Matlab Implementation

QAM Transmitter:

QAM transmitter comprises of an encoder block, which allocates the 16 quantized levels of data to 4 levels of I and Q components each. Both I and Q are pulse shaped using Root Raised Cosine filter and then multiplied with sine and cosine respectively. The two streams are then added together.

Initialization

close all;
clc;
len=100;           % Number of points in calculation
Fd=1;              % Sampling rate of digital message
Fs=1;              % Sampling rate of Analog message
M=16;              % M-ary number

Message Formation

% Mapping to I and Q
msg_d = randint(1,len,[0 1]); % Generating random bits
figure;
stem(msg_d(1:40),'b-','filled'); % plotting digital bit stream
title('Random Bits');
xlabel('Bit Index');
ylabel('Value');

Fig-1 Digital bit stream
Mapping to I and Q

```matlab
msg_a=modmap(msg_d,1,1,'qask',M);
sigI1=msg_a(:,1);
sigQ1=msg_a(:,2);

a=zeros(length(sigI1),15);
b=zeros(length(sigQ1),15);
sigI=[sigI1 a]';
sigI=sigI(:);
sigQ=[sigQ1 b]';
sigQ=sigQ(:);

figure;
plot(sigI(1:800));
title('Inphase component');
figure;
plot(sigQ(1:800));
title('Quadrature component');
```

![Fig-2 Inphase component](image1)

![Fig-3: Quadrature component](image2)
Pulse shaping the signals
A root raised cosine finite impulse response filter is used to filter the data streams before modulation onto the quadrature carriers. When passed through a band limited channel, rectangular pulses suffer from the effects of time dispersion and tend to smear into one another. There is always a danger of intersymbol interference between signals. So pulse shaping eliminates inter-symbol interference by ensuring that at a given symbol instance, the contribution to the response from all other symbols is zero.

```matlab
over = 16; % oversampling factor
rolloff=.5;
pulse = rcosine(1,over,'sqrt',rolloff); %basic raised-cosine pulseshape
[val,pos] = max(pulse);
figure; impz(pulse,1);
title('Impulse Response');
sigI2 = filter(pulse,1,sigI); % signal after pulse shaping
sigI2 = sigI2(pos:length(sigI2)); % discard transient
sigQ2 = filter(pulse,1,sigQ); % signal after pulse shaping
sigQ2 = sigQ2(pos:length(sigQ2)); % discard transient
```

**Fig -4 Root Raised Cosine Filter**
Modulation of I and Q channels
\begin{verbatim}
n=1:length(sigI2);
c=cos(2*pi*n/10); % cosine signal
s=sin(2*pi*n/10); % sine signal
modsigI=sigI2.*c'; % Modulating with cosine
modsigQ=sigQ2.*s'; % Modulating with sine
modsig1 = modsigI+modsigQ;
\end{verbatim}

Addition of noise
\[\text{noise} = 0.07 \times \text{randn(length(modsig1),1)};\]
\[\text{modsig} = \text{modsig1} + \text{noise};\] % Addition of noise to modulated signal

\textbf{Fig-5} Modulated signal after adding noise

\textbf{QAM Receiver}

Demodulation of the received signal is done by using coherent sine and cosine signals. The two streams are then passed through RRC filter. The signal is sampled and decision is taken by the Slicer. The original symbols are generated by decoding I and Q symbols.
Demodulation

\[ \text{recI} = \text{modsig}.*c(1:\text{length(modsig)})'; \quad \% \text{Demodulation of signal I} \]
\[ \text{recQ} = \text{modsig}.*s(1:\text{length(modsig)})'; \quad \% \text{Demodulation of signal Q} \]

Root Raised Cosine Filtering

\[ \text{recI} = \text{filter(pulse,1,recI)}; \quad \% \text{signal after RRC filter} \]
\[ \text{recQ} = \text{filter(pulse,1,recQ)}; \quad \% \text{signal after RRC filter} \]
\[ \text{recI} = \text{recI(pos:end)}; \]
\[ \text{recQ} = \text{recQ(pos:end)}; \]

Low Pass Filtering

\[ \text{Num} = \text{remez(16,[0 0.2 0.3 1],[1 1 0 0])}; \]
\[ \text{recI}\_\text{filt} = \text{filter(Num,1,recI)}; \quad \% \text{Passing received signal I through low pass filter} \]
\[ \text{recQ}\_\text{filt} = \text{filter(Num,1,recQ)}; \quad \% \text{Passing received I through LPF} \]
\[ \text{recI1} = \text{recI}\_\text{filt}(9:\text{end}); \quad \% \text{Truncate response tail} \]
\[ \text{recQ1} = \text{recQ}\_\text{filt}(9:\text{end}); \quad \% \text{Truncate response tail} \]

Sampling

\[ \text{recI2} = \text{recI1}(1:16:\text{length(recI1)}); \]
\[ \text{recQ2} = \text{recQ1}(1:16:\text{length(recQ1)}); \]

Slicer

\[ \text{for } i = 1:\text{length(recI2)} \]
\[ \quad \text{if (recI2(i) > 0)} \]
\[ \quad \quad \text{recI2(i) = 1;} \]
\[ \quad \text{elseif (recI2(i) < 0)} \]
\[ \quad \quad \text{recI2(i) = -1;} \]
\[ \quad \text{end} \]
\[ \quad \text{if (recQ2(i) > 0)} \]
\[ \quad \quad \text{recQ2(i) = 1;} \]
\[ \quad \text{elseif (recQ2(i) < 0)} \]
\[ \quad \quad \text{recQ2(i) = -1;} \]
\[ \quad \text{end} \]
\[ \text{end} \]

\[ \text{sig\_rec} = [\text{recI2 recQ2}]; \quad \% \text{Received signal after detection} \]
\[ \text{sig\_final} = \text{demodmap(sig\_rec,1,1,'qask',16)}; \quad \% \text{Final received signal} \]
Plotting figures

```matlab
figure;
plot(1.8*sigI2(1:500), 'r-'); hold;
plot(recI1(1:500), 'b-'); grid on; % After Demodulation
title('Comparison b/w signals');
xlabel('Index'); ylabel('Amplitude');
legend('Signal B4 Modulation', 'Signal after Demodulation');
figure;
stem(msg_d(1:40), 'r-'); hold; % Original data
stem(sig_final(1:40), 'b-'); grid on; % Received data
title('comparison b/w Original and Recieved Data');
xlabel('index'); ylabel('Integer value');
legend('Original Data', 'Recieved Data');
```

**Fig-7** Comparison between signals before modulation and after demodulation

**Fig-8** Comparison between original data and received data