```
% Checking monostatic signals

% Test name
test = 'bpm_1111';

%% Import Data to MATLAB
test_csv = strcat(test,'.csv');
test_csv_directory = strcat('/Users/george/Desktop/Process Data/output_data/ADC/',↙
test_csv);
test_data = csvread(test_csv_directory);
test_data1 = test_data;

%% Restructure
samplesPerChirp = 512;
chirpsPerFrame = 64;
[numSamples,numRxAnt] = size(test_data);
numFrames = numSamples/(samplesPerChirp*chirpsPerFrame);

numSamples = samplesPerChirp*chirpsPerFrame*numFrames;
test_data = test_data(1:numSamples,:);
test_data = test_data/2^16;

test_data = reshape(test_data,[samplesPerChirp,chirpsPerFrame,numFrames,numRxAnt]);
test_data = permute(test_data,[1,2,4,3]);

%% Print out Phases of Received Chirps Prior to FFTs

sample_num = 400;  % ADC Sample Number out of 512 ADC samples per chirp
res_block = 9;     % Resolution block you want to look at (For 0011, it is a 4 block↙
sequence of chirps)
frame_num = 20;

y = sample_num;
chirp1 = rad2deg(angle(test_data(:,(res_block*4)-3,1,frame_num)));
chirp2 = rad2deg(angle(test_data(:,(res_block*4)-2,1,frame_num)));
chirp3 = rad2deg(angle(test_data(:,(res_block*4)-1,1,frame_num)));
chirp4 = rad2deg(angle(test_data(:,(res_block*4),1,frame_num)));
Tx1 = [chirp1(y) chirp2(y) chirp3(y) chirp4(y)]
chirp1 = rad2deg(angle(test_data(:,(res_block*4)-3,2,frame_num)));
chirp2 = rad2deg(angle(test_data(:,(res_block*4)-2,2,frame_num)));
chirp3 = rad2deg(angle(test_data(:,(res_block*4)-1,2,frame_num)));
chirp4 = rad2deg(angle(test_data(:,(res_block*4),2,frame_num)));
Tx2 = [chirp1(y) chirp2(y) chirp3(y) chirp4(y)]
chirp1 = rad2deg(angle(test_data(:,(res_block*4)-3,3,frame_num)));
chirp2 = rad2deg(angle(test_data(:,(res_block*4)-2,3,frame_num)));
chirp3 = rad2deg(angle(test_data(:,(res_block*4)-1,3,frame_num)));
chirp4 = rad2deg(angle(test_data(:,(res_block*4),3,frame_num)));
Tx3 = [chirp1(y) chirp2(y) chirp3(y) chirp4(y)]
chirp1 = rad2deg(angle(test_data(:,(res_block*4)-3,4,frame_num)));
chirp2 = rad2deg(angle(test_data(:,(res_block*4)-2,4,frame_num)));
chirp3 = rad2deg(angle(test_data(:,(res_block*4)-1,4,frame_num)));
chirp4 = rad2deg(angle(test_data(:,(res_block*4),4,frame_num)));
Tx4 = [chirp1(y) chirp2(y) chirp3(y) chirp4(y)]

%% FFTs
reshapeVector = 1:chirpsPerFrame;
all_dataCubes = zeros(samplesPerChirp,chirpsPerFrame,numRxAnt,numFrames);
```

```
for frameIndex = 1:numFrames
    dataCube = test_data(:,:,:,frameIndex);
    dataCube = dataCube(:,reshapeVector,:);

    for chirpIndex = 1:chirpsPerFrame
        for rxIndex = 1:numRxAnt
            % Blackman window
            dataCube(:,chirpIndex,rxIndex) = dataCube(:,chirpIndex,rxIndex) .* blackman↙
(samplesPerChirp);

            % first FFT
            dataCube(:,chirpIndex,rxIndex) = fft(dataCube(:,chirpIndex,rxIndex));
        end
    end

    for rangeBin = 1:samplesPerChirp
        %for every virtual antenna
        for rxIndex = 1:numRxAnt
            % multiply by hanning window (datacube)
            dataCube(rangeBin,:,rxIndex) = dataCube(rangeBin,:,rxIndex) .* hanning↙
(chirpsPerFrame)';

            % 2D FFT
            dataCube(rangeBin,:,rxIndex) = fft(dataCube(rangeBin,:,rxIndex));
        end
    end
    all_dataCubes(:,:,:,frameIndex) = dataCube;
end

%% Plot the Power vs Bin

% Where to Analyze (arbitrary numbers below... can be changed).
range_bin = 11;
doppler_bin = 1;
frameIndex = 10;
rx_num = 1;

tiledlayout(1,2)

% Across Range Bin
nexttile
plot(20*log10(abs(all_dataCubes(:,doppler_bin,rx_num,frameIndex))));
title('Range FFT')

% Across Velocity Bin
nexttile
plot(20*log10(abs(all_dataCubes(range_bin,:,rx_num,frameIndex))));
title('Doppler FFT')
```