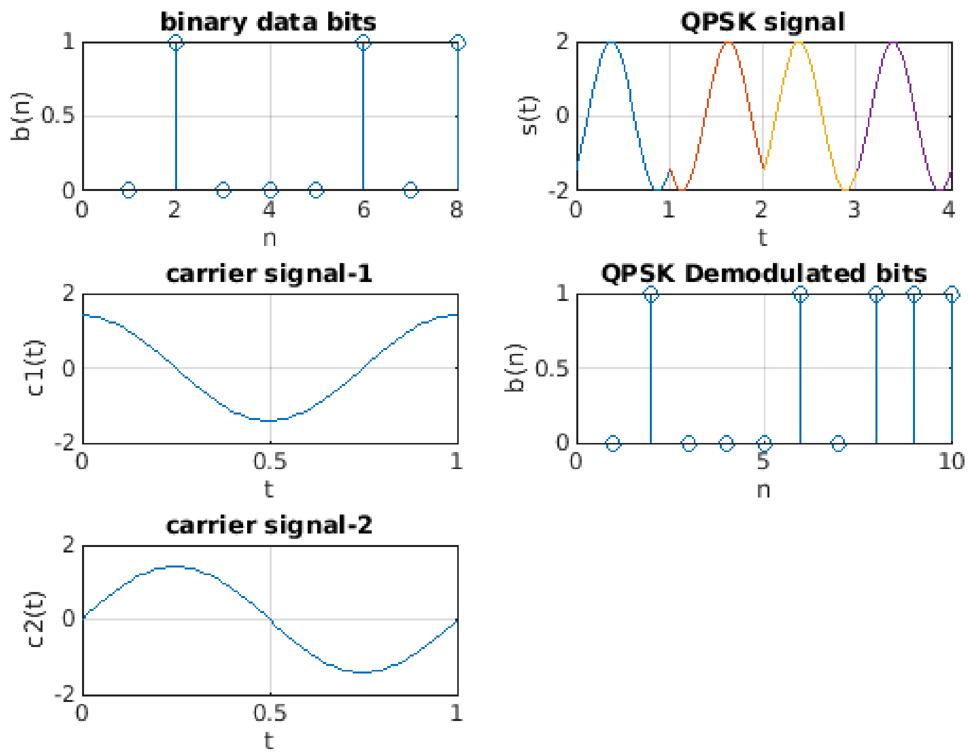
QPSK MODULATION AND DEMODULATION

```
clc;
close all;

Tb=1;
t=0:(Tb/100):Tb;
fc=1;

%Carrier signal
c1=sqrt(2/Tb)*cos(2*pi*fc*t);
c2=sqrt(2/Tb)*sin(2*pi*fc*t);
subplot(3,2,3);
plot(t,c1);
title('carrier signal-1');
xlabel('t');
ylabel('c1(t)');grid on;
subplot(3,2,5);
plot(t,c2);
title('carrier signal-2');
xlabel('t');
ylabel('c2(t)');grid on;

%Message
N=8;
m=rand(1,N);
t1=0;
t2=Tb;
for i=1:2:(N-1)
    t=[t1:(Tb/100):t2];
    if m(i)>0.5
        m(i)=1;
        m_s=ones(1,length(t));
    else
        m(i)=0;
        m_s=-1*ones(1,length(t));
    end

    %Odd bits modulated signal
    odd_sig(i,:)=c1.*m_s;
    if m(i+1)>0.5
        m(i+1)=1;
        m_s=ones(1,length(t));
    else
        m(i+1)=0;
        m_s=-1*ones(1,length(t));
    end

    %Even bits modulated signal
    even_sig(i,:)=c2.*m_s;

    %QPSK signal
```

```
    qpsk=odd_sig+even_sig;

    %QPSK modulated signal
    subplot(3,2,2);
    plot(t,qpsk(i,:));
    title('QPSK signal');
    xlabel('t');
    ylabel('s(t)');
    grid on; hold on;
    t1=t1+(Tb+.01);
    t2=t2+(Tb+.01);
end
hold off

%Input binary data
subplot(3,2,1);
stem(m);
title('binary data bits');
xlabel('n');
ylabel('b(n)');grid on;

% QPSK Demodulation
t1=0;
t2=Tb;
for i=1:N-1
    t=[t1:(Tb/100):t2];
    x1=sum(c1.*qpsk(i,:));
    x2=sum(c2.*qpsk(i,:));
    %Decision device
    if (x1>0&&x2>0)
        demod(i)=1;
        demod(i+1)=1;
    elseif (x1>0&&x2<0)
        demod(i)=1;
        demod(i+1)=0;
    elseif (x1<0&&x2<0)
        demod(i)=0;
        demod(i+1)=0;
    elseif (x1<0&&x2>0)
        demod(i)=0;
        demod(i+1)=1;
    end
    t1=t1+(Tb+.01);
    t2=t2+(Tb+.01);
end
subplot(3,2,4);
stem(demod);
title('QPSK Demodulated bits');
xlabel('n');
ylabel('b(n)');
grid on;
```

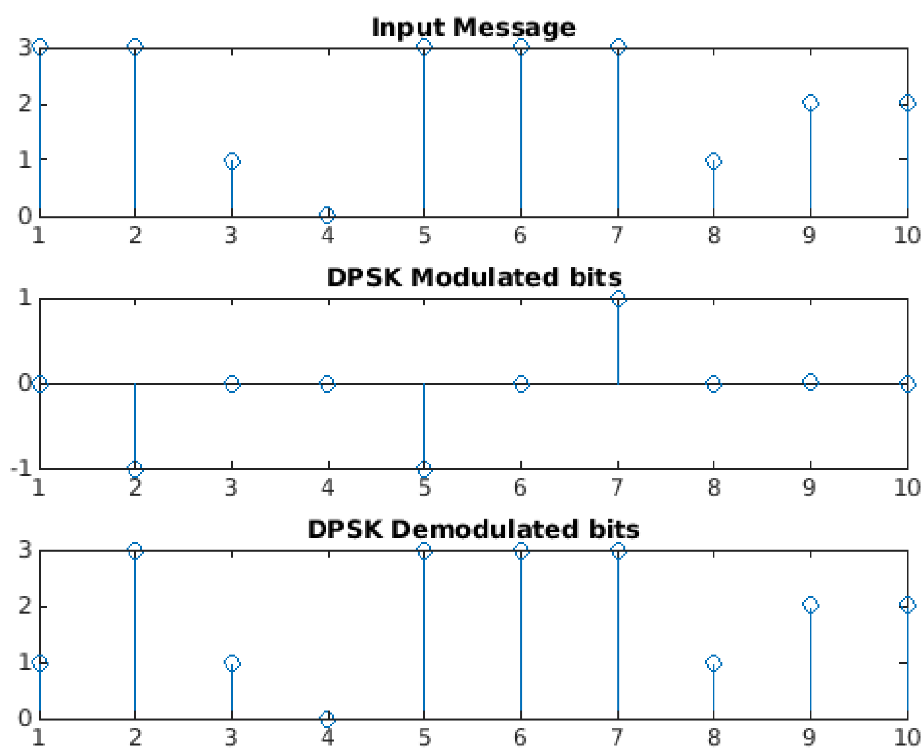


Published with MATLAB® R2020b

DPSK

```
clc;
close all;

M = 4; % Alphabet size
dataIn = randi([0 M-1],10,1); % Random message
subplot(3,1,1);
stem(dataIn);
title('Input Message');
txSig = dpskmod(dataIn,M); % Modulate
subplot(3,1,2);
stem(txSig);
title('DPSK Modulated bits');
rxSig = txSig*exp(2i*pi*rand());
dataOut = dpskdemod(rxSig,M);
subplot(3,1,3);
stem(dataOut);
title('DPSK Demodulated bits');
```

```
Warning: Using only the real component of complex data.
```



Published with MATLAB® R2020b

QAM MODULATION AND DEMODULATION

```
clc;
close all;

M=16;
fprintf('\n\n\n');
% Input  chaking loop
Ld=log2(M);
ds=ceil(Ld);
dif=ds-Ld;
if(dif~=0)
   error('the value of M is only acceptable if log2(M)is an integer');
end
% Binary Information Generation
nbit=32;                        % No. of information bits
msg=round(rand(nbit,1));        % Information generation as binary
 form
%disp(' binary information at transmitter ');
%disp(msg);
%fprintf('\n\n');
% Representation of transmitting binary information as digital signal
 XXX
x=msg;
bp=.000001;                                                     % bit
 period
bit=[];
for n=1:1:length(x)
    if x(n)==1;
       se=ones(1,100);
    else x(n)==0;
        se=zeros(1,100);
    end
     bit=[bit se];
end
t1=bp/100:bp/100:100*length(x)*(bp/100);
figure(1)
subplot(3,1,1);
plot(t1,bit,'lineWidth',2.5);grid on;
axis([ 0 bp*length(x) -.5 1.5]);
ylabel('amplitude(volt)');
xlabel(' time(sec)');
title('transmitting information as digital signal');

% Binary information convert into symbolic form for M-array QAM
 modulation
M=M;                 % order of QAM modulation
msg_reshape=reshape(msg,log2(M),nbit/log2(M))';
%disp('Information are reshaped for convert symbolic form');
%disp(msg_reshape);
%fprintf('\n\n');
size(msg_reshape);
```

```
for(j=1:1:nbit/log2(M))
   for(i=1:1:log2(M))
       a(j,i)=num2str(msg_reshape(j,i));
   end
end

as=bin2dec(a);
ass=as';
figure(1)
subplot(3,1,2);
stem(ass,'Linewidth',2.0);
title('serial symbol for M-array QAM modulation at transmitter');
xlabel('n(discrete time)');
ylabel(' magnitude');
%disp('symbolic form information for M-array QAM ');
%disp(ass);
%fprintf('\n\n');

% Mapping for M-array QAM modulation
M=M;                 %order of QAM modulation
x1=[0:M-1];
p=qammod(ass,M)      %constalation design for M-array QAM acording to
 symbol
sym=0:1:M-1;      % considerable symbol of M-array QAM, just for
 scatterplot
pp=qammod(sym,M);                    %constalation diagram for M-
array QAM
scatterplot(pp),grid on;
title('consttelation diagram for M-array QAM');
%  M-array QAM modulation
RR=real(p)
II=imag(p)
sp=bp*2;           %symbol period for M-array QAM
sr=1/sp;           % symbol rate
f=sr*2;
t=sp/100:sp/100:sp;
ss=length(t);
m=[];

for(k=1:1:length(RR))
    yr=RR(k)*cos(2*pi*f*t);          % inphase or real component
    yim=II(k)*sin(2*pi*f*t);         % Quadrature or imagenary
 component
    y=yr+yim;
    m=[m y];
end
tt=sp/100:sp/100:sp*length(RR);
figure(1);
subplot(3,1,3);
plot(tt,m);
title('waveform for M-array QAM modulation acording to symbolic
 information');
xlabel('time(sec)');
ylabel('amplitude(volt)');
```

```
% M-array QAM demodulation
m1=[];
m2=[];
for n=ss:ss:length(m)
  t=sp/100:sp/100:sp;
  y1=cos(2*pi*f*t);            % inphase component
  y2=sin(2*pi*f*t);            % quadrature component
  mm1=y1.*m((n-(ss-1)):n);
  mm2=y2.*m((n-(ss-1)):n);
  z1=trapz(t,mm1)              % integration
  z2=trapz(t,mm2)
  zz1=round(2*z1/sp)
  zz2=round(2*z2/sp)
  m1=[m1 zz1]
  m2=[m2 zz2]
end
% Demapping for M-array QAM modulation
clear i;
clear j;
for (k=1:1:length(m1))
gt(k)=m1(k)+j*m2(k);
end
gt
ax=qamdemod(gt,M);
figure(3);
subplot(2,1,1);
stem(ax,'linewidth',2);
title(' re-obtain symbol after M-array QAM demodulation ');
xlabel('n(discrete time)');
ylabel(' magnitude');
%disp('re-obtain symbol after M-array QAM demodulation ');
%disp(ax);
%fprintf('\n\n');
bi_in=dec2bin(ax);
[row col]=size(bi_in);
p=1;
 for(i=1:1:row)
     for(j=1:1:col)
         re_bi_in(p)=str2num(bi_in(i,j));
         p=p+1;
     end
 end
%disp('re-obtain binary information after M-array QAM demodulation');
%disp(re_bi_in')
%fprintf('\n\n');
% representation of receiving binary information as digital signal
x=re_bi_in;
bp=.000001;      % bit period
bit=[];
for n=1:1:length(x)
    if x(n)==1;
       se=ones(1,100);
    else x(n)==0;
```

```
        se=zeros(1,100);
    end
     bit=[bit se];
end
t1=bp/100:bp/100:100*length(x)*(bp/100);
figure(3)
subplot(2,1,2);
plot(t1,bit,'lineWidth',2.5);grid on;
axis([ 0 bp*length(x) -.5 1.5]);
ylabel('amplitude(volt)');
xlabel(' time(sec)');
title('receiving information as digital signal after M-array QAM
 demoduation');
```

```
p =

  Columns 1 through 4

  -3.0000 + 1.0000i  -3.0000 + 1.0000i  -1.0000 - 1.0000i   3.0000 -
 3.0000i

  Columns 5 through 8

   3.0000 + 3.0000i   1.0000 - 1.0000i  -1.0000 - 1.0000i   1.0000 -
 3.0000i


RR =

    -3    -3    -1     3     3     1    -1     1


II =

     1     1    -1    -3     3    -1    -1    -3


z1 =

  -2.9417e-06


z2 =

   1.0036e-06


zz1 =

    -3
```

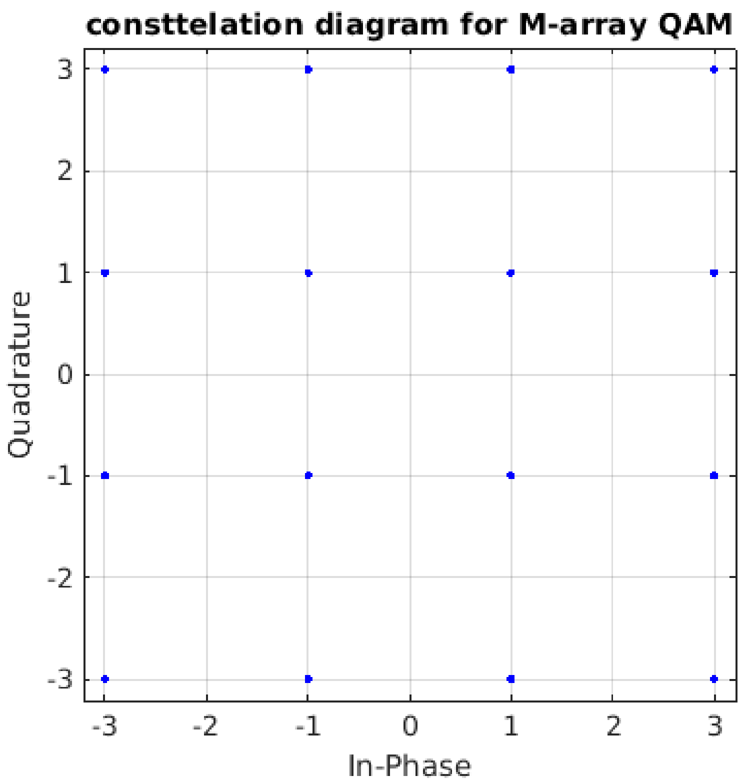
```
zz2 =

     1

m1 =

    -3

m2 =

     1

z1 =

  -2.9417e-06

z2 =

   1.0036e-06

zz1 =

    -3

zz2 =

     1

m1 =

    -3    -3

m2 =

     1     1

z1 =

  -9.7891e-07

z2 =
```

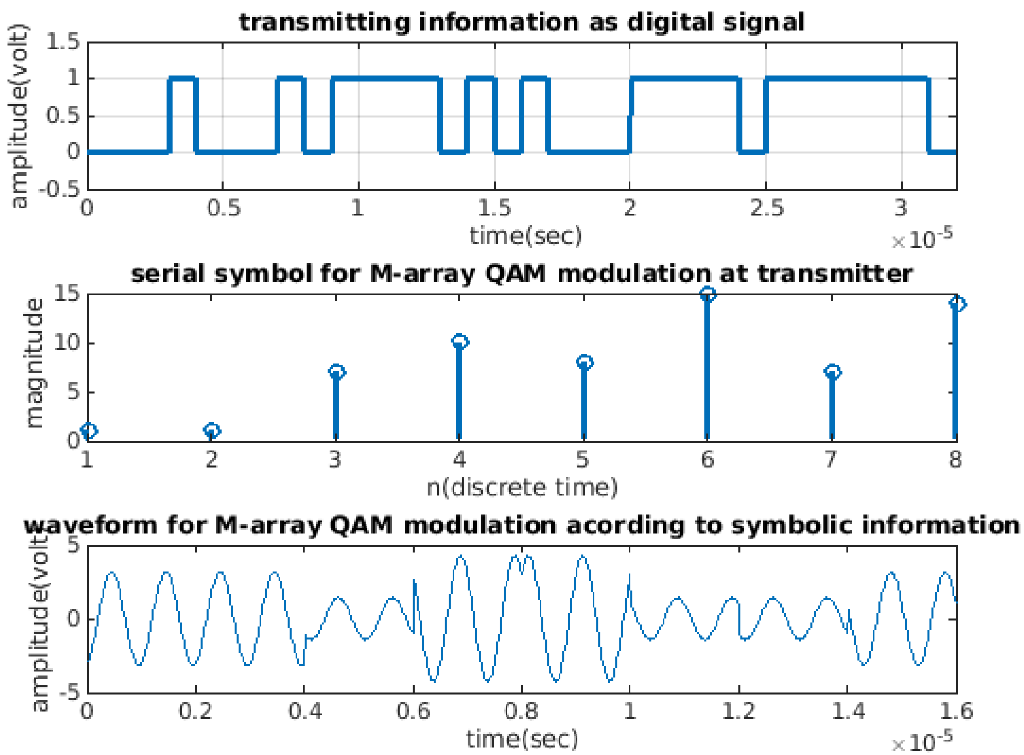
```
  -9.9860e-07

zz1 =

    -1

zz2 =

    -1

m1 =

    -3    -3    -1

m2 =

     1     1    -1

z1 =

   2.9442e-06

z2 =

  -3.0033e-06

zz1 =

     3

zz2 =

    -3

m1 =

    -3    -3    -1     3

m2 =

     1     1    -1    -3

z1 =
```

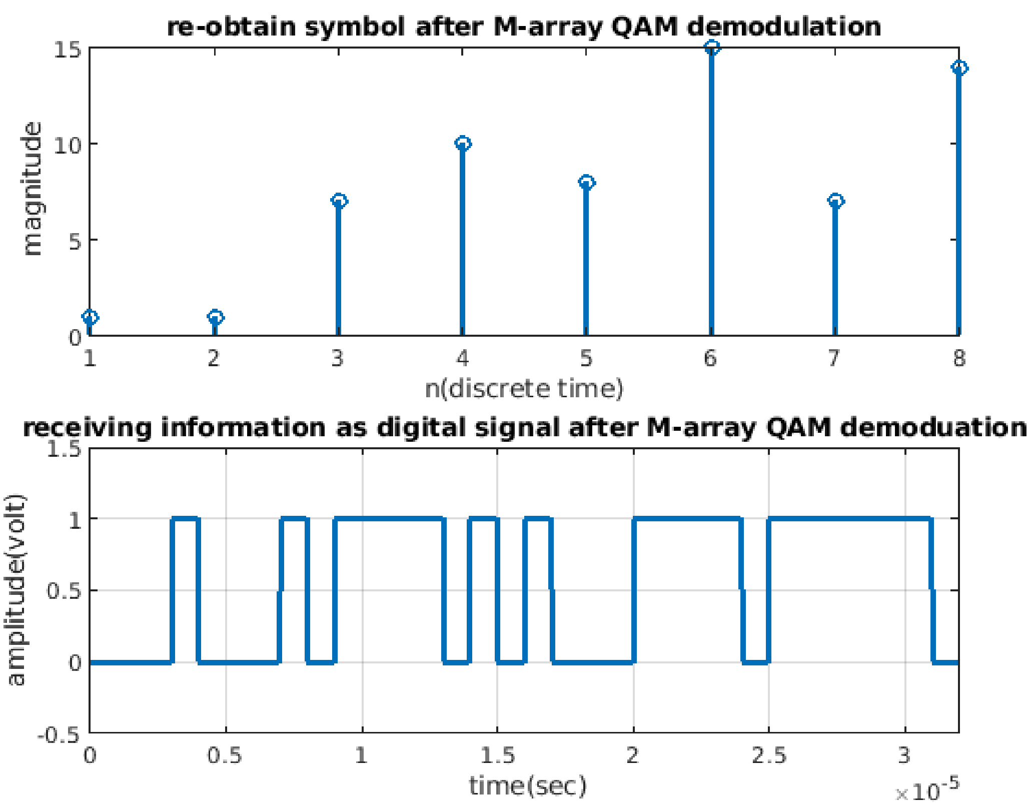
```
   2.9367e-06

z2 =

   2.9958e-06

zz1 =

     3

zz2 =

     3

m1 =

    -3    -3    -1     3     3

m2 =

     1     1    -1    -3     3

z1 =

   9.8140e-07

z2 =

  -1.0011e-06

zz1 =

     1

zz2 =

    -1

m1 =

    -3    -3    -1     3     3     1
```

```
m2 =

     1     1    -1    -3     3    -1


z1 =

  -9.7891e-07


z2 =

  -9.9860e-07


zz1 =

    -1


zz2 =

    -1


m1 =

    -3    -3    -1     3     3     1    -1


m2 =

     1     1    -1    -3     3    -1    -1


z1 =

   9.8389e-07


z2 =

  -3.0008e-06


zz1 =

     1


zz2 =

    -3
```

```
m1 =

    -3    -3    -1     3     3     1    -1     1


m2 =

     1     1    -1    -3     3    -1    -1    -3


gt =

  Columns 1 through 4

  -3.0000 + 1.0000i  -3.0000 + 1.0000i  -1.0000 - 1.0000i   3.0000 -
 3.0000i

  Columns 5 through 8

   3.0000 + 3.0000i   1.0000 - 1.0000i  -1.0000 - 1.0000i   1.0000 -
 3.0000i
```




transmitting information as digital signal
amplitude(volt)
time(sec)
serial symbol for M-array QAM modulation at transmitter
magnitude
n(discrete time)
waveform for M-array QAM modulation acording to symbolic information
amplitude(volt)
time(sec)

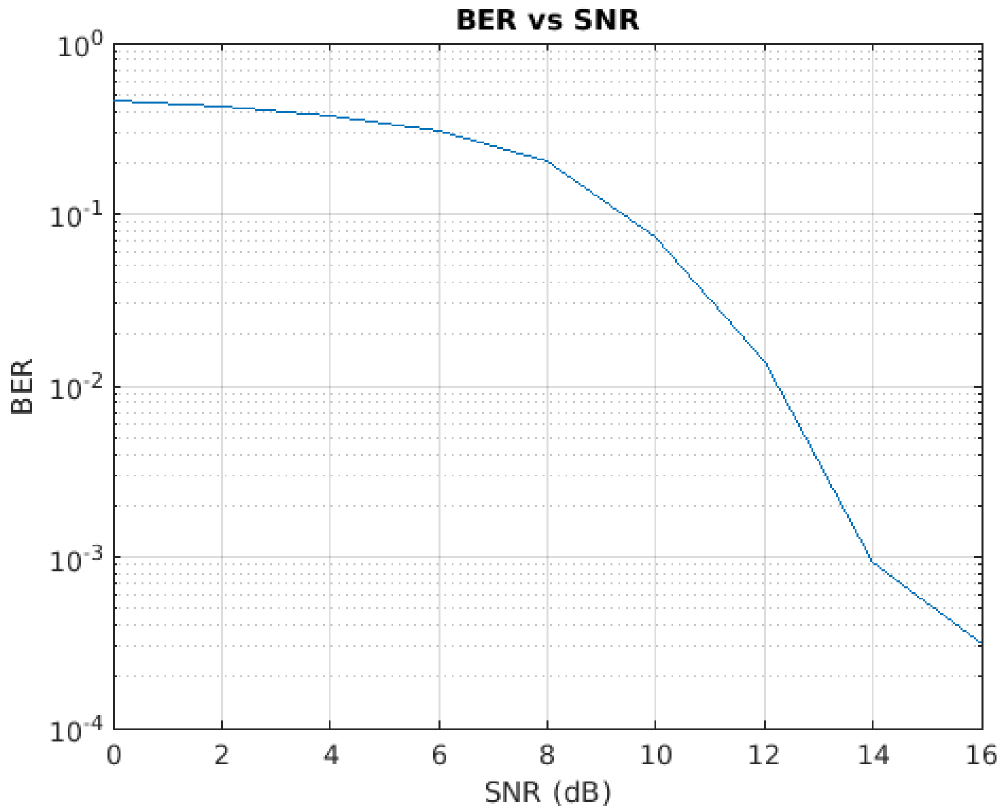

re-obtain symbol after M-array QAM demodulation
magnitude
n(discrete time)
receiving information as digital signal after M-array QAM demoduation
amplitude(volt)
time(sec)

QAM MODULATION AND DEMODULATION

```
clc;
close all;

% Generating and coding data
t_data=randi([0,1], 1, 9600);
x=1;
si=1; %for BER rows

for d=1:100;
data=t_data(x:x+95);
x=x+96;
k=3;
n=6;
s1=size(data,2); % Size of input matrix
j=s1/k;

% Convolutionally encoding data
constlen=7;
codegen = [171 133]; % Polynomial
trellis = poly2trellis(constlen, codegen);
codedata = convenc(data, trellis);

%Interleaving coded data
s2=size(codedata,2);
j=s2/4;
matrix=reshape(codedata,j,4);
intlvddata = matintrlv(matrix',2,2)'; % Interleave.
intlvddata=intlvddata';

% Binary to decimal conversion
dec=bi2de(intlvddata','left-msb');

%16-QAM Modulation
M=16;
y = qammod(dec,M);
% scatterplot(y);

% Pilot insertion
lendata=length(y);
pilt=3+3j;
nofpits=4;
k=1;
for i=(1:13:52)
pilt_data1(i)=pilt;
for j=(i+1:i+12);
pilt_data1(j)=y(k);
k=k+1;
end
end
pilt_data1=pilt_data1'; % size of pilt_data =52
```

```
pilt_data(1:52)=pilt_data1(1:52); % upsizing to 64
pilt_data(13:64)=pilt_data1(1:52); % upsizing to 64
for i=1:52
pilt_data(i+6)=pilt_data1(i);
end

% IFFT
ifft_sig=ifft(pilt_data',64);

% Adding Cyclic Extension
cext_data=zeros(80,1);
cext_data(1:16)=ifft_sig(49:64);
for i=1:64
cext_data(i+16)=ifft_sig(i);
end

% Channel
% SNR
o=1;
for snr=0:2:50
ofdm_sig=awgn(cext_data,snr,'measured'); % Adding white Gaussian Noise

% RECEIVER
%Removing Cyclic Extension
for i=1:64
rxed_sig(i)=ofdm_sig(i+16);
end

% FFT
ff_sig=fft(rxed_sig,64);

% Pilot Synch
for i=1:52
synched_sig1(i)=ff_sig(i+6);
end
k=1;
for i=(1:13:52)
for j=(i+1:i+12);
synched_sig(k)=synched_sig1(j);
k=k+1;
end
end

% Demodulation
dem_data= qamdemod(synched_sig,16);

% Decimal to binary conversion
bin=de2bi(dem_data','left-msb');
bin=bin';

% De-Interleaving
deintlvddata = matdeintrlv(bin,2,2); % De-Interleave
deintlvddata=deintlvddata';
deintlvddata=deintlvddata(:)';
```

```
%Decoding data
n=6;
k=3;
decodedata =vitdec(deintlvddata,trellis,5,'trunc','hard'); % decoding
 datausing veterbi decoder
rxed_data=decodedata;

% Calculating BER
rxed_data=rxed_data(:)';
errors=0;
c=xor(data,rxed_data);
errors=nnz(c);
BER(si,o)=errors/length(data);
o=o+1;
end % SNR loop ends here
si=si+1;
end % main data loop

% Time averaging for optimum results
for col=1:25; %%%change if SNR loop Changed
ber(1,col)=0;
for row=1:100;
ber(1,col)=ber(1,col)+BER(row,col);
end
end
ber=ber./100;

figure
i=0:2:48;
semilogy(i,ber);
title('BER vs SNR');
ylabel('BER');
xlabel('SNR (dB)');
grid on
```



Published with MATLAB® R2020b