Matched Filter in octave

by Aswin Ajayan

November 25, 2018

Procedure followed in simulations:
first the pulse vector p was created in octave with sufficient data points and supplied sampling frequency fs. the point at which the point p falls below 10^{-2}, supplied to the program as "tolerance" was found out and the sequence $< p >$ was truncated (as p_trunc).

length of the sequence p_trunc for a sampling frequency of 10 Hz and tolerance of 0.01 error in amplitude was found to be 27.

10000 symbols were generated randomly using rand function as

$$sk = -1 + 2 * (rand(1, k) > 0.5);$$

The symbols were modulated with the truncated pulse p_trunc and passed through the filter $h[n]$. impulse invariance was used for sampling

$$i.e. \boxed{h_d[n] = T_s * h_c(nT_s)}$$

$$(\sum_k s_k.p_{trunc}[n - kN]) * h_{trunc}[n]$$

$h[n]$ was truncated to an amplitude error tolerance of 10^{-3}.

AWGN noise was added to the modulated signal using randn function

$$tx\_sig = signal + f(\sigma). * randn();$$

where randn follows a normal distribution;

matched filter was constructed as $g[-n]$ where

$$g = p_{trunc} * h_{trunc}$$

the recieved message was convolved with the matched filter and sampled at intervals of L where L is the length of p_{trunc} and bit error rate was calculated and plotted

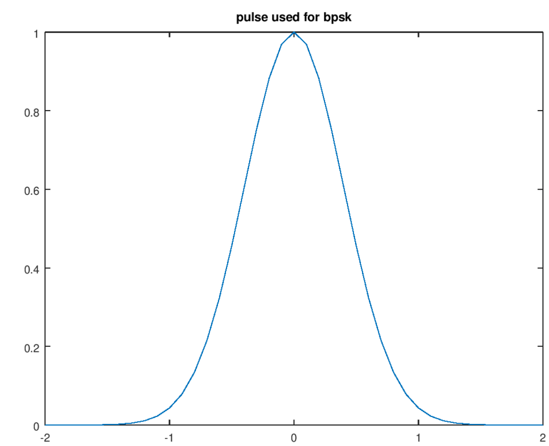

Figure 1.1: truncated pulse used for signalling

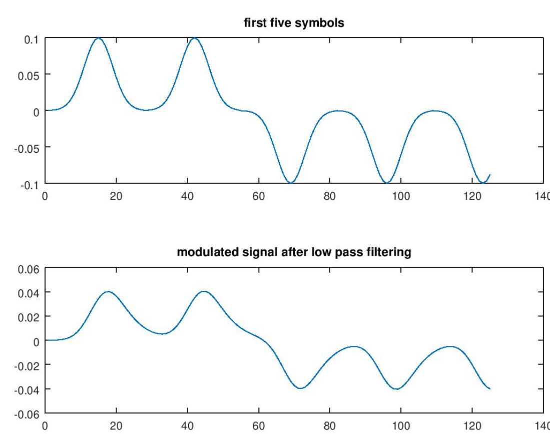

Figure 1.2: transmitted signal (noise is not introduced)

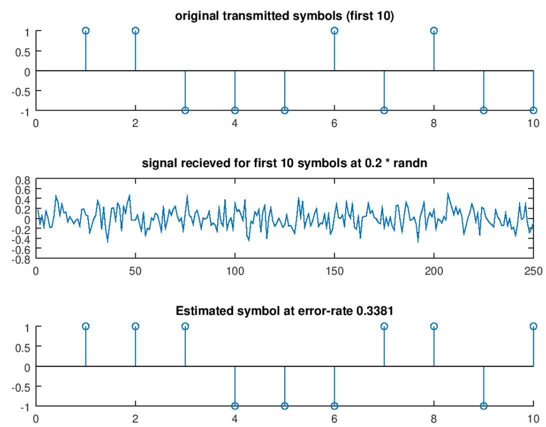

Figure 1.3: symbol detection using matched filter reciever

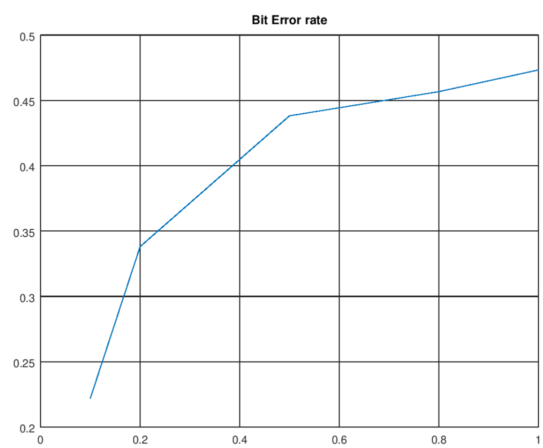

Figure 1.4: bit error rated plotted against variance parameteres

Aliasing considerations

2.1 aliasing in pulse

pulse wave form $p(t) = e^{-\pi t^2}$

$$\rightarrow P(f) = e^{-\pi f^2}$$

after sampling with a frequency f_s, the Fourier transform of the sampled sequence is

$$X_s(f) = \sum_k X(f - kf_s)$$

given that at zero frequency, aliasing should not be more than 10^{-2} i.e

$$\sum_{k \neq 0} X(f - kf_s) < 10^{-2}$$

$$\sum_{k \neq 0} e^{-\pi k^2 fs^2} < 10^{-2}$$

for the minimum value of f_s ,

$$2 * \sum_{k=1}^{\infty} e^{-\pi k^2 fs^2} < 10^{-2}$$

```
close all;
fs = 0:0.01:1;
k = 1:10;
y = zeros(1,length(fs));
temp = ones(1,10);
for i = 1:length(fs)
    for ki = k
        temp = exp(-1*pi*(ki*fs(i))^2);
    end
    y(1,i) = sum(temp);
end
plot(fs,y);
hold on;
tol = ones(1,length(fs))*0.01;
plot(fs,tol);
```

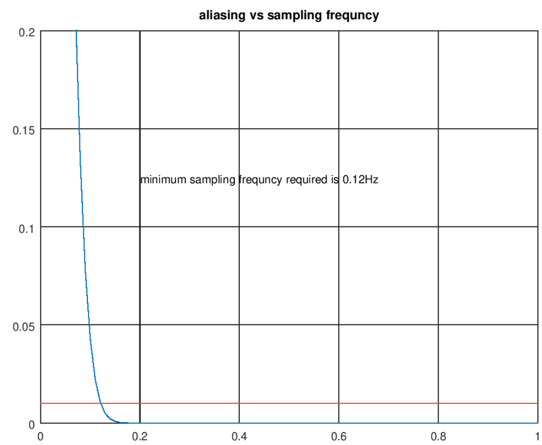

Figure 2.1: minimum fs to avoid aliasing

```
title("aliasing vs sampling frequncy");
ylim([0 0.2])
grid on;
idx = lookup(y,0.01);
text(0.2,0.125,["minimum sampling frequncy required is " num2str(fs(↩
    idx)) "Hz"]);
print -dpng "aliasing.png"
```

2.2 code

```
close all;
clear all;

%setting the parameters
%let fs be the sampling frequency
    fs = 10;
    tolerance = 0.01;
%generating a pulse
    t = -2:1/fs:2;
    p = exp(-pi*t.^2);
    figure;
        plot(t,p);
        title("pulse used for bpsk");
        print -dpng "pulse.png"
%finding the points at which amplitude goes beyond <tolerance ↩
    specified>
    idx = lookup(p((fs*2)+1:end),tolerance);
    limit=t(idx+(fs*2));
    disp(["p at " num2str(limit) " is " num2str(exp(-pi*limit*limit)↩
        )] )
    disp(["data rate = " num2str(1/limit) ])
    width = limit;
%need to truncate the pulse in the limits .
    t_trunc = t((2*fs)-idx:(2*fs)+idx);
```

```
    p_trunc = p((2*fs)-idx:(2*fs)+idx)/fs;  %multiplication by Ts as ↩
        dictated by impulse invariance method
    disp([" length of the sequence P_trunc = " num2str(length(p_trunc))↩
        ])
%modelling the low pass filter , taking the sampling freq(fs = 10) and↩
     tolerance = 0.001
    tolerance = 0.001
    RC = 0.5;
    t = 0:1/fs:3;
    h = exp(-t/RC);
    idx = lookup(h,tolerance);
    h_trunc = 2*h(1:idx)/fs; %h is mutlipied by 2 to normalise the ↩
        gain .
    disp([" length of the sequence h at " num2str(fs) " and tolerance ↩
        " num2str(tolerance) " is " num2str(length(h_trunc))])
    figure
        plot(t,h);
        title("low pass filter at sending end");
%calculating the value of 'N' , given Rb = 0.4
    N = fs/0.4;
%generating random symbols for bpsk modulation
    k = 10000; %be the number of arbitrary symbols
    sk  = -1 + 2*(rand(1,k)>0.5);


%generating pulses to be transmitted .
    mod_sig = [];
       for i = sk
            mod_sig =[mod_sig i.*[p_trunc  zeros(1,N-length(p_trunc))↩
                ]];
    endfor
%%plotting the modulated signal for first five symbol durations
        figure;
        subplot(2,1,1)
            plot(mod_sig(1:5*N));
            title(" first five symbols ");

%low pass filtering by h_n
    mod_sig_filtered = conv(h_trunc,mod_sig);
        subplot(2,1,2)
            plot(mod_sig_filtered(1:5*N));
            title("modulated signal after low pass filtering")
            print -djpg "modulatedsignal.jpg"
%addition of  gaussian noise
    len_tx_sig = length(mod_sig_filtered);
    tx_sig = zeros(5,len_tx_sig);
    len_tx_sig = length(mod_sig_filtered);
    variance_param = [0.1  0.2  0.5  0.8  1.0];
    %multiplication by this parameter changes amplitude of the ↩
        gaussian noise
    for i = 1:5
        tx_sig(i,:) = mod_sig_filtered+ variance_param(i).*randn(1,↩
            len_tx_sig);

    endfor
%matched filter reciever
    basis = conv(p_trunc,h_trunc);
    matched_filter = basis(end:-1:1);
%decoding of symbols
    sk_hat = zeros(5,k);
    for i = 1:5
        matched_filter_out = conv(tx_sig(i,:),matched_filter);
        sk_hat(i,:) = -1 + 2.*(matched_filter_out(2*N:length(p_trunc):↩
            end-length(matched_filter))>0);
    end

%calculation of bit error rate
    error_rate = zeros(1,5);
```

```
    for i = 1:5
        error_rate(i) = sum((sk - sk_hat(i,1:k)) != 0) / k;
    end
    figure;

        plot([0.1 0.2 0.5 0.8 1.0],error_rate);
        title("Bit Error rate");
        grid on;
        print -dpng "bit-error-rate.png"

%plotting symbols

    figure;
    subplot(3,1,1)
        stem(sk(1:10));
        title("original transmitted symbols (first 10)");

    subplot(3,1,2)
        plot(tx_sig(2,1:10*N));
        title(["signal recieved for first 10 symbols at 0.2 * randn"] ↩
            );
    subplot(3,1,3)
        stem(sk_hat(2,1:10));
        title(["Estimated symbol at error-rate " num2str(error_rate(2)↩
            )]);
    print -dpng "symbol-detection.png"
```