Implementation of Matched Median Filter on QAM Signals

Table of Contents

Initialization

```
M = 16;                       % Size of signal constellation
k = log2(M);                  % Number of bits per symbol
n = 30000;                    % Number of bits to process
numSamplesPerSymbol = 1;      % Oversampling factor
rng default                   % Use default random number generator
dataIn = randi([0 1],n,1);    % Generate vector of binary data
dataInMatrix = reshape(dataIn,length(dataIn)/k,k);    % Reshape data
 into binary k-tuples, k = log2(M)
dataSymbolsIn = bi2de(dataInMatrix);
dataMod = qammod(dataSymbolsIn,M,0,'bin');
```

Introducing AWGN Noise

```
% Defining Eb/No ratio to generate a noisy signal (AWGN Noise)

EbNo = 10;

snr = EbNo + 10*log10(k) - 10*log10(numSamplesPerSymbol);
receivedSignal = awgn(dataMod,snr,'measured');
```

QAM Demodulation

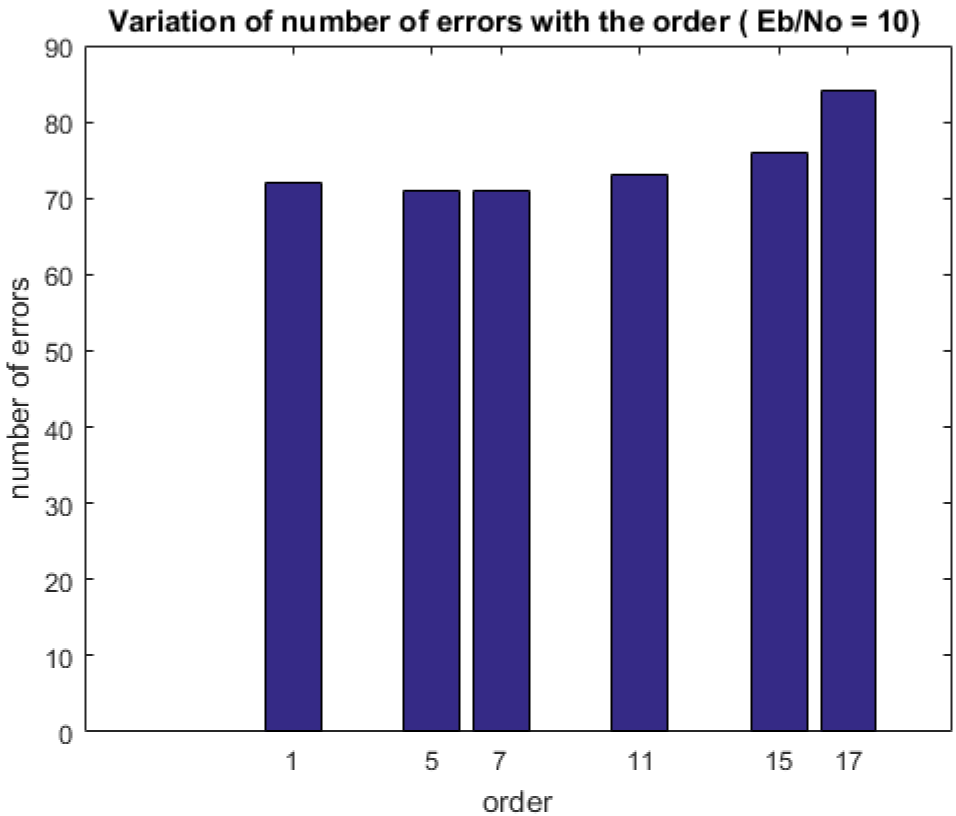
```
% Without median filtering

dataSymbolsOut = qamdemod(receivedSignal,M,0,'bin');
dataOutMatrix = de2bi(dataSymbolsOut,k);
dataOut = dataOutMatrix(:);

% With median filtering


% Complex to Constellation Diagram
```

```
mag=abs(receivedSignal);
phase=angle(receivedSignal);

sm1=mag.*cos(phase);
sm2=mag.*sin(phase);

% The orthogonal basis

fs=100;
t=0:(1/fs):1;
cosine=(1/sqrt(51))*cos(2*pi*t);
sine=(1/sqrt(50))*sin(2*pi*t);

% Reconstructing the QAM signal

result=sm1*cosine +sm2*sine;

% Signal without filter

result_nf=result;

% Apply median filtering

i=1;

while i<=7500
    result(i,:)=medfilt1(result(i,:),7);
    i=i+1;
end

% To get the coefficients of the Basis functions

cos_coeff=result*cosine';
sin_coeff=result*sine';

% To get the phase angle

phase_result=atan(sin_coeff./cos_coeff);

% As the phase crosses pi, we need to limit the value to obtain same
% values

i=1;
while i<=length(phase_result)

        if phase(i) < -pi/2
            phase_result(i) = phase_result(i) - pi;
        end
        if phase(i) > pi/2
            phase_result(i) = pi + phase_result(i);
        end
        i=i+1;
```

```
end

% Calculating amplitude

mag_result=(cos_coeff.^2 + sin_coeff.^2).^0.5;

% Final filtered QAM signal

filtered_result=mag_result.*exp(j*phase_result);


% Demodulation of filtered signal

filtered_dataoutput=qamdemod(filtered_result,M,0,'bin');
filtered_binaryresult=de2bi(filtered_dataoutput,k);
filtered_out=filtered_binaryresult(:);

%  Bit error Rate

[numErrors1,ber1] = biterr(dataIn,dataOut);

[numErrors2,ber2]=biterr(dataIn,filtered_out);

% Calculating number of errors with different orders

nE1 = mdnFilter(dataIn,receivedSignal, 1);
nE5 = mdnFilter(dataIn,receivedSignal, 5);
nE7 = mdnFilter(dataIn,receivedSignal, 7);
nE11 = mdnFilter(dataIn,receivedSignal, 11);
nE15 = mdnFilter(dataIn,receivedSignal, 15);
nE17 = mdnFilter(dataIn,receivedSignal, 17);

nE =[nE1,nE5,nE7,nE11,nE15,nE17];
n =[1,5,7,11,15,17];
```

Error rate with Eb/No

```
i=1;
u=[];
v=[];
while i<=15
    a=qamerr(i,dataMod,dataIn,'Without_Filter');
    b = qamerr(i,dataMod,dataIn,'Filter');
    u = [u;a];
    v = [v;b];
    i=i+1;
end
u=u(:);

bar(n,nE);
xlabel('order');
ylabel('number of errors');
title('Variation of number of errors with the order ( Eb/No = 10) ');
```



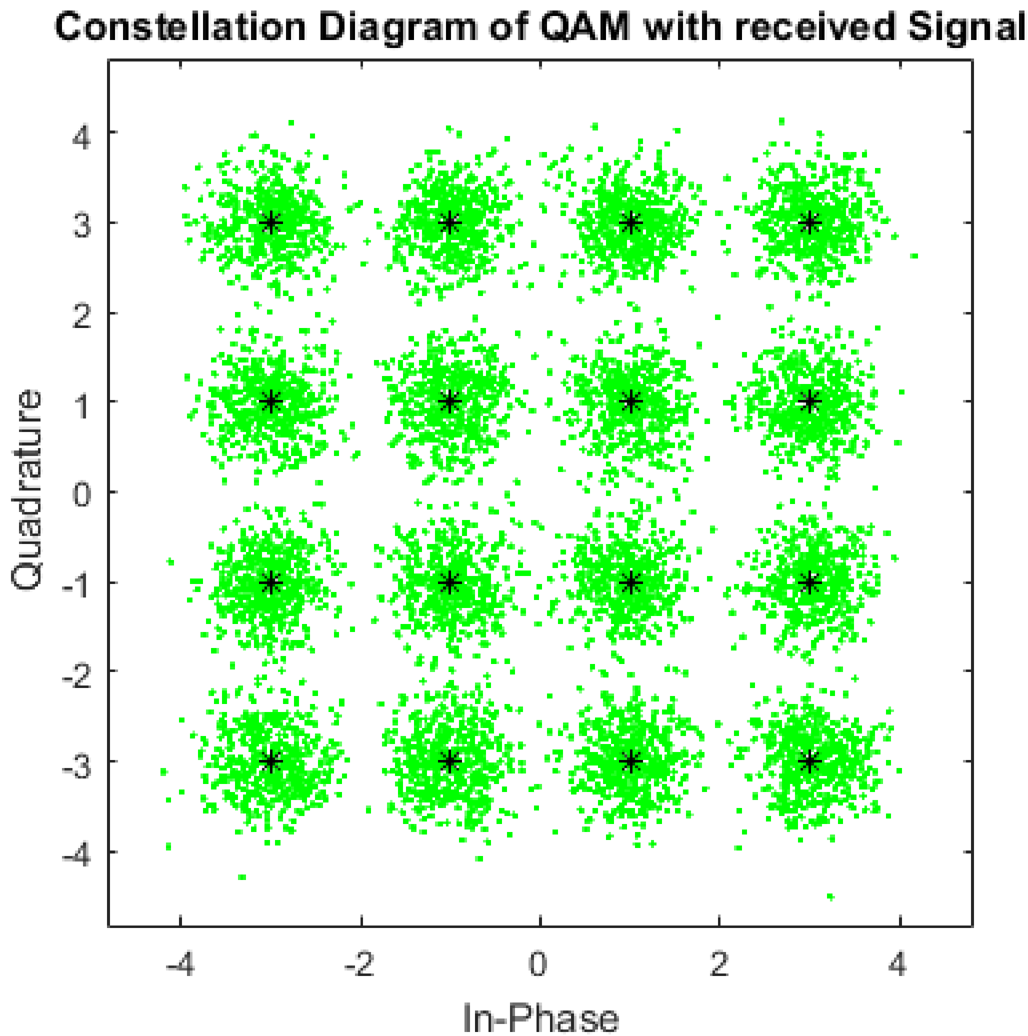
Visualizing the Signals

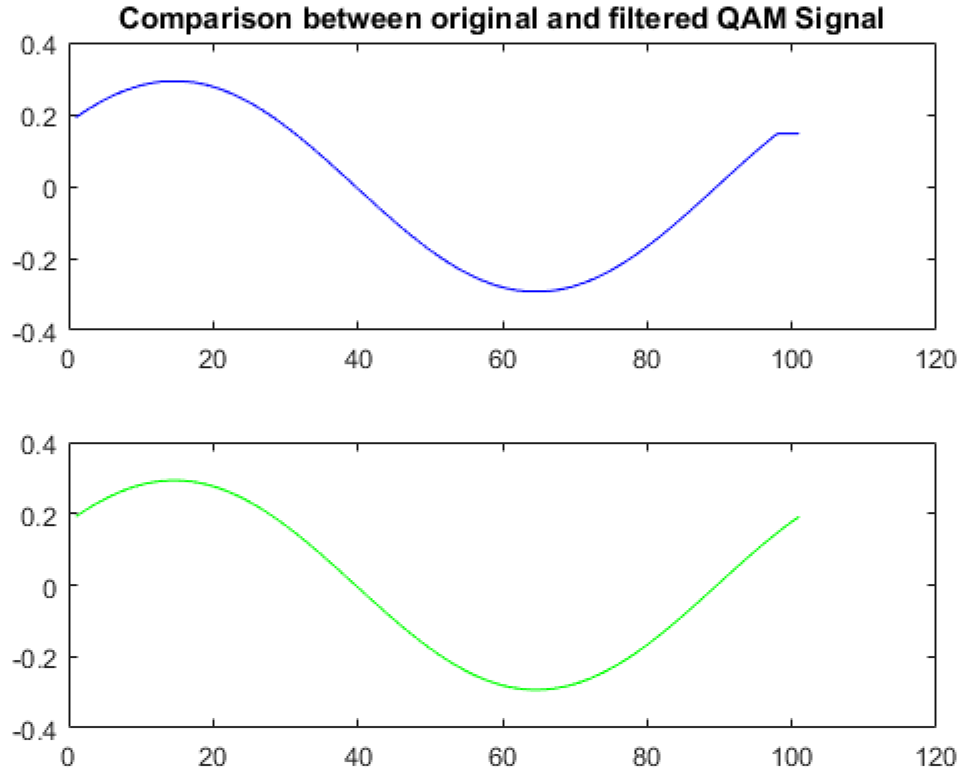
```
sPlotFig = scatterplot(receivedSignal,1,0,'g.');
hold on
scatterplot(dataMod,1,0,'k*',sPlotFig);
title('Constellation Diagram of QAM with received Signal');
```

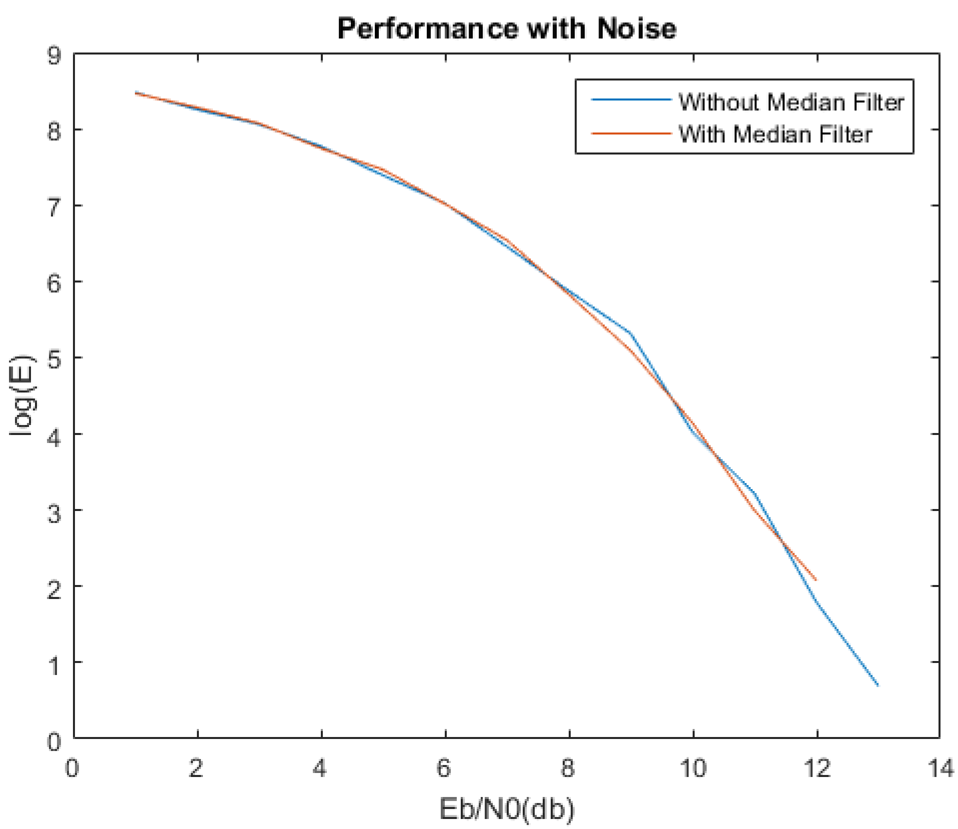
```
figure();
subplot(2,1,1);
plot(result(1,:),'b');
title('Comparison between original and filtered QAM Signal');
subplot(2,1,2);
plot(result_nf(1,:),'g');
```

```
figure;
plot(log(u));
hold on;
plot(log(v));
xlabel('Eb/N0(db)');
ylabel('log(E)');
```

```
legend('Without Median Filter', 'With Median Filter');
title('Performance with Noise');
```




Comparison between original and filtered QAM Signal
0.4
0.2
0
-0.2
-0.4
0
20
40
60
80
100
120


Performance with Noise
Without Median Filter
With Median Filter
log(E)
Eb/N0(db)
0
1
2
3
4
5
6
7
8
9
0
2
4
6
8
10
12
14

User defined Functions

```
% mdnFilter
% This function is used to calculate the number of errors giving the
% original data input signal, and the filtered signal with different
 orders
% function [numErrors] = mdnFilter(dataIn,receivedSignal,order)
% M = 16;                        % Size of signal constellation
% k = log2(M);                   % Number of bits per symbol
%
% mag=abs(receivedSignal);
% phase=angle(receivedSignal);
%
% sm1=mag.*cos(phase);
% sm2=mag.*sin(phase);
%
% % the orthogonal basis
% fs=100;
% t=0:(1/fs):1;
% cosine=(1/sqrt(51))*cos(2*pi*t);
% sine=(1/sqrt(50))*sin(2*pi*t);
%
% % QAM signal
% result=sm1*cosine +sm2*sine;
%
% % Apply median filtering here using built-in function
% i=1;
%
% while i<=7500
%     result(i,:)=medfilt1(result(i,:),order);
%     i=i+1;
% end
%
% % to get the coefficients of the sine and cos terms
% cos_coeff=result*cosine';
% sin_coeff=result*sine';
%
% % to get the phase angle
% phase_result=atan(sin_coeff./cos_coeff);
%
% % as the phase crosses \pi, we need to limit the value to obtain
 same
% % values
% i=1;
% while i<=length(phase_result)
%
%         if phase(i) < -pi/2
%             phase_result(i) = phase_result(i) - pi;
%         end
%         if phase(i) > pi/2
%             phase_result(i) = pi + phase_result(i);
%         end
%         i=i+1;
```

```
%
% end
%
% % calculating amplitude
% mag_result=(cos_coeff.^2 + sin_coeff.^2).^0.5;
%
% % final filtered QAM signal
% filtered_result=mag_result.*exp(j*phase_result);
%
%
% % Demodulation of filtered signal
% filtered_dataoutput=qamdemod(filtered_result,M,0,'bin');
% filtered_binaryresult=de2bi(filtered_dataoutput,k);
% filtered_out=filtered_binaryresult(:);
% [numErrors,~]=biterr(dataIn,filtered_out);
%
%
% end


% qamerr
% This function calculates the number of errors for different Eb/
N0(SNR)
% values
% function [ error ] = qamerr( EbNo,dataMod,dataIn,char)
%
% k=4;
% numSamplesPerSymbol=1;
%
% snr = EbNo + 10*log10(k) - 10*log10(numSamplesPerSymbol);
% receivedSignal = awgn(dataMod,snr,'measured');
%
% % QAM Demodulation
%
% if strcmp(char,'Without_filter')
%
%     % (A) Without filtering
%
%     dataSymbolsOut = qamdemod(receivedSignal,16,0,'bin');
%     dataOutMatrix = de2bi(dataSymbolsOut,k);
%     dataOut = dataOutMatrix(:);
%
%     [error,ber] = biterr(dataIn,dataOut);
% else
%
% error = mdnFilter(dataIn,receivedSignal, 7);
%
% end

% end
```

Observations

```
% The median filter was observed to be performing better with an order
 of
% 7.
% We can observe the effect of Noise on the the communication system
% through the constellation diagrams.As the noise is increased,the
 points
% become more clustered around the signal and as the noise is
 reduced,it is
% more scattered.
% The vector encoded signals were converted into their sinusoidal
 forms for
% visualizing the effect of filtering .It can be observed that the
% filtering makes the signals very smooth and devoid of impulsive
% variations.
% The number of errors in detection has reduced in the case of
 filtered
% signals although not by a significant amount.
```

Conclusions

```
% The following conclusions can be drawn from the Implementation

% The matched median filter gave better results than the output
 obtained
% without the filter.The number of errors depended on the order of the
 median filter. The
% observations conclude that the filter worked best when the order was
 7
% and the errors increased when the order was lesser or greater than
 that.The number of errors is reduced effectively by the median filter
 when the
% Eb/N0 ratio is between 7 to 11.
% It can be concluded that a median filter has it's importance
 magnified in
% a system having impulsive noise.Even though Additive gaussian noise
 was
% used in the implementation,the effect of the filter can be clearly
 seen.
```

Published with MATLAB® R2015b