuantización y Codificación

Implementar el código en matlab descrito en la parte inferior del documento. Dado el código responder las siguientes preguntas:

1. **Problema 1:**

- ¿Qué sucede cuando muestreamos por debajo de la frecuencia de Nyquist?
- ¿Cómo afecta el número de bits al error de cuantización?
- Calcula teóricamente la relación señal a ruido (SNR) esperado y compáralo con el medido.

2. **Problema 2:**

- ¿Cómo afecta la reducción de la frecuencia de muestreo a la calidad del audio?
- Explica la relación entre el número de bits y el SNR observado.
- ¿Por qué aparecen componentes frecuenciales adicionales en el espectro después de la cuantización?
- Modifica la frecuencia fundamental en el Problema 1 y observa cómo cambian los resultados.
- Prueba con diferentes formas de onda (cuadrada, triangular) y analiza los efectos.
- Implementa cuantización no uniforme (ley μ o ley A) y compara con la cuantización uniforme.

3. **Problema 3:**

- Explica con tus propias palabras el teorema de muestreo de Nyquist-Shannon.
- Describe los tipos de error en la cuantización: error de granularidad y de sobrecarga.
- ¿Qué es la cuantización PCM?
- ¿Qué ventajas tiene la codificación PCM frente a otras técnicas?

```
%% PROBLEMAS MUESTREO, CUANTIZACIÓN Y CODIFICACIÓN DE UNA SEÑAL SENOIDAL
% Objetivo: Analizar los efectos del muestreo y cuantización en una señal
analógica

clear all; close all; clc;

%% Parámetros de la señal original
f = 5;              % Frecuencia de la señal (Hz)
A = 2;              % Amplitud de la señal (V)
fs1 = 20;           % Frecuencia de muestreo 1 (Hz) - SUBMUESTREO
fs2 = 50;           % Frecuencia de muestreo 2 (Hz) - SOBREMUESTREO
fs3 = 10;           % Frecuencia de muestreo 3 (Hz) - NYQUIST
t_final = 1;        % Tiempo final de simulación (s)

%% 1. Señal analógica original
t_analogico = 0:0.001:t_final;
x_analogico = A * sin(2*pi*f*t_analogico);

figure('Position', [100, 100, 1200, 800]);
subplot(3,4,1);
plot(t_analogico, x_analogico, 'b-', 'LineWidth', 2);
title('Señal Analógica Original');
xlabel('Tiempo (s)');
ylabel('Amplitud (V)');
grid on;

%% 2. Muestreo con diferentes frecuencias
% Submuestreo (fs1 = 20 Hz)
t_muestreo1 = 0:1/fs1:t_final;
x_muestreo1 = A * sin(2*pi*f*t_muestreo1);

% Sobremuestreo (fs2 = 50 Hz)
t_muestreo2 = 0:1/fs2:t_final;
x_muestreo2 = A * sin(2*pi*f*t_muestreo2);

% Muestreo en Nyquist (fs3 = 10 Hz)
t_muestreo3 = 0:1/fs3:t_final;
x_muestreo3 = A * sin(2*pi*f*t_muestreo3);

% Graficar señales muestreadas
subplot(3,4,2);
stem(t_muestreo1, x_muestreo1, 'r-', 'LineWidth', 1.5, 'Marker', 'o');
hold on;
plot(t_analogico, x_analogico, 'b--', 'LineWidth', 0.5);
title(['Muestreo: fs = ' num2str(fs1) ' Hz (Submuestreo)']);
xlabel('Tiempo (s)');
ylabel('Amplitud (V)');
legend('Muestras', 'Señal original');
grid on;

subplot(3,4,3);
stem(t_muestreo2, x_muestreo2, 'r-', 'LineWidth', 1.5, 'Marker', 'o');
hold on;
plot(t_analogico, x_analogico, 'b--', 'LineWidth', 0.5);
title(['Muestreo: fs = ' num2str(fs2) ' Hz (Sobremuestreo)']);
```

```
xlabel('Tiempo (s)');
ylabel('Amplitud (V)');
legend('Muestras', 'Señal original');
grid on;

subplot(3,4,4);
stem(t_muestreo3, x_muestreo3, 'r-', 'LineWidth', 1.5, 'Marker', 'o');
hold on;
plot(t_analogico, x_analogico, 'b--', 'LineWidth', 0.5);
title(['Muestreo: fs = ' num2str(fs3) ' Hz (Nyquist)']);
xlabel('Tiempo (s)');
ylabel('Amplitud (V)');
legend('Muestras', 'Señal original');
grid on;

%% 3. Cuantización
n_bits = 3;         % Número de bits para cuantización
niveles = 2^n_bits; % Número de niveles de cuantización

% Rango de cuantización
rango_max = A;
rango_min = -A;
paso_cuantizacion = (rango_max - rango_min) / niveles;

% Cuantizar la señal sobremuestreada
x_cuantizado = zeros(size(x_muestreo2));
indices_cuantizacion = zeros(size(x_muestreo2));

for i = 1:length(x_muestreo2)
    % Encontrar el nivel de cuantización más cercano
    nivel = round((x_muestreo2(i) - rango_min) / paso_cuantizacion);
    nivel = max(0, min(nivel, niveles-1)); % Limitar entre 0 y niveles-1
    x_cuantizado(i) = rango_min + nivel * paso_cuantizacion;
    indices_cuantizacion(i) = nivel;
end

% Graficar señal cuantizada
subplot(3,4,5);
stairs(t_muestreo2, x_cuantizado, 'g-', 'LineWidth', 2);
hold on;
stem(t_muestreo2, x_muestreo2, 'ro', 'MarkerSize', 4);
title(['Cuantización (' num2str(n_bits) ' bits, ' num2str(niveles) '
niveles)']);
xlabel('Tiempo (s)');
ylabel('Amplitud (V)');
legend('Señal cuantizada', 'Muestras originales');
grid on;

% Error de cuantización
error_cuantizacion = x_muestreo2 - x_cuantizado;
subplot(3,4,6);
plot(t_muestreo2, error_cuantizacion, 'm-', 'LineWidth', 2);
title('Error de Cuantización');
xlabel('Tiempo (s)');
ylabel('Error (V)');
```

```
grid on;

%% 4. Codificación (Representación binaria)
% Convertir índices de cuantización a binario
codigos_binarios = cell(length(indices_cuantizacion), 1);
for i = 1:length(indices_cuantizacion)
    codigos_binarios{i} = dec2bin(indices_cuantizacion(i), n_bits);
end

% Mostrar algunos códigos binarios
fprintf('=== PROBLEMA 1: SEÑAL SENOIDAL ===\n');
fprintf('Primeros 10 códigos binarios (%d bits):\n', n_bits);
for i = 1:min(10, length(codigos_binarios))
    fprintf('Muestra %2d: Valor = %6.3f V -> Código = %s\n', ...
            i, x_muestreo2(i), codigos_binarios{i});
end

% Graficar códigos binarios (primeros 20)
subplot(3,4,7);
t_mostrar = t_muestreo2(1:min(20, length(t_muestreo2)));
codigos_mostrar = indices_cuantizacion(1:min(20,
length(indices_cuantizacion)));
stem(t_mostrar, codigos_mostrar, 'k-', 'LineWidth', 1.5, 'Marker', 's');
title(['Índices de Cuantización (Primeros ' num2str(length(t_mostrar))
')']);
xlabel('Tiempo (s)');
ylabel('Índice');
grid on;

%% 5. Reconstrucción
% Interpolación para reconstruir la señal
t_reconstruido = 0:0.001:t_final;
x_reconstruido = interp1(t_muestreo2, x_cuantizado, t_reconstruido,
'linear');

subplot(3,4,8);
plot(t_reconstruido, x_reconstruido, 'r-', 'LineWidth', 2);
hold on;
plot(t_analogico, x_analogico, 'b--', 'LineWidth', 1);
title('Señal Reconstruida vs Original');
xlabel('Tiempo (s)');
ylabel('Amplitud (V)');
legend('Reconstruida', 'Original');
grid on;

%% 6. Análisis espectral
N = 1024; % Puntos para FFT
f_axis = linspace(0, fs2, N);

% Espectro de la señal original muestreada
X_original = fft(x_muestreo2, N);
X_original = abs(X_original(1:N/2));

% Espectro de la señal cuantizada
X_cuantizado = fft(x_cuantizado, N);
```

```
X_cuantizado = abs(X_cuantizado(1:N/2));

subplot(3,4,9);
plot(f_axis(1:N/2), X_original, 'b-', 'LineWidth', 1.5);
title('Espectro - Señal Original Muestreada');
xlabel('Frecuencia (Hz)');
ylabel('Magnitud');
grid on;

subplot(3,4,10);
plot(f_axis(1:N/2), X_cuantizado, 'r-', 'LineWidth', 1.5);
title('Espectro - Señal Cuantizada');
xlabel('Frecuencia (Hz)');
ylabel('Magnitud');
grid on;

%% Cálculo de métricas
SNR = 10 * log10(var(x_muestreo2) / var(error_cuantizacion));
fprintf('\nMétricas de calidad:\n');
fprintf('SNR = %.2f dB\n', SNR);
fprintf('Error cuadrático medio = %.6f\n', mean(error_cuantizacion.^2));
fprintf('Resolución de cuantización = %.4f V\n', paso_cuantizacion);
```