```
%Sagar Shah
%ELE 660 HW 19
%OFDM Channel estimation using pilot insertion method
clear all
clc
close

%   ---------------
%   A: Setting Parameters
%   ---------------
n=256;                                  % Number of bits to be transmitted
M=16;                                   % 16 QAM Constellation
bit2sym = log2(M);                      % Number of bits per symbol
h=[0.1+0.2i  0.2-0.1i 0.2+0.2i 03-0.1i 0.3-0.7i 0.4+0.6i 0.5-0.7i 0.2+0.5i
0.8+0.4i 0.5+0.5i];                       % Channel coefficients
cp_len =length(h)-1;                    % Length of cyclic prefix
no_of_rows=n/bit2sym;
block_size = 64;                        % Size of each ofdm block
no_of_ifft_points = block_size;         % 64 points for the FFT
no_of_fft_points = block_size;          % 64 points for the IFFT

%   ---------------------------------------------
%   B: TRANSMITTER
%   ---------------------------------------------
%   1.  Generate 1 x 256 vector of binary bits
bitdata=randi([0 1],1,n);               % Generation of the binary stream
figure(1);
stem(bitdata);                          % Plotting input binary stream
grid on;
xlabel('data points');ylabel(' bits stream');title('TX binary data');

%   2.  Perform 16 QAM modulation
X=(reshape(bitdata,no_of_rows,bit2sym)); % Seperating two bits for symbols
sym=qammod(bi2de(X),M);                 % 16 QAM Modulation
scatterplot(sym);
title('QAM modulated symbols')  % Plotting QAM symbols
num_columns=length(sym)/block_size;
y = reshape( sym, block_size, num_columns);       % Serial to parellel 1 to
64

%  Create empty matix to put the IFFT'd data
cp_start = block_size-cp_len;
cp_end = block_size;

%  3. Operate columnwise & do CP
for i=1:num_columns
    ifft_data_matrix(:,i) = ifft((y(:,i)),no_of_ifft_points);
    %   Compute Cyclic Prefix
    for j=1:cp_len
       actual_cp(j,i) = ifft_data_matrix(j+cp_start,i);
    end
    %   Copy the CP to the existing block to create the actual OFDM block
    ifft_data(:,i) = vertcat(actual_cp(:,i),ifft_data_matrix(:,i));
end

%  4. Convert to serial stream for transmission
```

```
[rows_ifft_data cols_ifft_data]=size(ifft_data);
len_ofdm_data = rows_ifft_data*cols_ifft_data;

%   Actual OFDM signal to be transmitted
ofdm_signal = reshape(ifft_data, 1, len_ofdm_data);
figure(3);
plot(real(ofdm_signal)); xlabel('Time'); ylabel('Amplitude');
title('OFDM Signal');grid on;

%   5. Pass the ofdm signal through the channel
received_signal=conv(ofdm_signal,conj(h));
l1=length(received_signal);
received_signal(:,(l1-cp_len+1):l1)=[];
received_signal(:,1:cp_len)=[];              % Remove CP
figure(4);
plot(real(received_signal));xlabel('Time'); ylabel('Amplitude');
title('Received Signal W/O CP');grid on;

%   ----------------------------------------
%   C:  RECEIVER
%   ----------------------------------------
%   6.  Convert Data back to "parallel" form to perform FFT
recvd_signal_matrix = reshape(received_signal,block_size, cols_ifft_data);

%   7.  Perform FFT
for i=1:cols_ifft_data
    %   FFT
    RL(:,i)=fft(recvd_signal_matrix(:,i),no_of_fft_points);
    fft_data_matrix(:,i) = RL(:,i)./fft(h',no_of_fft_points);

end

%   8.  Symbol recovery
recvd_serial_data = reshape(fft_data_matrix, 1,(block_size*num_columns));
qam_demodulated_data = qamdemod(recvd_serial_data,M);
qam_demodulated_datas=de2bi(qam_demodulated_data);



%Channel Estimation
m=nextpow2(length(h));
p=block_size/m;
pilot1(1,:)= sym(1:4:64,1);
pilot2(1,:)=RL(1:4:64,1);
HL=pilot2./pilot1;
estm_channel=ifft(HL,16);

display('Known pilots at RX');
pilot2
display('Known pilots at TX');
pilot1
display('Original Channel');
h %Original Channel
display('Estimated Channel');
estm_channel(:,length(h)+1:length(estm_channel))=[];
```

```
estm_channel=conj(estm_channel)
```