```
% MATLAB Simulation of the Discrete Zak Transform for ECE 4806 - Team
14 (AFRL SDR University Challenge)
% Fall'23 - Spring'24
% Last modified: 30 April 2024
% Created by: Danielle Felder
%% This code simulates a 30x50 matrix through the Discrete Zak
Transform (DZT) at the Baseband level.
%{
The following code simulates the simplified Zak Transform for OTFS.
The matrix A is defined to
contain one complex pilot symbol in the center and four complex data
symbols in the corners of
the grid. Subsequently, the Inverse Fast-Fourier Transform (IFFT) is
applied to each row, followed
by serialization (by column) to facilitate transmission through the
channel (transmitted data).

The channel's impulse response, characterized by three complex paths,
is then convolved with
the transmitted data, accounting for delay-Doppler effects, resulting
in the received data. To
analyze channel estimation, the output results are summed and
reshaped to observe the effects.

The received data is then parallelized (by column) back into an MxN
array shape. Finally, the
Fast-Fourier Transform (FFT) is applied to the rows, resulting in a
matrix that matches Matrix A.
 %}
clc; clear;
%% Matrix Definition & DZT
% Create MxN (50x30) matrix
matrix_A = zeros(50,30);
% Find the center of the matrix
center_row = 25;
center_col = 15;
% Make the center pilot symbol and the data symbols complex
matrix_A(center_row,center_col) = 5+2j; % pilot symbol
matrix_A(5,7) = 2 + 3j; % data symbol
matrix_A(5,23) = 3 - 3j; % data symbol
matrix_A(45,7) = 4 + 3j; % data symbol
matrix_A(45,23) = 5 - 3j; % data symbol

% Take the inverse fast-fourier transform of each row
ifft_A = ifft(matrix_A,[], 2);
```

```
% P/S Conversion
serialized_data = reshape(ifft_A, 1,[]);
% Simulate Channel transmission (Tx)
tx_signal = serialized_data;

% channel estimation
M = 30;
N = 50;
% Create vector
nvec = 0:M*N-1;
nvec = nvec(:);
% delay; Doppler; %channel effects;
dD = [ 0 1 3 ; 0 2 -3 ; 1 1+j 1-j];
[nr, np] = size(dD);
y_path = zeros(M*N,1);
%rp = zeros(M*N, np);
for path = 1:np;
    hd = zeros(1+dD(1,path),1); % impulse response definition
    hd(1+dD(1,path)) = dD(3,path);
    pd = filter(hd,1,tx_signal); % received signal
    fd = -dD(2,path)/(M*N);
    rp(:,path) =
pd'.*exp(j*2*pi*fd*nvec).*exp(-j*2*pi*fd*dD(1,path));% transposed pd
to match dim of nvec
    y_path = y_path + rp(:,path);
end ;

% [num_c, num_r] = size(matrix_A);
 m = reshape(conj(y_path),N,M).';
 %m = reshape((y_path), N,M);

matrix_rx = fft(m,[],1).';
%shift = fftshift(ff);

% Define the post-filtered result from line 55 as the received data
rx_signal = pd;


% S/P Conversion - demodulation
[num_rows,num_columns] = size(ifft_A);
parallelized_data = reshape(rx_signal,num_rows,num_columns);
%parallel_data = parallel_data.';

% Take the fast-fourier tranform of each row
```

```
fft_A = fft(parallelized_data,[],2);

%% Plots
% - Delay-Doppler Domain plots - grid plots

% Plot of the matrix sent through the DZT using bar3 - fig 1
figure;
bar3(abs(matrix_A));
xlabel('Doppler (Frequency (normalized))');
ylabel('Delay (samples)');
zlabel('Amplitude')
title('Matrix before Modulation', 'FontSize',14);
colorbar;

% Plot of the original impulse response using bar3 - fig 2
figure;
surface(abs(matrix_A));
%hold off;
title('Surface View of Pilot Symbol and Data Symbols',
'FontSize',14);
xlabel('Doppler (Frequency (normalized))');
ylabel('Delay (samples)');
zlabel('Amplitude');
colorbar;
grid on;

% Plot of the matrix after iift - fig 3
figure;
bar3(abs(ifft_A));
xlabel('Doppler (Frequency (normalized))');
ylabel('Delay (samples)');
zlabel('Amplitude')
title('IFFT of each Row in the Delay-Doppler Domain', 'FontSize',14);
colorbar;

% Plot of the serialized values to reshape into matrix - fig 4
figure;
plot(abs(serialized_data));
title('Serialized values for the Transmitted Signal', 'FontSize',14);
xlabel('Time');
ylabel('Amplitude');
grid on;


% channel estimation - fig 5
```

```
figure;
surface(abs(matrix_rx));
title('Channel Estimation', 'FontSize',14);
xlabel('Doppler (Frequency (normalized))');
ylabel('Delay (samples)');
grid on;
colorbar;


% Plot of the parallelized values to reshape into matrix - fig 6
figure;
plot(abs(rx_signal));
title('Values for the Received Signal before Parallelization',
'FontSize',14);
xlabel('Time');
ylabel('Amplitude');
grid on;


% Plot of the matrix after fft - fig 7
figure;
bar3(abs(parallelized_data));
xlabel('Doppler (Frequency (normalized))');
ylabel('Delay (samples)');
zlabel('Amplitude')
title('Matrix after Parallelization', 'FontSize',14);
colorbar;


% Plot of the matrix after fft - fig 8
figure;
surface(abs(fft_A));
xlabel('Doppler (Frequency (normalized))');
ylabel('Delay (samples)');
zlabel('Amplitude')
title('Surface View of Final Matrix', 'FontSize',14);
colorbar;


% Plot of the matrix after fft - fig 9
figure;
bar3(abs(fft_A));
xlabel('Doppler (Frequency (normalized))');
ylabel('Delay (samples)');
zlabel('Amplitude')
```

```
title('Matrix after Modulation is Complete', 'FontSize',14);
colorbar;
```