1、%% Using X-axis data as an example, filtering method, x\_DOT3 as raw acceleration data

Wc = 2 \* 10 / 60; % computes the normalized cutoff frequency for the Butterworth filter, with 10 Hz as the cutoff frequency and 60 Hz as the sampling frequency

[lags, rxx] = butter(2, Wc, 'low'); % Design a second-order Butterworth low pass filter for X-axis data

x\_FilDOT3 = filtfilt(lags, rxx, x\_DOT3); % Apply the filter to the x\_DOT3 acceleration data

% Plot the original acceleration data for the X-axis

subplot(2, 1, 1);

plot(t\_DOT3, x\_DOT3, 'r', 'linewidth', 1.5);

title('Original Acceleration Data (x\\_DOT3)');

% Plot the filtered acceleration data for the X-axis

subplot(2, 1, 2);

plot(t\_DOT3, x\_FilDOT3, 'r', 'linewidth', 1.5);

title('Filtered Acceleration Data using Butterworth Low-pass Filter (x\\_FilDOT3)');

2、%% Using X-axis data as an example, calculating the Unbiased Autocorrelation Coefficient Sequence

[rxx\_dot3, lags] = xcov(x\_FilDOT3, x\_FilDOT3, 'unbiased'); % X-axis

rxx\_dot3 = rxx\_dot3 - mean(rxx\_dot3); % This line removes the mean from the autocorrelation sequence to center it around zero

% Normalize the coefficients to 1.0 at zero lag

rxx\_dot3 = rxx\_dot3 / rxx\_dot3((numel(rxx\_dot3) + 1) / 2);

% Plot the unbiased autocorrelation coefficient sequence

figure;

subplot(311); plot(lags, rxx\_dot3);

xlim([-300, 300]); % The window length is set at a constant 5 s, 5 \* 60 Hz

xlabel('Lags\\_dot3');

ylabel('Unbiased autocorrelation coefficient');

title('X-axis');

grid on;

3、 % Calculate gait harmonic ratio in AP (X), VT (Z), ML (Y) directions, acceleration data is accx, accz, accy

% Calculate the number of DOT3\_HR

startx\_DOT3 = 43; % Starting position

peri = 31; % Gait period value from Figure 2 (D1)

numHRs\_DOT3 = floor((420 - startx\_DOT3 + 1) / (peri \* 2)); % 420 is the total number of data points, calculating the number of HRs

% Initialize arrays to store each HR

HRs\_AP\_d3 = zeros(1, numHRs\_DOT3);

HRs\_ML\_d3 = zeros(1, numHRs\_DOT3);

HRs\_VT\_d3 = zeros(1, numHRs\_DOT3);

% Calculate the harmonic ratio for each HR

% Set the range for the gait period

startRow = startx\_DOT3;

endRow = startRow + (peri \* 2);

for i = 1:numHRs\_DOT3

% Calculate data within the current gait period

accxD3 = x\_FilDOT3(startRow:endRow, :);

accyD3 = y\_FilDOT3(startRow:endRow, :);

acczD3 = z\_FilDOT3(startRow:endRow, :);

% Check data length consistency

if length(accxD3) ~= length(acczD3) || length(accxD3) ~= length(accyD3)

error('Input data lengths are inconsistent. Please ensure the lengths of acceleration data in three directions are the same.');

end

Fs = 60; % Hz, sampling frequency

N\_d3 = length(acczD3); % Signal length

f = Fs \* (0:N\_d3/2-1) / N\_d3; % Frequency vector

y\_APd3 = fft(accxD3) / N\_d3; % Perform Fourier transform

y\_VTd3 = fft(acczD3) / N\_d3; % Perform Fourier transform

y\_MLd3 = fft(accyD3) / N\_d3; % Perform Fourier transform

% Calculate single-sided amplitude spectrum

NN\_d3 = round(N\_d3 / 2);

xa\_w3 = abs(y\_APd3(1:NN\_d3 + 1, :));

za\_w3 = abs(y\_VTd3(1:NN\_d3 + 1, :));

ya\_w3 = abs(y\_MLd3(1:NN\_d3 + 1, :));

% Remove DC component

xa\_d3 = xa\_w3(2:end, :);

ya\_d3 = ya\_w3(2:end, :);

za\_d3 = za\_w3(2:end, :);

% Extract amplitudes of the first 20 harmonics

harmonics = 20;

if length(xa\_d3) < harmonics || length(ya\_d3) < harmonics || length(za\_d3) < harmonics

error('Number of harmonics exceeds signal length. Please check gait period and data length.');

end

harmonics\_3x = xa\_d3(1:harmonics);

harmonics\_3y = ya\_d3(1:harmonics);

harmonics\_3z = za\_d3(1:harmonics);

% Calculate gait harmonic ratio

HR\_AP\_d3 = sum(harmonics\_3x(2:2:end)) / sum(harmonics\_3x(1:2:end)); % Sum of even harmonics divided by sum of odd harmonics

HR\_ML\_d3 = sum(harmonics\_3y(1:2:end)) / sum(harmonics\_3y(2:2:end)); % Sum of odd harmonics divided by sum of even harmonics

HR\_VT\_d3 = sum(harmonics\_3z(2:2:end)) / sum(harmonics\_3z(1:2:end)); % Sum of even harmonics divided by sum of odd harmonics

% Visualize amplitudes of the first 20 harmonics

figure;

subplot(3, 1, 1);

bar(1:harmonics, harmonics\_3x);

xlabel('D3 Sequence');

ylabel('Amplitude');

title('AP-axis First 20 Harmonics Amplitude');

subplot(3, 1, 2);

bar(1:harmonics, harmonics\_3y);

xlabel('D3 Sequence');

ylabel('Amplitude');

title('ML-axis First 20 Harmonics Amplitude');

subplot(3, 1, 3);

bar(1:harmonics, harmonics\_3z);

xlabel('D3 Sequence');

ylabel('Amplitude');

title('VT-axis First 20 Harmonics Amplitude');

% Store the current HR values

HRs\_AP\_d3(i) = HR\_AP\_d3;

HRs\_ML\_d3(i) = HR\_ML\_d3;

HRs\_VT\_d3(i) = HR\_VT\_d3;

% Update startRow and endRow for the next HR calculation

startRow = endRow + 1;

endRow = startRow + (peri \* 2) - 1; % The length of the gait period is peri \* 2

end

% Calculate the mean HR

mean\_HR\_AP\_d3 = mean(HRs\_AP\_d3);

mean\_HR\_ML\_d3 = mean(HRs\_ML\_d3);

mean\_HR\_VT\_d3 = mean(HRs\_VT\_d3);

% Output results

fprintf('Mean HR\\_AP\\_d3 = %f\n', mean\_HR\_AP\_d3); % Display gait harmonic ratio in the AP direction

fprintf('Mean HR\\_ML\\_d3 = %f\n', mean\_HR\_ML\_d3); % Display gait harmonic ratio in the ML direction

fprintf('Mean HR\\_VT\\_d3 = %f\n', mean\_HR\_VT\_d3); % Display gait harmonic ratio in the VT direction