```
%% Simulación de vibraciones en puente peatonal colgante
%% Las señales se generaron en función de las ecuaciones propuestas
%% en el artículo: F. Lüleci, F. N. Catbas y O. Avci, “Generative Adversarial
%% Networks for Labeled Acceleration Data Augmentation for Structural %% %%
%% Damage  Detection,"  J. Civ. Struct. Health Monit.,
%% vol. 2 , no. 1, pp. 181–198, (2022).
%%
%% Se usan los métodos de la FFT, Transformada Wavelet y ESPRIT
clear all
close all

%% PARÁMETROS GENERALES
fs = 200;              % Mayor frecuencia de muestreo para alta resolución
t = 0:1/fs:60;         % Vector de tiempo más largo (60 segundos)
N = length(t);         % Número de muestras
ruido = 0.15;          % Nivel de ruido gaussiano (aumentado)

%% SEÑAL SANA (Healthy State)
f1 = 1.25;  A1 = 1.0;   % Modo fundamental (Hz)
f2 = 3.80;  A2 = 0.6;   % Segundo modo
f3 = 7.45;  A3 = 0.3;   % Tercer modo

sana = A1*sin(2*pi*f1*t) + ...
      A2*sin(2*pi*f2*t + pi/4) + ...
      A3*sin(2*pi*f3*t + pi/3);
sana = sana + ruido*randn(size(sana));

%% FALLAS CRÍPTICAS (Difíciles de detectar con FFT/Wavelet)
% Falla 1: Componentes cercanas a frecuencias modales (separación < resolución
FFT)
delta_f = 0.02;        % Separación crítica (Hz)
falla1 = sana + 0.08*sin(2*pi*(f1+delta_f)*t + pi/5) + ... % Componente fantasma
f1+0.02Hz
        0.05*sin(2*pi*(f2-delta_f)*t);                % Componente fantasma f2-0.02Hz

% Falla 2: Modulación de fase lenta (indetectable en espectro de magnitud)
f_mod_phase = 0.015;    % Frecuencia de modulación muy lenta (Hz)
modulacion_phase = 0.8*sin(2*pi*f_mod_phase*t);
falla2 = A1*sin(2*pi*f1*t + modulacion_phase) + ... % Solo afecta al modo
fundamental
        A2*sin(2*pi*f2*t + pi/4) + ...
        A3*sin(2*pi*f3*t + pi/3);
falla2 = falla2 + ruido*randn(size(sana));

% Falla 3: Componentes transitorias periódicas (enmascaradas por ruido)
f_trans = 0.4;         % Frecuencia de ocurrencia de transitorios (Hz)
transitorios = zeros(size(t));
```

```
for i = 1:length(t)
    if mod(t(i), 1/f_trans) < 0.02  % Pulsos cortos cada 2.5 segundos
        transitorios(i) = 0.5*exp(-50*(mod(t(i),1/f_trans)/0.02).^2).*sin(2*pi*15*t(i));
    end
end
falla3 = sana + transitorios;


%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Graficamos las señales sana y con fallas
figure
subplot(4,1,1)
plot(t,sana,'k');
title('Estado Sano')
subplot(4,1,2)
plot(t, falla1,'r');
title('Fallo 1: Dezplazamiento en las frecuencias f1 y f2')
subplot(4,1,3)
plot(t, falla2,'g');
title('Fallo 2: Precencia de modulación de fase lenta')
subplot(4,1,4)
plot(t,falla3,'b');
title('Fallo 3: Componentes transitorias periódicas')

figure
subplot(4,1,1)
plot(t,sana,'k');
xlim([0 10])
title('Estado Sano')
subplot(4,1,2)
plot(t, falla1,'r');
xlim([0 10])
title('Fallo 1: Dezplazamiento en las frecuencias f1 y f2')
subplot(4,1,3)
plot(t, falla2,'g');
xlim([0 10])
title('Fallo 2: Precencia de modulación de fase lenta')
subplot(4,1,4)
plot(t,falla3,'b');
xlim([0 10])
title('Fallo 3: Componentes transitorias periódicas')


%% ANÁLISIS FFT (Muestra limitaciones)
figure('Name', 'Análisis FFT')
frequencies = linspace(0, fs/2, N/2+1);
```

```
% Señal sana
Y_sana = fft(sana);
P2_sana = abs(Y_sana/N);
P1_sana = P2_sana(1:N/2+1);
P1_sana(2:end-1) = 2*P1_sana(2:end-1);
subplot(4,1,1)
plot(frequencies, P1_sana, 'b')
title('FFT - Estado Sano'), xlabel('Frecuencia (Hz)'), ylabel('|P1|')
xlim([0 10]), grid minor, ylim([0 0.6])

% Falla 1
Y_f1 = fft(falla1);
P2_f1 = abs(Y_f1/N);
P1_f1 = P2_f1(1:N/2+1);
P1_f1(2:end-1) = 2*P1_f1(2:end-1);
subplot(4,1,2)
plot(frequencies, P1_f1, 'r')
title('FFT - Falla 1: Componentes fantasmas no distinguibles'),
xlabel('Frecuencia (Hz)'), ylabel('|P1|'), xlim([0 10]), grid minor, ylim([0 0.6])

% Falla 2
Y_f2 = fft(falla2);
P2_f2 = abs(Y_f2/N);
P1_f2 = P2_f2(1:N/2+1);
P1_f2(2:end-1) = 2*P1_f2(2:end-1);
subplot(4,1,3)
plot(frequencies, P1_f2, 'm')
title('FFT - Falla 2: Modulación de fase invisible'),
xlabel('Frecuencia (Hz)'), ylabel('|P1|'), xlim([0 10]), grid minor, ylim([0 0.6])

% Falla 3
Y_f3 = fft(falla3);
P2_f3 = abs(Y_f3/N);
P1_f3 = P2_f3(1:N/2+1);
P1_f3(2:end-1) = 2*P1_f3(2:end-1);
subplot(4,1,4)
plot(frequencies, P1_f3, 'g')
title('FFT - Falla 3: Transitorios no detectables'),
xlabel('Frecuencia (Hz)'), ylabel('|P1|'), xlim([0 20]), grid minor, ylim([0 0.6])

%% ANÁLISIS WAVELET (Muestra limitaciones)
figure('Name', 'Análisis Wavelet (CWT)')
fb = cwtfilterbank('SignalLength', N, 'SamplingFrequency', fs, ...
    'FrequencyLimits', [0.5 20], 'Wavelet', 'amor');

% Señal sana
subplot(4,1,1)
```

```
cwt(sana, 'FilterBank', fb);
title('Wavelet - Estado Sano'), colormap turbo

% Falla 1
subplot(4,1,2)
cwt(falla1, 'FilterBank', fb);
title('Wavelet - Falla 1: Componentes fantasmas no visibles'), colormap turbo

% Falla 2
subplot(4,1,3)
cwt(falla2, 'FilterBank', fb);
title('Wavelet - Falla 2: Modulación de fase no detectable'), colormap turbo

% Falla 3
subplot(4,1,4)
cwt(falla3, 'FilterBank', fb);
title('Wavelet - Falla 3: Transitorios enmascarados por ruido'), colormap turbo

%% ANÁLISIS CON ESPRIT (Detección de fallas con visualización gráfica)
p = 16;                % Número de componentes a estimar

figure('Name', 'Resultados ESPRIT')
subplot(4,1,1)
[freqs_sana, amps_sana] = esprit_detect(sana, fs, p);
stem(freqs_sana, amps_sana, 'b', 'LineWidth', 1.5)
title('ESPRIT - Estado Sano: Componentes principales')
xlabel('Frecuencia (Hz)'), ylabel('Amplitud Relativa')
xlim([0 20]), grid on, ylim([0 1])

subplot(4,1,2)
[freqs_f1, amps_f1] = esprit_detect(falla1, fs, p);
stem(freqs_f1, amps_f1, 'r', 'LineWidth', 1.5)
hold on
% Marcamos componentes fantasma
stem([f1+delta_f, f2-delta_f], [0.08, 0.05], 'k--', 'LineWidth', 1.5)
title('ESPRIT - Falla 1: Detecta componentes fantasmas (líneas negras)')
xlabel('Frecuencia (Hz)'), ylabel('Amplitud Relativa')
xlim([0 20]), grid on, ylim([0 1])
legend('Componentes detectadas', 'Componentes reales añadidas')

subplot(4,1,3)
[freqs_f2, amps_f2] = esprit_detect(falla2, fs, p);
stem(freqs_f2, amps_f2, 'm', 'LineWidth', 1.5)
title('ESPRIT - Falla 2: Detecta distorsión en componentes modales')
xlabel('Frecuencia (Hz)'), ylabel('Amplitud Relativa')
xlim([0 20]), grid on, ylim([0 1])
```

```
subplot(4,1,4)
[freqs_f3, amps_f3] = esprit_detect(falla3, fs, p);
stem(freqs_f3, amps_f3, 'g', 'LineWidth', 1.5)
hold on
% Marcamos componente transitoria
stem(15, 0.3, 'k--', 'LineWidth', 1.5)
title('ESPRIT - Falla 3: Detecta componente transitoria de 15 Hz (línea negra)')
xlabel('Frecuencia (Hz)'), ylabel('Amplitud Relativa')
xlim([0 20]), grid on, ylim([0 1])

%%%%%%%%% Mejora la visualización%%%%%%%%
figure
[freqs_f1, amps_f1] = esprit_detect(falla1, fs, p);
stem(freqs_f1, amps_f1, 'r', 'LineWidth', 1.5)
hold on
% Marcamos componentes fantasma
stem([f1+delta_f, f2-delta_f], [0.08, 0.05], 'k--', 'LineWidth', 1.5)
title('ESPRIT - Falla 1: Detecta componentes fantasmas (líneas negras)')
xlabel('Frecuencia (Hz)'), ylabel('Amplitud Relativa')
xlim([1 4]), grid on, ylim([0 1])
legend('Componentes detectadas', 'Componentes reales añadidas')

%%%%%%%%% Mejora la visualización%%%%%%%%
figure
[freqs_f2, amps_f2] = esprit_detect(falla2, fs, p);
stem(freqs_f2, amps_f2, 'm', 'LineWidth', 1.5)
title('ESPRIT - Falla 2: Detecta distorsión en componentes modales')
xlabel('Frecuencia (Hz)'), ylabel('Amplitud Relativa')
xlim([1.2 1.3]), grid on, ylim([0 1])

%% FUNCIÓN ESPRIT MEJORADA (con estimación de amplitud)
function [frecs, amps] = esprit_detect(signal, fs, p)
    N = length(signal);
    m = floor(N/2);     % Tamaño de subespacio

    % Matriz de Hankel
    H = hankel(signal(1:m), signal(m:N));

    % Descomposición SVD
    [U, S, ~] = svd(H, 'econ');
    U = U(:,1:p);

    % Subespacios
    U1 = U(1:end-1, :);
    U2 = U(2:end, :);

    % Resolver ecuación ESPRIT
```

```
    Phi = U1\U2;

    % Calcular frecuencias
    eigvals = eig(Phi);
    frecs = angle(eigvals);
    frecs = frecs/(2*pi)*fs;
    frecs = abs(frecs(frecs > 0 & frecs < fs/2));

    % Estimar amplitudes usando pseudoinversa
    t_vector = (0:N-1)/fs;
    A = exp(1i*2*pi*frecs(:)*t_vector);
    x_est = pinv(A.') * signal(:);
    amps = abs(x_est);

    % Normalizar amplitudes
    amps = amps/max(amps);

    % Filtrar componentes débiles
    umbral = 0.05;  % 5% de la amplitud máxima
    idx_validos = amps > umbral;
    frecs = frecs(idx_validos);
    amps = amps(idx_validos);

    % Ordenar por frecuencia
    [frecs, idx] = sort(frecs);
    amps = amps(idx);
end
```