Experiment No: 5

Pulse Shaping and Matched Filtering

Objective

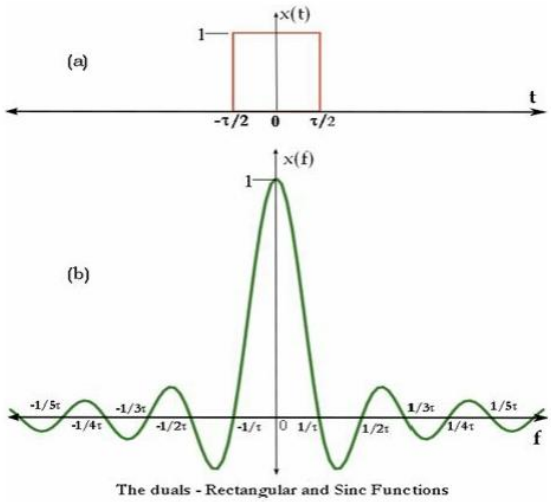
1. Generate a string of message bits.
2. Use root raised cosine pulse *p(t)* as the shaping pulse, and generate the corresponding baseband signal with a fixed bit duration Tb. You may use roll-off factor as $\alpha = 0.4$.
3. Simulate transmission of baseband signal via an AWGN channel
4. Apply matched filter with frequency response $Pr(f) = P*(f)$ to the received signal.
5. Sample the signal at mTb and compare it against the message sequence.

Software Used:

MATLAB R2018a

Theory:

In signal processing, a matched filter (originally known as a North filter) is obtained by correlating a known signal, or template, with an unknown signal to detect the presence of the template in the unknown signal. This is equivalent to convolving the unknown signal with a conjugated time-reversed version of the template.



The duals - Rectangular and Sinc Functions

The matched filter is the optimal linear filter for maximizing the signal to noise ratio (SNR) in the presence of additive stochastic noise. Matched filters are commonly used in radar, in which a known signal is sent out, and the reflected signal is examined for common elements of the out-going signal. Pulse compression is an example of matched filtering. It is so called because impulse response is matched to input pulse signals. Two-dimensional matched filters are commonly used in image processing, e.g., to improve SNR for X-ray.

The matched filter is also used in communications. In the context of a communication system that sends binary messages from the transmitter to the receiver across a noisy channel, a matched filter can be used to detect the transmitted pulses in the noisy received signal. In signal processing, a root-raised-

cosine filter (RRC), sometimes known as square-root-raised-cosine filter (SRRC), is frequently used as the transmit and receive filter in a digital communication system to perform matched filtering. This helps in minimizing inter-symbol interference (ISI).

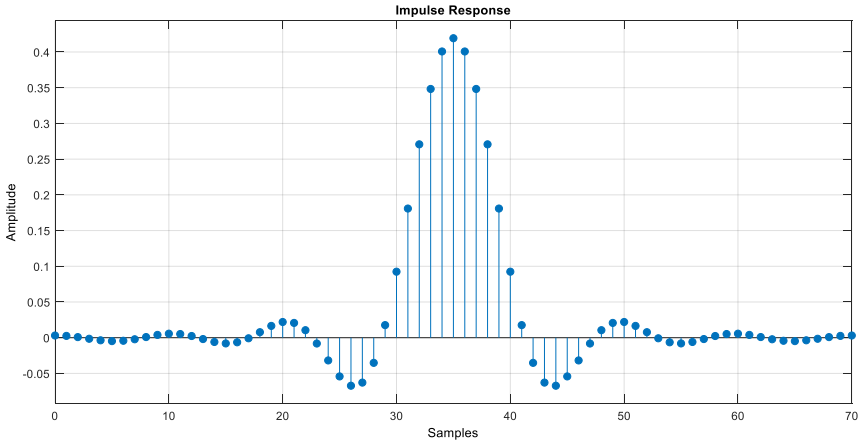
$$p(t) = \begin{cases} \frac{\pi}{4} sinc\left(\frac{nT_s}{\tau}\right), & if \left(1 - \left[\frac{2\alpha n T_s}{\tau}\right]^2\right) = 0 \\ \frac{cos\left(\frac{\alpha\pi n T_s}{\tau}\right)}{1 - \left(\frac{2\alpha n T_s}{\tau}\right)^2}, & if\ n = 0 \\ \frac{sinc\left(\frac{nT_s}{\tau}\right) cos\left(\frac{\alpha\pi n T_s}{\tau}\right)}{1 - \left(\frac{2\alpha n T_s}{\tau}\right)^2}, & otherwise \end{cases}$$

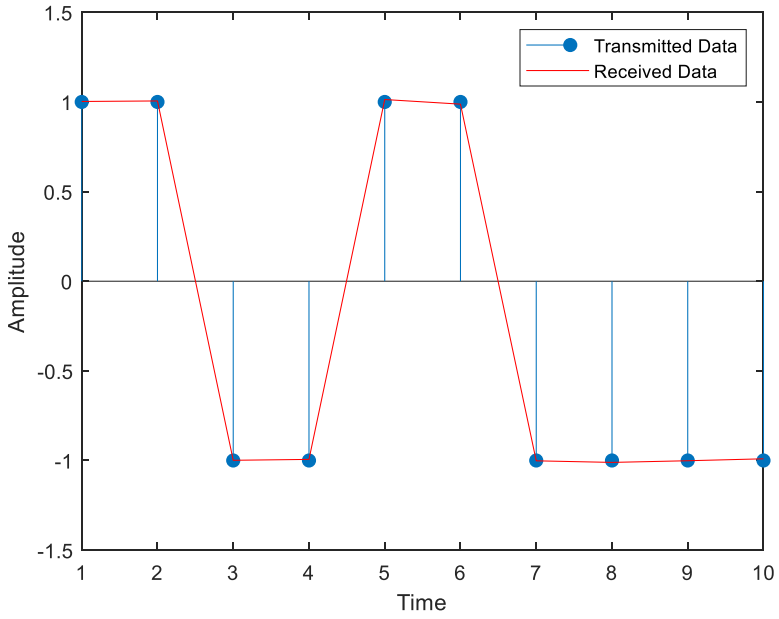
Program:

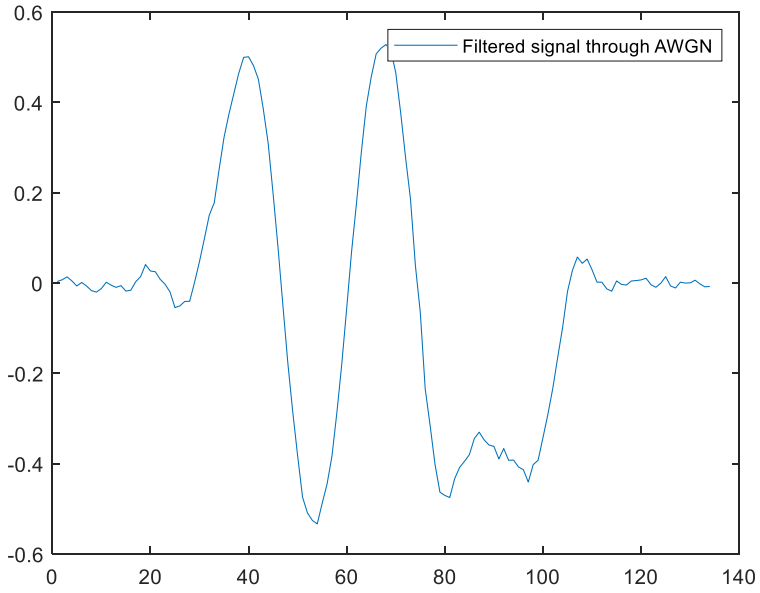
```
% Pulse Shaping and Matched Filtering
close all;
clear;
rolloff = 0.4;      % Roll off factor
span = 10;          % Filter span in symbols
sps = 7;            % Samples per symbol
M = 4;              % Modulation order
k = log2(M);        % Bits per symbol

% Create a vector of bipolar    data.
BP_Data = 2*randi([0 1],10, 1) - 1;

% Generate the square-root, raised cosine filter coefficients.
rctFilt = rcosdesign(rolloff, span, sps,'sqrt');
fvtool(rctFilt,'impulse')

% Upsample and filter the data for pulse shaping.
UP_s = upfirdn(BP_Data, rctFilt, sps,1);

% Using the number of bits per symbol (k)
% and the number of samples per symbol (sps),
% convert the ratio of energy per bit to noise power spectral
density (EbNo)
% to an SNR value for use by the awgn function.
EbNo = 100;
snr = EbNo + 10*log10(k) - 10*log10(sps);
filtlen = 10;       % Filter length in symbols

rxSignal = awgn(UP_s,snr,'measured');% filtering the signal through
an AWGN channel.

% Add noise.
rxSignal = rxSignal + randn(size(rxSignal))*.01;

rxFiltSignal = upfirdn(rxSignal,rctFilt,1,sps);        % Downsample
and filter
rxFiltSignal = rxFiltSignal(filtlen + 1:end - filtlen); % Account
for delay

% Plotting the function
```

```
figure;
stem(BP_Data,'filled')
hold on
plot(rxFiltSignal,'r')
xlabel('Time'); ylabel('Amplitude');
legend('Transmitted Data','Received Data')
figure;
plot(rxSignal)
legend('Filtered signal through AWGN')
```

Output







Result

Studied and verified pulse shaping and matched filtering