Project 1 – Matched Filtering

ECE538

John Claus

11/09/2019

Table of Contents

Problem Statement

Part 1

Model a RADAR system given the below parameters that can transmit both an unmodulated pulse and a signal using Barker-13 pulse compression. Employ a matched filter for receiver processing that processes the received signal using the transmitted pulse and matched filtering. Processing is required both for a received signal with and without noise.

RADAR Properties	
Pt	*10 dBW*
Fc	*1 GHz*
RCS	*30 bBsm*
PW	*1 micros*
Gt	*20 dB*
Gr	*20 dB*
Lsys	*0 dB*
Latm	*0 dB*

Create plots of both the unmodulated and Barker-13 received signals and their matched filter outputs, with and without noise.

Part 2

Using the algorithm developed in Part 1, create a matched filter output of a received signal, with noise, both with a signal and without. Collect these values of 100k iterations and use the resulting data to develop a probability density graph for Pd, the probability of correct target identification, and Pfa, the probability of false target identification. These probabilities being calculated through increasing threshold levels of the by iteratively multiplying various THR values, 1 to 100 in increments of 5, with the calculated noise power level.

Model Description

Part 1

The initial values were defined as follows, where a sampling frequency of 4x that of the center frequency was chosen to mitigate aliasing, provide decent fidelity, and use minimal memory.

```
% Initial values
G_db = 20;       %db
RCS_db = 30;     %dbsm
Noise_db = 3;    %db
Lsys_db = 0;     %db
Latm_db = 0;     %db
Pt_db = 10;      %dbW
R = 10000;       %m
c = physconst('LightSpeed');     %mps
Fc = 1e9;        %Hz
Fs = 4*Fc;       %Hz
PW = 1e-6;       %m
```

Next these initial values were converted from dB and more constants were calculated from the initial values for use in the received power, Pr, equation and plotting of the signals. The LOC variable was created as a placeholder on the received signal of where the pulse begins and the expected zero order return on the matched filter.

```
% Convert inital values from db
G = 10^(G_db/10);
RCS = 10^(RCS_db/10);
Lsys = 10^(Lsys_db/10);
Latm = 10^(Latm_db/10);
Pt = 10^(Pt_db/10);

% Calculate constants
lambda = c / Fc;
T = 4 * PW;
Ts = 1/Fs;
N = round(T*Fs);
t = (0:(N-1)).*Ts;
LOC = Ts*(round(N/2));
```

The Barker- 13 Chirp constants were calculated by dividing the pulse width with the number of chirps, 13, and applying that value to the sampling variable, N.

```
% Calculate Barker Code-13 constants
% +++++--++-+-+
Chirp = PW / 13;
Chirp_L = Chirp / Ts;
Chirp_N = round(Chirp / Ts);
```

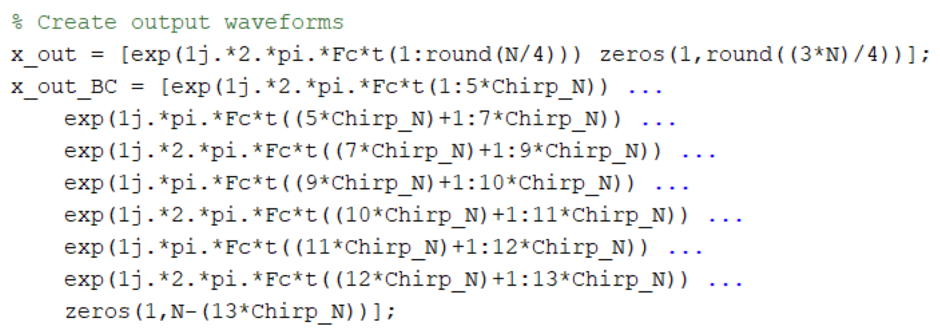
The received power constant was then calculated using the above constants and the equation

$$P_r = {P_t G_r G_t \lambda^2 RCS}\Big/{4\pi^3 R^4 L_{sys} L_{atm}}$$

Additionally, the receiver noise was scaled by the calculated received power value to allow for a correctly scaled sigma squared value in the injected noise equations later in the project.

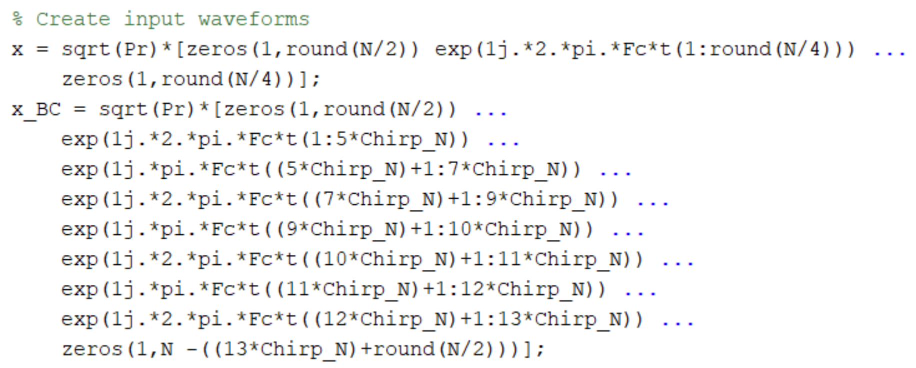
```
% Calculate Received Power
Pr = (Pt*(G^2)*(lambda^2)*RCS)/(((4*pi)^3)*(R^4)*Lsys*Latm);
sigma_sq = Pr*(10^(Noise_db/10));
```

The output waveforms for the both the 1 µs unmodulated pulse and the 1 µs Barker-13 pulse were then created within the first quarter portion of the 4 µs total signal. The last 3 µs of the signal were set to be zero. This output waveform was created to be used solely for the matched filter processing. The Barker-13 pulse implemented alternating 180 degree phase shifts as per the pattern defined for that specific coding design. This was done by stepping the signal creation in parts by the chirp N lengths for the phase required, The phasing shift in the signal was accomplished by either using 2π or π in the signal exponential depending on the phase shift required.



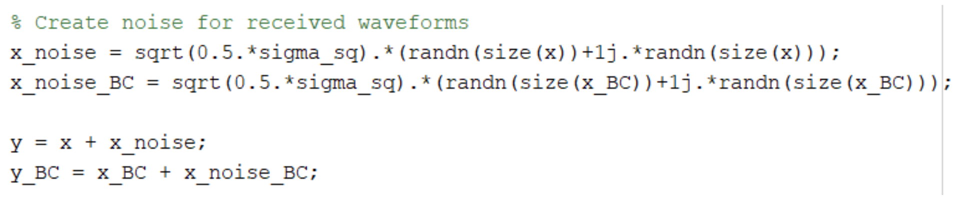
```
% Create output waveforms
x_out = [exp(1j.*2.*pi.*Fc*t(1:round(N/4))) zeros(1,round((3*N)/4))];
x_out_BC = [exp(1j.*2.*pi.*Fc*t(1:5*Chirp_N)) ...
    exp(1j.*pi.*Fc*t((5*Chirp_N)+1:7*Chirp_N)) ...
    exp(1j.*2.*pi.*Fc*t((7*Chirp_N)+1:9*Chirp_N)) ...
    exp(1j.*pi.*Fc*t((9*Chirp_N)+1:10*Chirp_N)) ...
    exp(1j.*2.*pi.*Fc*t((10*Chirp_N)+1:11*Chirp_N)) ...
    exp(1j.*pi.*Fc*t((11*Chirp_N)+1:12*Chirp_N)) ...
    exp(1j.*2.*pi.*Fc*t((12*Chirp_N)+1:13*Chirp_N)) ...
    zeros(1,N-(13*Chirp_N))];
```

Similar to how the output waveforms were created, the received signal or input waveform was created in parts. The only difference between these two signals are the location of the pulses in the signal. A location at the halfway point was chosen strictly for ease of processing and graphical implementation. This time distance was not chosen to be reflective the time required for the return pulse from the target 10km away. Additionally, the scaling of the received power calculation was applied to the equations to reflect an accurate representation of signal strength at the receiver. This was later significantly important with the addition of the noise on this signal.



```
% Create input waveforms
x = sqrt(Pr)*[zeros(1,round(N/2)) exp(1j.*2.*pi.*Fc*t(1:round(N/4))) ...
    zeros(1,round(N/4))];
x_BC = sqrt(Pr)*[zeros(1,round(N/2)) ...
    exp(1j.*2.*pi.*Fc*t(1:5*Chirp_N)) ...
    exp(1j.*pi.*Fc*t((5*Chirp_N)+1:7*Chirp_N)) ...
    exp(1j.*2.*pi.*Fc*t((7*Chirp_N)+1:9*Chirp_N)) ...
    exp(1j.*pi.*Fc*t((9*Chirp_N)+1:10*Chirp_N)) ...
    exp(1j.*2.*pi.*Fc*t((10*Chirp_N)+1:11*Chirp_N)) ...
    exp(1j.*pi.*Fc*t((11*Chirp_N)+1:12*Chirp_N)) ...
    exp(1j.*2.*pi.*Fc*t((12*Chirp_N)+1:13*Chirp_N)) ...
    zeros(1,N -((13*Chirp_N)+round(N/2)))];
```

The noise signal was created separate for each signal type. This was not required but was done for consistency in the design of the model to allow complete independence for troubleshooting purposes between the two unmodulated and Barker-13 signals. A third signal y was also created for both the unmodulated and Barker-13 signals added to their respective noise signals.



```
% Create noise for received waveforms
x_noise = sqrt(0.5.*sigma_sq).*(randn(size(x))+1j.*randn(size(x)));
x_noise_BC = sqrt(0.5.*sigma_sq).*(randn(size(x_BC))+1j.*randn(size(x_BC)));

y = x + x_noise;
y_BC = x_BC + x_noise_BC;
```

Finally, the matched filter was implemented by using the x correlation MATLAB function. This was done for scenarios with and without noise on both the unmodulated and Barker-13 signals in addition to a signal just comprised of noise for part 2 of the project.

```
% Matched Filtering - No modulation
xcorOut = xcorr(x_out,y);
numberOfLags = length(xcorOut);
lagVector = ((1:(2*N-1)));
xcorTimeVec = lagVector.*Ts;
```

The outputs of these function were then plot and their figures can be found in the Results section of this paper.

Part 2

The equations used for the unmodulated signal were used in the calculation of the probability density functions due to there lower reliability and SNR than that of the Barker-13 code. These equations were iteratively run 100,000x and the value at the expected pulse location were taken for each and added to a list of values for later processing.

```
% Creates List of Values at Expected 0th Matched Filter Output
for iter = 1:max_iter
    x = sqrt(Pr)*[zeros(1,round(N/2)) exp(1j.*2.*pi.*Fc*t(1:round(N/4))) ...
        zeros(1,round(N/4))];
    x_noise = sqrt(0.5.*sigma_sq).*(randn(size(x))+1j.*randn(size(x)));
    y = x + x_noise;
    xcorOut = xcorr(x_out,y);
    xcorOut_NO = xcorr(x_out,x_noise);
    index = round(length(xcorOut)/4);
    noise_op = abs(xcorOut_NO(index));
    signal_op = abs(xcorOut(index));
    match_sig_op(1,iter) = signal_op;
    match_noise_op(1,iter) = noise_op;
end
```

These lists were then analyzed by first squaring the variance of the real portion of the list of noise only values collected from the matched filter output. Next varying levels of the threshold multiplier scaled this noise power level to create a threshold of detection. If a detection occurred from noise without signal then it was processed for the Pfa calculation and a detection occurred when there was a pulse present, then it was processed for the Pd calculation.

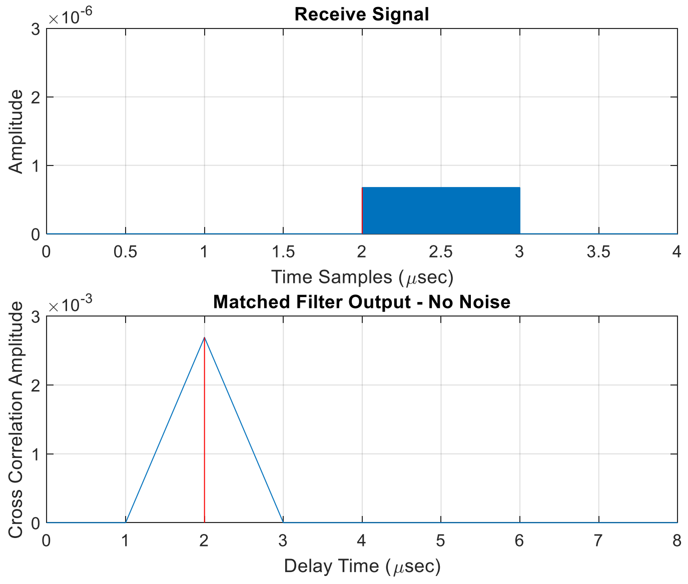
```
% Calculates Pd and Pfa from the Matched Filter Output Data
iter2 = 1;
for THR = 1:5:100
    noisePwr = var(real(match_noise_op)).*2;
    threshold = noisePwr*THR;
    Pfa(iter2) = sum(abs(match_noise_op).^2>threshold)/max_iter;
    Pd(iter2)  = sum(abs(match_sig_op).^2>threshold)/max_iter;
    thr_graph(iter2) = THR;
    iter2 = iter2 + 1;
end
```

A graphical representation of Pfa and Pd given varying THR was then created and can seen in the Results section of this paper.

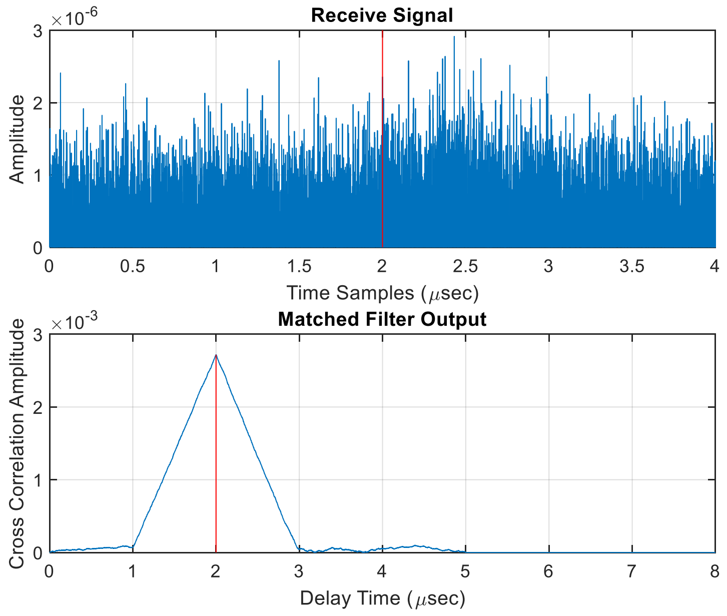
Results

Part 1

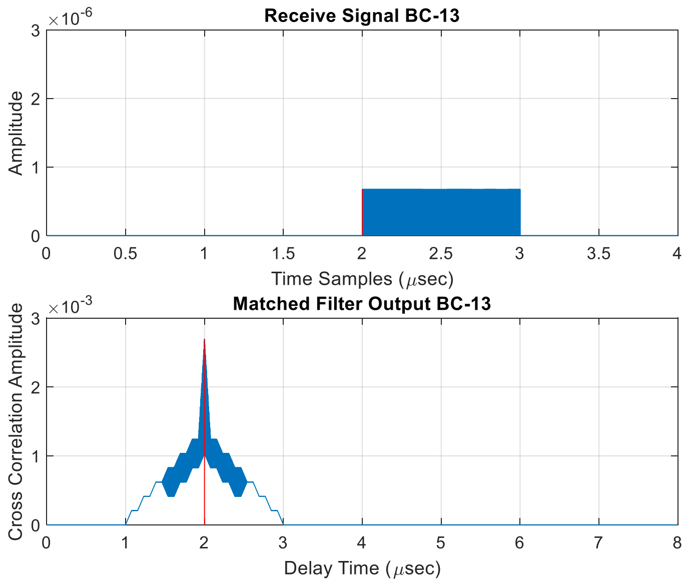
The plot seen in the following figure displays the both the received signal and matched filter output for the unmodulated signal without noise. The matched filter output aligns with the received pulse timing thereby verifying proper operation.



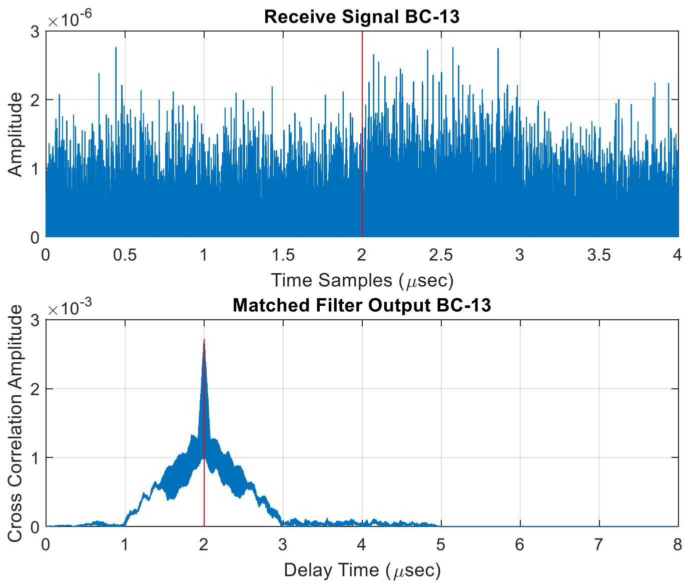
The next plot incorporates noise with the unmodulated signal at 3dB. The received signal clearly shows this additive noise's effect on the appearance of the signal. The matched filter output, however, is nearly identical to that of the noiseless signal. This filter clearly works extremely well in filtering out this additive noise as its peak is still correctly center at the pulse's time delay.



The plot in the below figure is that of the Barker-13 signal and matched filter output without noise. The received signal looks nearly identical; however the matched filter output is significantly different due to the phase shifting in the pulse. The center lobe is considerable higher and narrower. It is still correctly aligned with the timing of the received pulse.

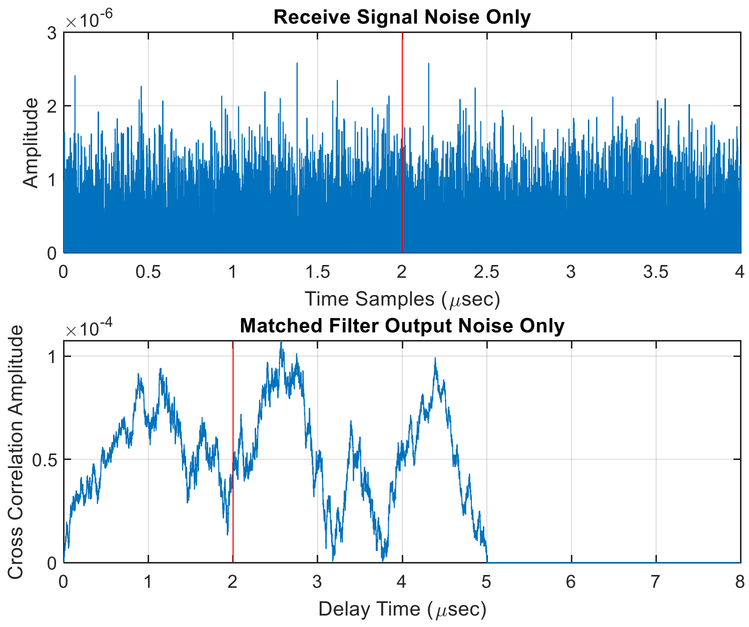


Again, the received Barker-13 signal appears at first look to be identical to the received signal for the unmodulated signal. However, the matched filter output has a much higher SNR and still aligns with the pulse location on the received signal despite the high noise in the signal.

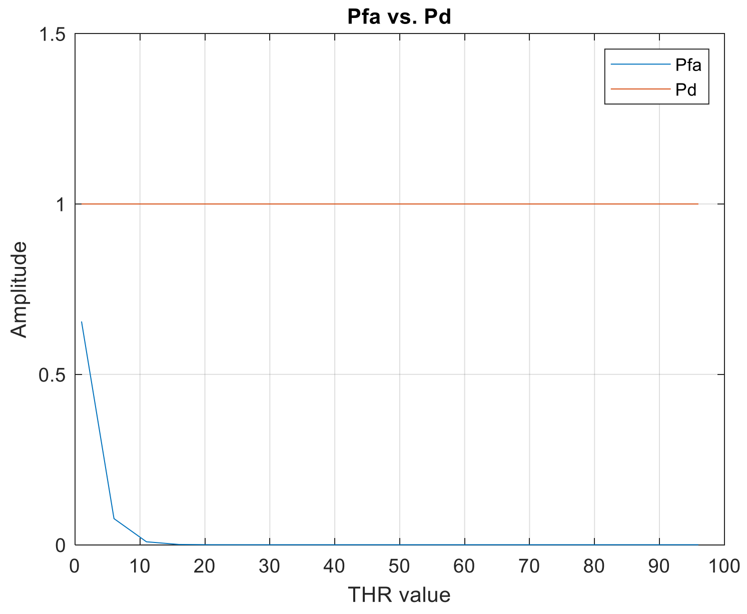


Part 2

The two signals used for this section of the project were the unmodulated signal with noise and noise alone. A graphical representation of the unmodulated signal can be seen in part 1 of this section. A graphical representation of both the signal and its matched filter output can be seen below.

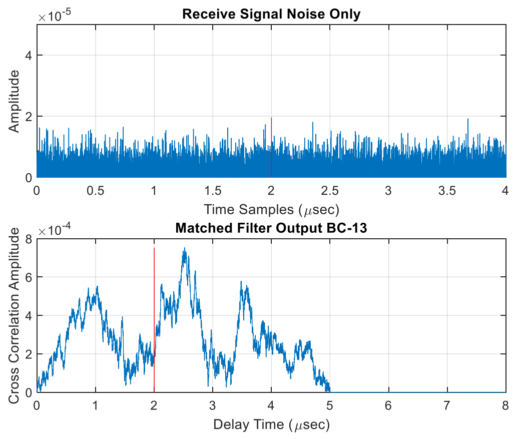


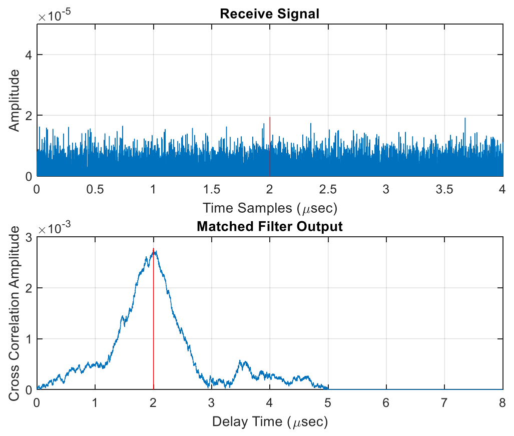
The iterative comparison of both signals measured at the expected zero order matched filter output shows that the Pd is at 100% and Pfa varies initially but rapidly approaches 0%. The results here show that matched filtering has a significant effect on finding a pulse buried within a noisy environment.



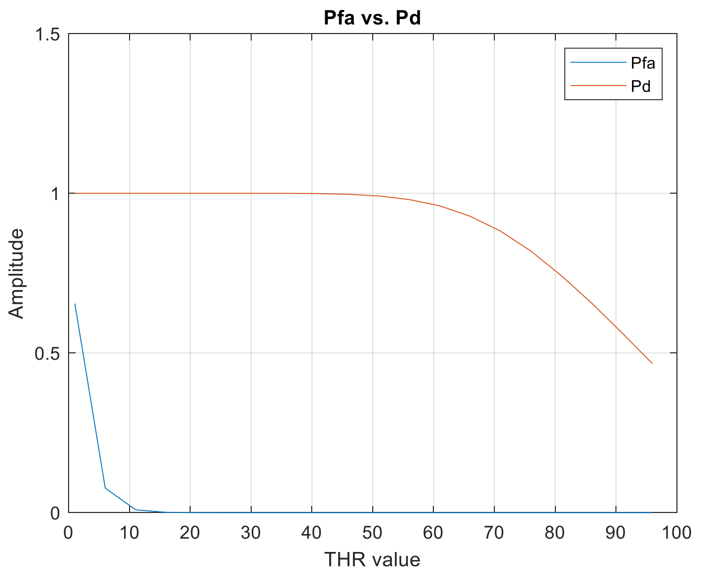
Discussion

Although these results were ideal, I concluded they were too good to be true. A more realistic model would not have a Pd of 100% despite the thresholding multiplier. Perhaps the received power was incorrectly calculated, or the RCS was so massive that the matched filtering would work regardless. Therefore, out of curiosity, I decided to push the limits of the filter by significantly increasing the system noise to 20dB. As can be seen in the figures below, the received signal in both the noise only and unmodulated signal with noise are nearly identical. However, their matched filter output is dramatically different. Detection still occurred at the expected location.





Additionally, the probaility graphs have a much clearer response to the thresholding than the earlier model had.



In conclusion, this project was enlightening for showing the power of matched filtering in signal processing. Additionally, the incorporation of the processing technique with coded modulation as with the Barker – 13 method shows its value even greater. If a target’s velocity is not of concern this could be a great way to increase resolution and compensate for a noisy system.

Appendix – MATLAB Code

```
%%%% John Claus %%%%
%%%% ECE538     %%%%
%%%% Project 1  %%%%
%%%% 11/08/2019 %%%%

clear all
close all

%%% Problem 1 %%%
% Initial values
G_db = 20;       %db
RCS_db = 30;     %dbsm
Noise_db = 3;    %db
Lsys_db = 0;     %db
Latm_db = 0;     %db
Pt_db = 10;      %dbW
R = 10000;       %m
c = physconst('LightSpeed');    %mps
Fc = 1e9;        %Hz
Fs = 4*Fc;       %Hz
PW = 1e-6;       %m

% Convert inital values from db
G = 10^(G_db/10);
RCS = 10^(RCS_db/10);
Lsys = 10^(Lsys_db/10);
Latm = 10^(Latm_db/10);
Pt = 10^(Pt_db/10);

% Calculate constants
lambda = c / Fc;
T = 4 * PW;
Ts = 1/Fs;
N = round(T*Fs);
t = (0:(N-1)).*Ts;
LOC = Ts*(round(N/2));

% Calculate Barker Code-13 constants
% +++++--++-+-+
Chirp = PW / 13;
Chirp_L = Chirp / Ts;
Chirp_N = round(Chirp / Ts);

% Calculate Received Power
Pr = (Pt*(G^2)*(lambda^2)*RCS)/(((4*pi)^3)*(R^4)*Lsys*Latm);
sigma_sq = Pr*(10^(Noise_db/10));

% Create output waveforms
x_out = [exp(1j.*2.*pi.*Fc*t(1:round(N/4))) zeros(1,round((3*N)/4))];
x_out_BC = [exp(1j.*2.*pi.*Fc*t(1:5*Chirp_N)) ...
    exp(1j.*pi.*Fc*t((5*Chirp_N)+1:7*Chirp_N)) ...
    exp(1j.*2.*pi.*Fc*t((7*Chirp_N)+1:9*Chirp_N)) ...
    exp(1j.*pi.*Fc*t((9*Chirp_N)+1:10*Chirp_N)) ...
    exp(1j.*2.*pi.*Fc*t((10*Chirp_N)+1:11*Chirp_N)) ...
    exp(1j.*pi.*Fc*t((11*Chirp_N)+1:12*Chirp_N)) ...
    exp(1j.*2.*pi.*Fc*t((12*Chirp_N)+1:13*Chirp_N)) ...
    zeros(1,N-(13*Chirp_N))];

% Create input waveforms
x = sqrt(Pr)*[zeros(1,round(N/2)) exp(1j.*2.*pi.*Fc*t(1:round(N/4))) ...
    zeros(1,round(N/4))];
x_BC = sqrt(Pr)*[zeros(1,round(N/2)) ...
    exp(1j.*2.*pi.*Fc*t(1:5*Chirp_N)) ...
    exp(1j.*pi.*Fc*t((5*Chirp_N)+1:7*Chirp_N)) ...
    exp(1j.*2.*pi.*Fc*t((7*Chirp_N)+1:9*Chirp_N)) ...
    exp(1j.*pi.*Fc*t((9*Chirp_N)+1:10*Chirp_N)) ...
    exp(1j.*2.*pi.*Fc*t((10*Chirp_N)+1:11*Chirp_N)) ...
    exp(1j.*pi.*Fc*t((11*Chirp_N)+1:12*Chirp_N)) ...
    exp(1j.*2.*pi.*Fc*t((12*Chirp_N)+1:13*Chirp_N)) ...
    zeros(1,N -((13*Chirp_N)+round(N/2)))];

% Create noise for received waveforms
x_noise = sqrt(0.5.*sigma_sq).*(randn(size(x))+1j.*randn(size(x)));
x_noise_BC = sqrt(0.5.*sigma_sq).*(randn(size(x_BC))+1j.*randn(size(x_BC)));

y = x + x_noise;
y_BC = x_BC + x_noise_BC;

% Matched Filtering - No modulation
xcorOut = xcorr(x_out,y);
numberOfLags = length(xcorOut);
lagVector = ((1:(2*N-1)));
xcorTimeVec = lagVector.*Ts;

% Matched Filtering - No modulation - No Noise
xcorOut_NN = xcorr(x_out,x);
numberOfLags_NN = length(xcorOut_NN);
lagVector_NN = ((1:(2*N-1)));
xcorTimeVec_NN = lagVector_NN.*Ts;

% Matched Filtering - Barker Code 13
xcorOut_BC = xcorr(x_out_BC,y_BC);
numberOfLags_BC = length(xcorOut_BC);
lagVector_BC = ((1:(2*N-1)));
xcorTimeVec_BC = lagVector_BC.*Ts;

% Matched Filtering - Barker Code 13 - No Noise
xcorOut_BC_NN = xcorr(x_out_BC,x_BC);
numberOfLags_BC_NN = length(xcorOut_BC_NN);
lagVector_BC_NN = ((1:(2*N-1)));
xcorTimeVec_BC_NN = lagVector_BC_NN.*Ts;

% Matched Filtering - Noise Only
xcorOut_NO = xcorr(x_out,x_noise);
numberOfLags_NO = length(xcorOut_NO);
lagVector_NO = ((1:(2*N-1)));
xcorTimeVec_NO = lagVector_NO.*Ts;

% Plot Unmodulated Signal and Matched Filter Output
figure
subplot(2,1,1)
plot(t./1e-6,y);
ylim([0 3e-6])
ylabel('Amplitude')
xlabel('Time Samples (\musec)')
grid on;title('Receive Signal')
hold on
plot([LOC, LOC]./1e-6, [0 max(abs(y))],'r')
subplot(2,1,2)
plot(xcorTimeVec./1e-6,abs(xcorOut))
ylabel('Cross Correlation Amplitude')
xlabel('Delay Time (\musec)')
grid on;
title('Matched Filter Output')
hold on;
plot([LOC LOC]./1e-6,[0 max(abs(xcorOut))],'r')
hold off;

% Plot Unmodulated Signal and Matched Filter Output - No Noise
figure
subplot(2,1,1)
plot(t./1e-6,x);
ylim([0 3e-6])
ylabel('Amplitude')
xlabel('Time Samples (\musec)')
grid on;title('Receive Signal')
hold on
plot([LOC, LOC]./1e-6, [0 max(abs(x))],'r')
subplot(2,1,2)
plot(xcorTimeVec_NN./1e-6,abs(xcorOut_NN))
```

```matlab
ylabel('Cross Correlation Amplitude')
xlabel('Delay Time (\musec)')
grid on;
title('Matched Filter Output - No Noise')
hold on;
plot([LOC LOC]./1e-6,[0 max(abs(xcorOut_NN))],'r')
hold off;

% Plot Barker Code - 13 Signal and Matched Filter Output
figure
subplot(2,1,1)
plot(t./1e-6,y_BC);
ylim([0 3e-6])
ylabel('Amplitude')
xlabel('Time Samples (\musec)')
grid on;title('Receive Signal BC-13')
hold on
plot([LOC, LOC]./1e-6, [0 max(abs(y_BC))],'r')
subplot(2,1,2)
plot(xcorTimeVec_BC./1e-6,abs(xcorOut_BC))
ylabel('Cross Correlation Amplitude')
xlabel('Delay Time (\musec)')
grid on;
title('Matched Filter Output BC-13')
hold on;
plot([LOC LOC]./1e-6,[0 max(abs(xcorOut_BC))],'r')
hold off;

% Plot Barker Code - 13 Signal and Matched Filter Output - No Noise
figure
subplot(2,1,1)
plot(t./1e-6,x_BC);
ylim([0 3e-6])
ylabel('Amplitude')
xlabel('Time Samples (\musec)')
grid on;title('Receive Signal BC-13')
hold on
plot([LOC, LOC]./1e-6, [0 max(abs(x_BC))],'r')
subplot(2,1,2)
plot(xcorTimeVec_BC_NN./1e-6,abs(xcorOut_BC_NN))
ylabel('Cross Correlation Amplitude')
xlabel('Delay Time (\musec)')
grid on;
title('Matched Filter Output BC-13')
hold on;
plot([LOC LOC]./1e-6,[0 max(abs(xcorOut_BC_NN))],'r')
hold off;

% Plot Noise Only and Matched Filter Output
figure
subplot(2,1,1)
plot(t./1e-6,x_noise);
ylim([0 3e-6])
ylabel('Amplitude')
xlabel('Time Samples (\musec)')
grid on;title('Receive Signal Noise Only')
hold on
plot([LOC, LOC]./1e-6, [0 max(abs(x_noise))],'r')
subplot(2,1,2)
plot(xcorTimeVec_NO./1e-6,abs(xcorOut_NO))
ylabel('Cross Correlation Amplitude')
xlabel('Delay Time (\musec)')
grid on;
title('Matched Filter Output Noise Only')
hold on;
plot([LOC LOC]./1e-6,[0 max(abs(xcorOut_NO))],'r')
hold off;

%%% Problem 2 %%%
% Define Initial Variables and Zero Lists
max_iter = 100000;
```

```
match_sig_op = zeros(1,max_iter);
match_noise_op = zeros(1,max_iter);
Pfa = zeros(1,20);
Pd = zeros(1,20);
thr_graph = zeros(1,20);

% Creates List of Values at Expected 0th Matched Filter Output
for iter = 1:max_iter
    x = sqrt(Pr)*[zeros(1,round(N/2)) exp(1j.*2.*pi.*Fc*t(1:round(N/4))) ...
        zeros(1,round(N/4))];
    x_noise = sqrt(0.5.*sigma_sq).*(randn(size(x))+1j.*randn(size(x)));
    y = x + x_noise;
    xcorOut = xcorr(x_out,y);
    xcorOut_NO = xcorr(x_out,x_noise);
    index = round(length(xcorOut)/4);
    noise_op = abs(xcorOut_NO(index));
    signal_op = abs(xcorOut(index));
    match_sig_op(1,iter) = signal_op;
    match_noise_op(1,iter) = noise_op;
end

% Calculates Pd and Pfa from the Matched Filter Output Data
iter2 = 1;
for THR = 1:5:100
    noisePwr = var(real(match_noise_op)).*2;
    threshold = noisePwr*THR;
    Pfa(iter2) = sum(abs(match_noise_op).^2>threshold)/max_iter;
    Pd(iter2)  = sum(abs(match_sig_op).^2>threshold)/max_iter;
    thr_graph(iter2) = THR;
    iter2 = iter2 + 1;
end

% Plots Pd vs Pfa data
figure
plot(thr_graph,Pfa,thr_graph,Pd);
legend('Pfa','Pd')
ylim([0 1.5])
ylabel('Amplitude')
xlabel('THR value')
grid on;title('Pfa vs. Pd')
```