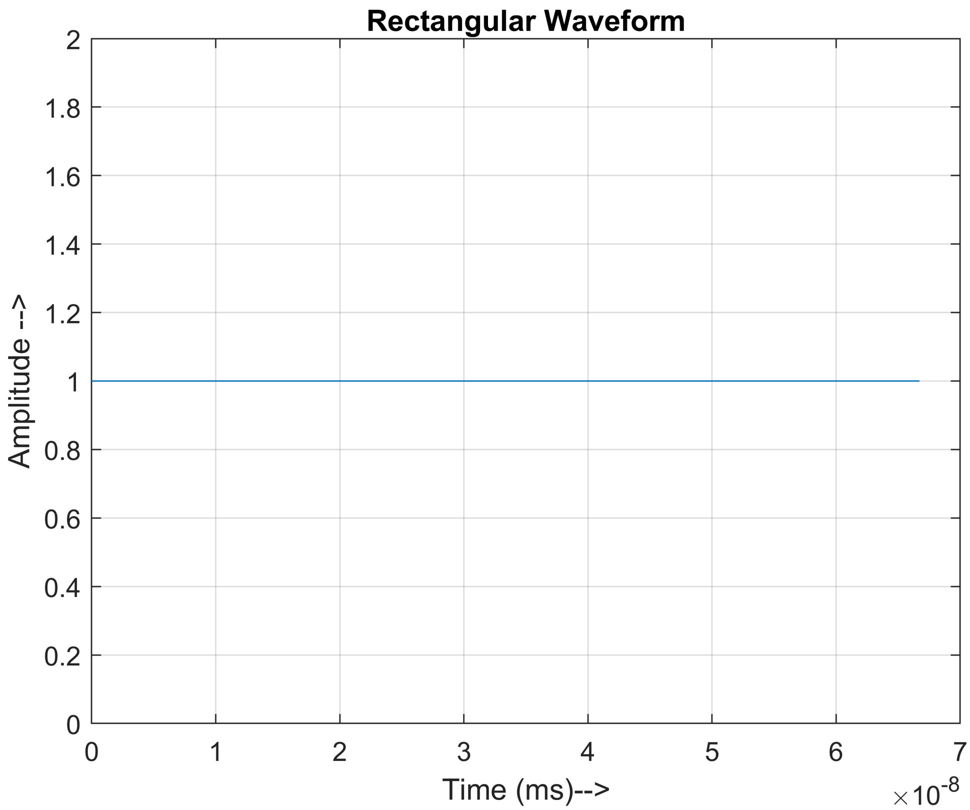
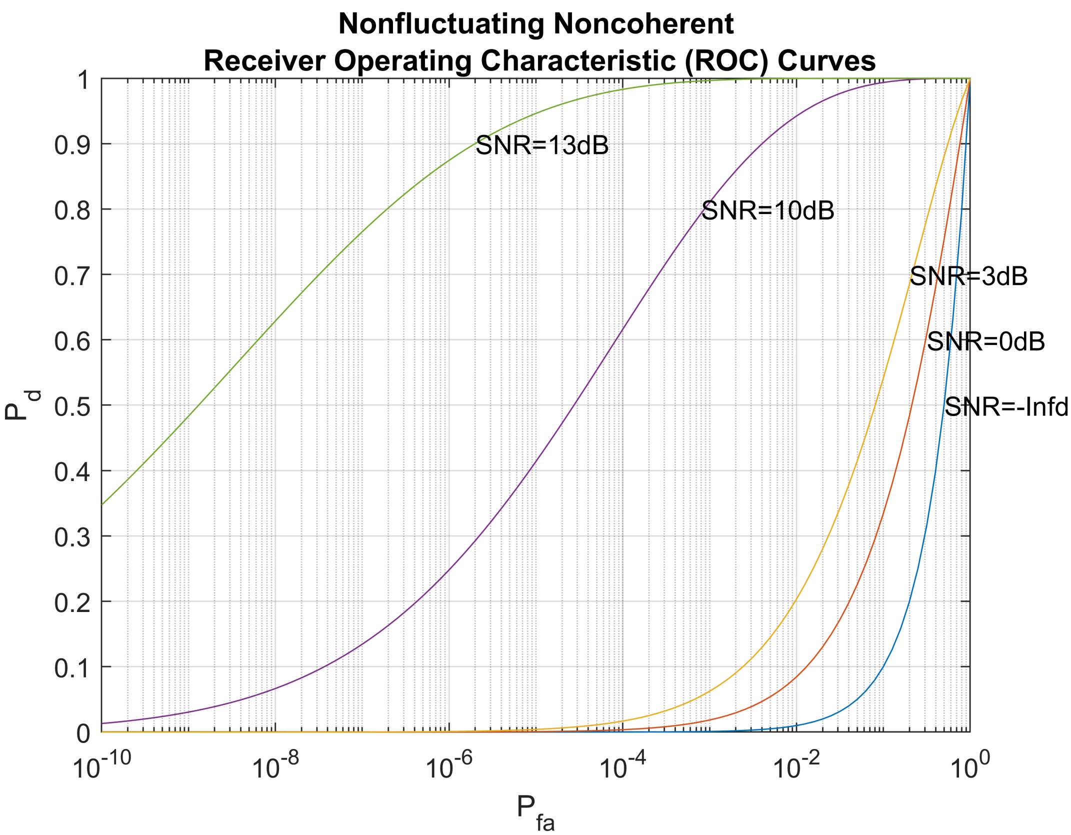
```
pd = 0.9;              % Probability of detection
pfa = 1e-6;            % Probability of false alarm
max_range = 5000;      % Maximum unambiguous range
range_res = 20;        % Required range resolution
tgt_rcs = 1;           % Required target radar cross section

import radarplot.*

% import peakfinder.*

%% Monostatic Radar System Design

prop_speed = physconst('LightSpeed');   % Propagation speed
pulse_bw = prop_speed/(2*range_res);    % Pulse bandwidth
pulse_width = 1/pulse_bw;               % Pulse width
prf = prop_speed/(2*max_range);         % Pulse repetition frequency
fs = 2*pulse_bw;                        % Sampling rate
waveform = phased.RectangularWaveform(...
    'PulseWidth',1/pulse_bw,...
    'PRF',prf,...
    'OutputFormat', "Pulses",...
    "NumPulses", 1,...
    'SampleRate',fs,...
    "PRFOutputPort", true);
```

```
% rectwav= step(waveform);
% nsamp = size(rectwav,1);
% t = [0:(nsamp-1)]/fs;
% plot(t*1000,real(rectwav))
%

plot(waveform);
title("Rectangular Waveform");
xlabel("Time (ms)-->");
ylabel("Amplitude -->");
```



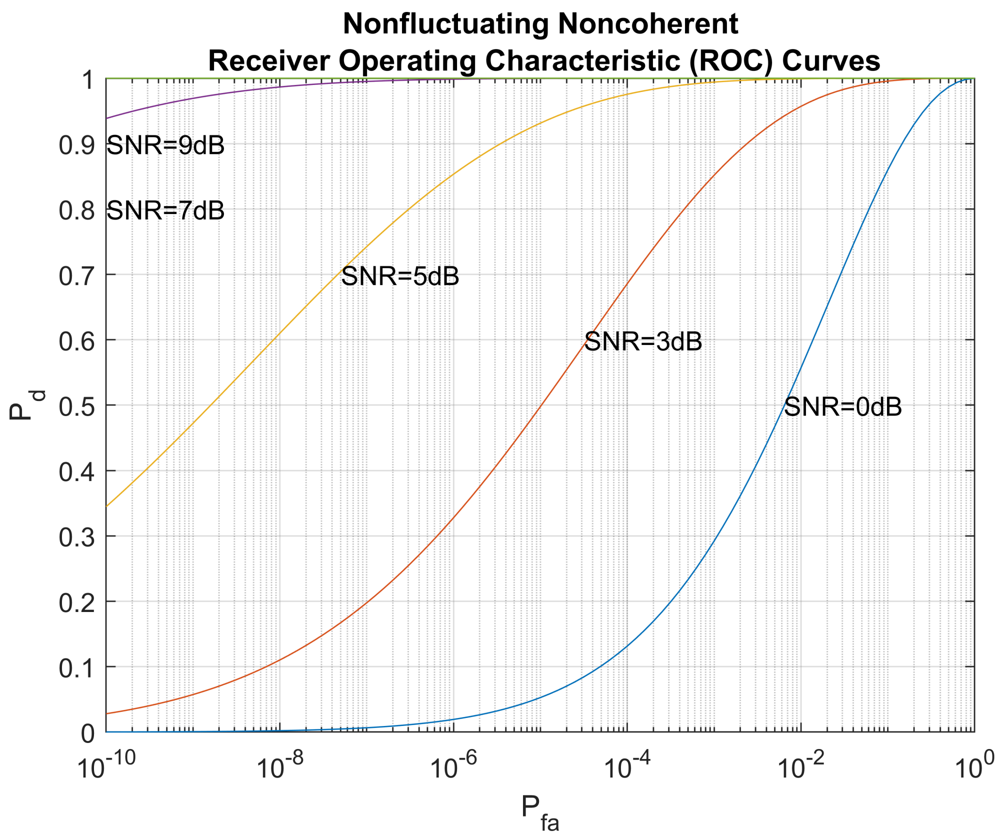

```
%Rx Noise Characteristics

noise_bw = pulse_bw;

receiver = phased.ReceiverPreamp(...
    'Gain',20,...
    'NoiseFigure',0,...
    'SampleRate',fs,...
    'EnableInputPort',true);
```

```
%Tx Design

snr_db = [-inf, 0, 3, 10, 13];
rocsnr(snr_db,'SignalType','NonfluctuatingNoncoherent');
```




```
%In the Graph, we see that the required SNR for Pfa= 1e-6 and pd= 0.9 must exceed 13db,
% which is very unrealistic
```

```
%Hence, we use Pulse Integration to reduce the SNR

% Using Coherent Integration, for better efficiency

num_pulse_int = 10;
rocsnr([0 3 5 7 9],'SignalType','NonfluctuatingNoncoherent',...
    'NumPulses',num_pulse_int);
```




```
snr_min = albersheim(pd, pfa, num_pulse_int)
```

```
snr_min = 4.9904
```

```
% Albershiem Equation Snr= A+ 0.12AB + 1.7B
% A= ln(0.62/Pfa) and B= ln(Pd/(1-Pd))
```

```
tx_gain = 20;

fc = 10e9;
lambda = prop_speed/fc;

peak_power = radareqpow(lambda,max_range,snr_min,pulse_width,...
    'RCS',tgt_rcs,'Gain',tx_gain)
```

```
peak_power = 1.3066e+04
```

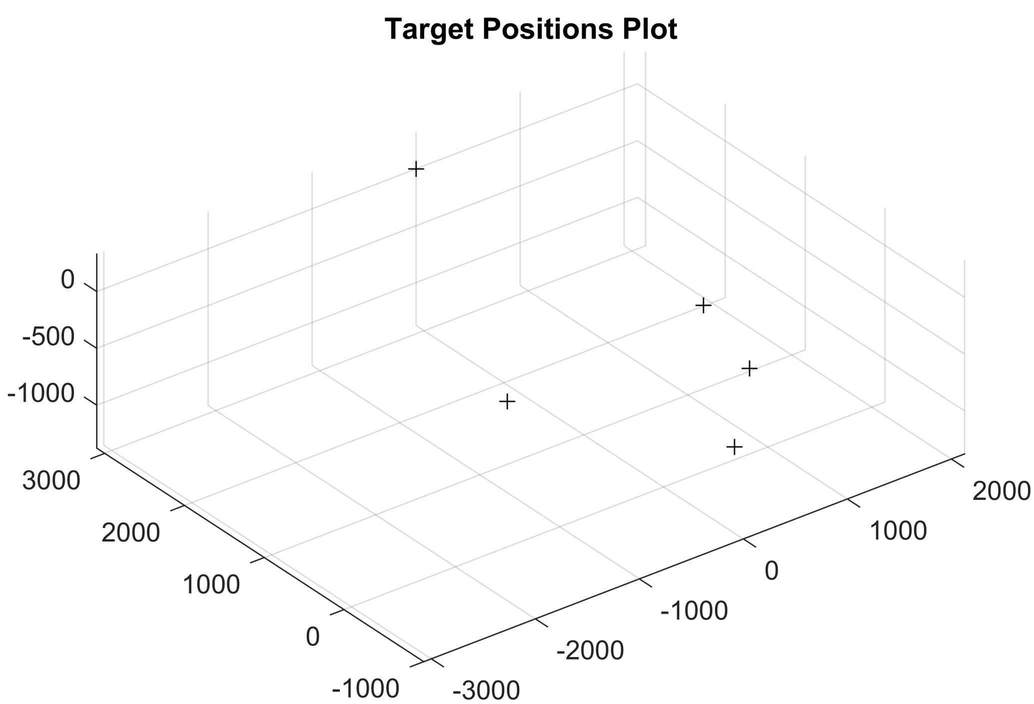
```
transmitter = phased.Transmitter(...
    'Gain',tx_gain,...
    'PeakPower',peak_power,...
    'InUseOutputPort',true);
```

```
% Radiator and Collector

% In a monostatic radar system, the radiator and the collector share the
% same antenna, so we will first define the antenna. To simplify the
% design, we choose an isotropic antenna.

% We assume that the antenna is stationary.

antenna = phased.IsotropicAntennaElement(...
    'FrequencyRange',[5e9 15e9]);

sensormotion = phased.Platform(...
    'InitialPosition',[0; 0; 0],...
    'Velocity',[0; 0; 0]);

radiator = phased.Radiator(...
    'Sensor',antenna,...
    'OperatingFrequency',fc);

collector = phased.Collector(...
    'Sensor',antenna,...
    'OperatingFrequency',fc);
```

```
%Simulation

%Test Targets

tgtpos = [[-1502;0;-100],[0;3100;0],[459;100;10], [559;-160;-1150], [59;-1000;100]];
tgtvel = [[0;0;0],  [0;0;0],    [0;0;0],        [0;0;0],        [0;0;0]];
tgtmotion = phased.Platform('InitialPosition',tgtpos,'Velocity',tgtvel);

tgtrcs = [1.6 2.2 10.05 5.5 0.99];
target = phased.RadarTarget('MeanRCS',tgtrcs,'OperatingFrequency',fc);

%Env Simulation

channel = phased.FreeSpace(...
    'SampleRate',fs,...
    'TwoWayPropagation',true,...
    'OperatingFrequency',fc);


fast_time_grid = unigrid(0,1/fs,1/prf,'[)');    % Time within the each pulse
slow_time_grid = (0:num_pulse_int-1)/prf;       %Time between tow successive pulses
```

```
receiver.SeedSource = 'Property';
receiver.Seed = 2009;
```

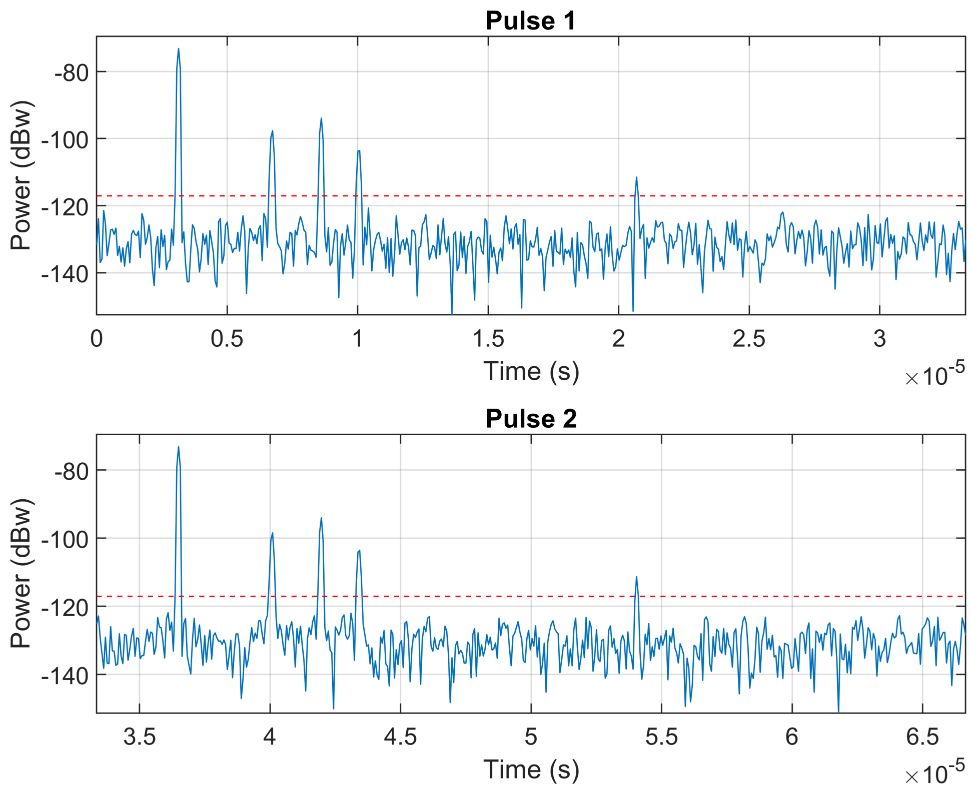
```
%% Range Plots

tgtcol= tgtpos';

txpos= tgtcol(:,1);
tyos= tgtcol(:, 2);
tzpos= tgtcol(:, 3);


Pl= alphaShape(txpos, tyos, tzpos);
plot(Pl, "FaceColor","none", "LineStyle","none", "Marker","+", "MarkerFaceColor","auto");
title("Target Positions Plot")
```




```
% Pre-allocating array for improved processing speeds
rxpulses = zeros(numel(fast_time_grid),num_pulse_int);

for m = 1:num_pulse_int

    % Update sensor and target positions
    [sensorpos,sensorvel] = sensormotion(1/prf);
```

```
    [tgtpos,tgtvel] = tgtmotion(1/prf);

    % Calculate the target angles as seen by the sensor
    [tgtrng,tgtang] = rangeangle(tgtpos,sensorpos);

    % Simulate propagation of pulse in direction of targets
    pulse = waveform();
    [txsig,txstatus] = transmitter(pulse);
    txsig = radiator(txsig,tgtang);
    txsig = channel(txsig,sensorpos,tgtpos,sensorvel,tgtvel);

    % Reflect pulse off of targets
    tgtsig = target(txsig);

    % Receive target returns at sensor
    rxsig = collector(tgtsig,tgtang);
    rxpulses(:,m) = receiver(rxsig,~(txstatus>0));
end


%% Range Detection

% The detector compares the signal power to a given threshold.

npower = noisepow(noise_bw,receiver.NoiseFigure,...
    receiver.ReferenceTemperature);
threshold = npower * db2pow(npwgnthresh(pfa,num_pulse_int,'noncoherent'));

%%
% Plotting the first two received pulses with the threshold
num_pulse_plot = 2;
helperRadarPulsePlot(rxpulses,threshold,...
    fast_time_grid,slow_time_grid,num_pulse_plot);



% The matched filter offers a processing gain which improves the detection threshold.


matchingcoeff = getMatchedFilter(waveform);
matchedfilter = phased.MatchedFilter(...
    'Coefficients',matchingcoeff,...
    'GainOutputPort',true);
[rxpulses, mfgain] = matchedfilter(rxpulses);

%%
% The matched filter introduces an intrinsic filter delay so that the locations of the peak are



matchingdelay = size(matchingcoeff,1)-1;
rxpulses = buffer(rxpulses(matchingdelay+1:end),size(rxpulses,1));

%%
```

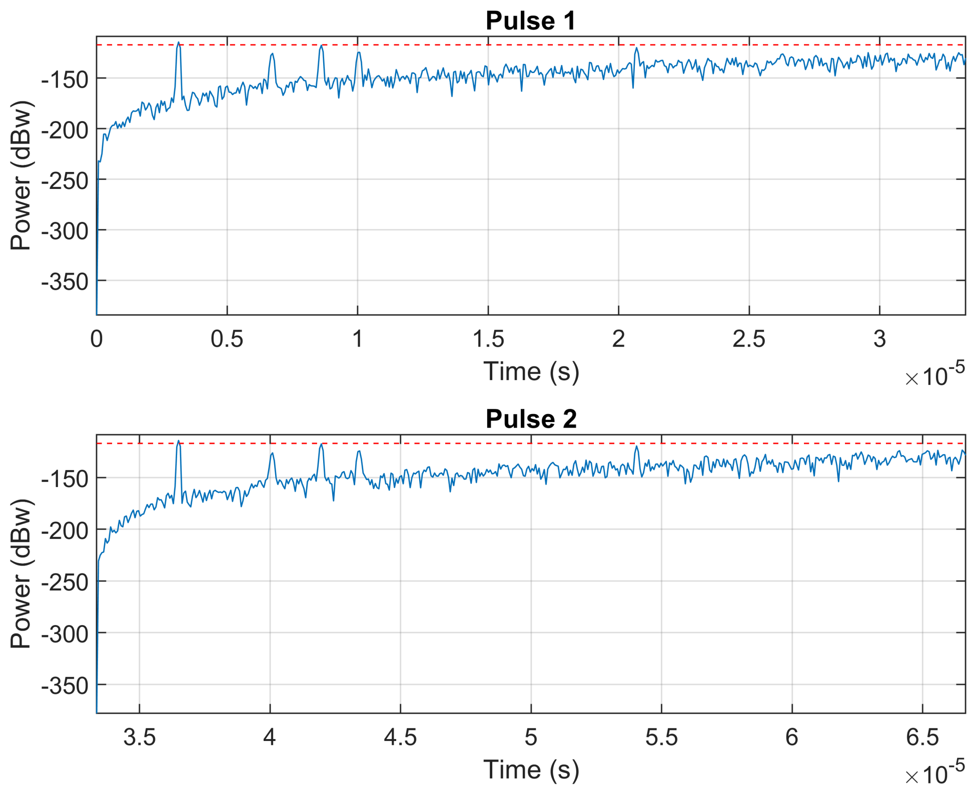
```
% The threshold is then increased by the matched filter processing gain.
threshold = threshold * db2pow(mfgain);

helperRadarPulsePlot(rxpulses,threshold,...
    fast_time_grid,slow_time_grid,num_pulse_plot);
```



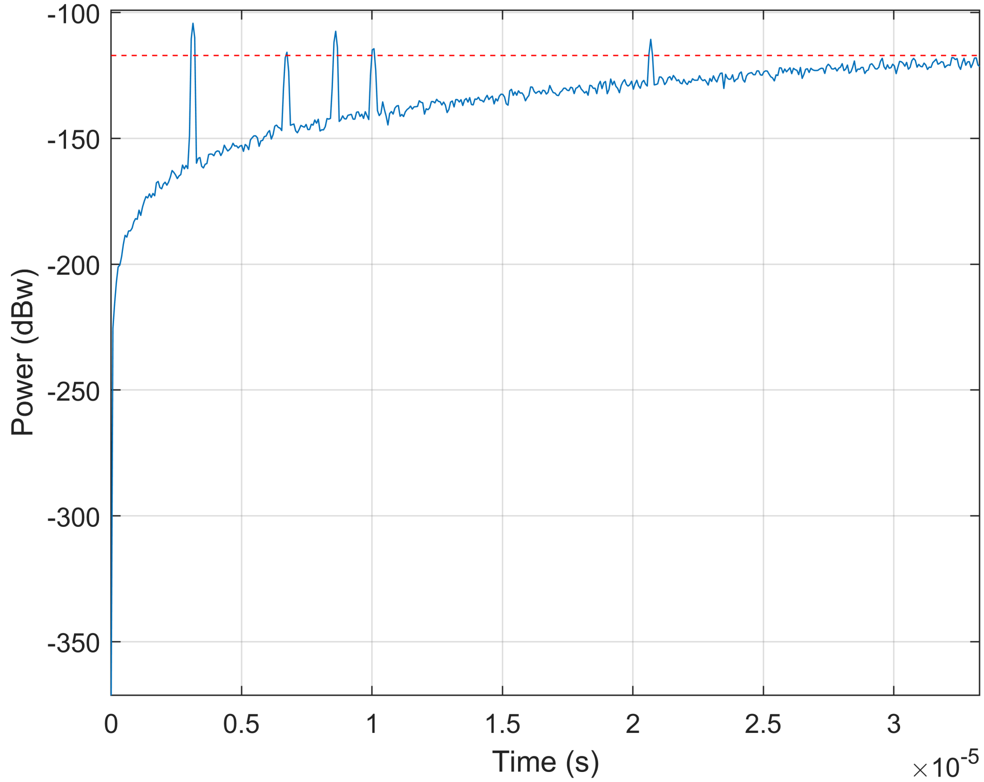

```
%%
% After the matched filter stage, the SNR is improved.  However, because
% the received signal power is dependent on the range, the return of a
% close target is still much stronger than the return of a target farther
% away. Therefore, as the above figure shows, the noise from a close range
% bin also has a significant chance of surpassing the threshold and
% shadowing a target farther away.  To ensure the threshold is fair to all
% the targets within the detectable range, we can use a time varying gain
% to compensate for the range dependent loss in the received echo.
%
% To compensate for the range dependent loss, we first calculate the range
% gates corresponding to each signal sample and then calculate the free
% space path loss corresponding to each range gate. Once that information
% is obtained, we apply a time varying gain to the received pulse so that
% the returns are as if from the same reference range (the maximum
% detectable range).

range_gates = prop_speed*fast_time_grid/2;

tvg = phased.TimeVaryingGain(...s
```

```
    'RangeLoss',2*fspl(range_gates,lambda),...
    'ReferenceLoss',2*fspl(max_range,lambda));

rxpulses = tvg(rxpulses);
```

```
%%
%  plotting the same pulses after the range normalization
helperRadarPulsePlot(rxpulses,threshold,...
    fast_time_grid,slow_time_grid,num_pulse_plot);
```




```
% The threshold is above the maximum power level
% contained in each pulse.  Therefore, nothing can be detected at this
% stage yet.


% We can further improve the SNR by noncoherently integrating the received pulses.

rxpulses = pulsint(rxpulses,'noncoherent');

helperRadarPulsePlot(rxpulses,threshold,...
    fast_time_grid,slow_time_grid,1);
```




```
% Range Detection

[~,range_detect] = findpeaks(rxpulses,'MinPeakHeight',sqrt(threshold));

%%
% The true ranges and the detected ranges of the targets are

true_range = round(tgtrng)
```

```
true_range = 1×5
        1505        3100         470        1289        1007
```

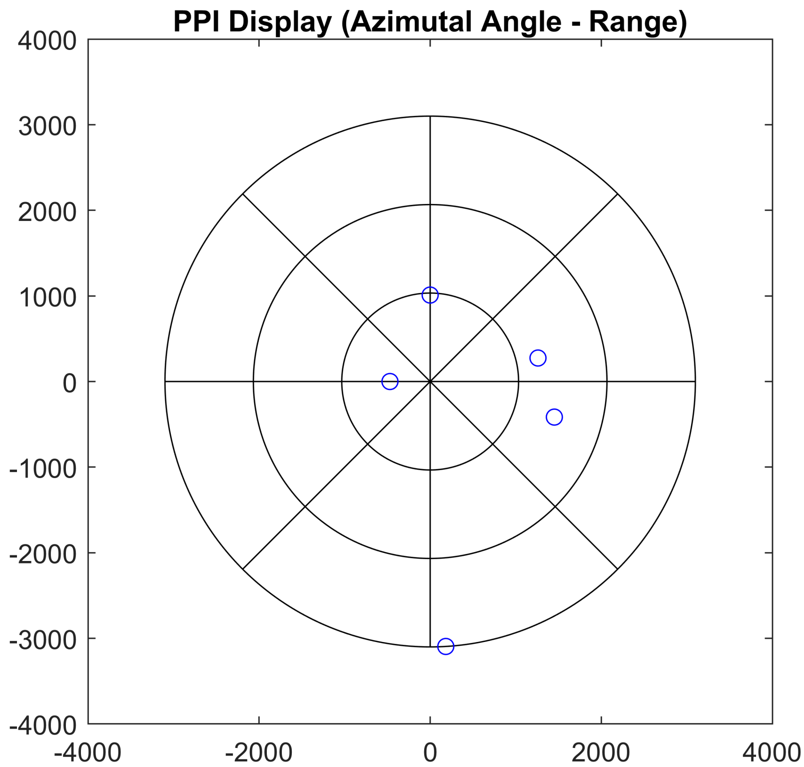
```
range_estimates = round(range_gates(range_detect))
```

```
range_estimates = 1×5
         470        1010        1290        1510        3100
```

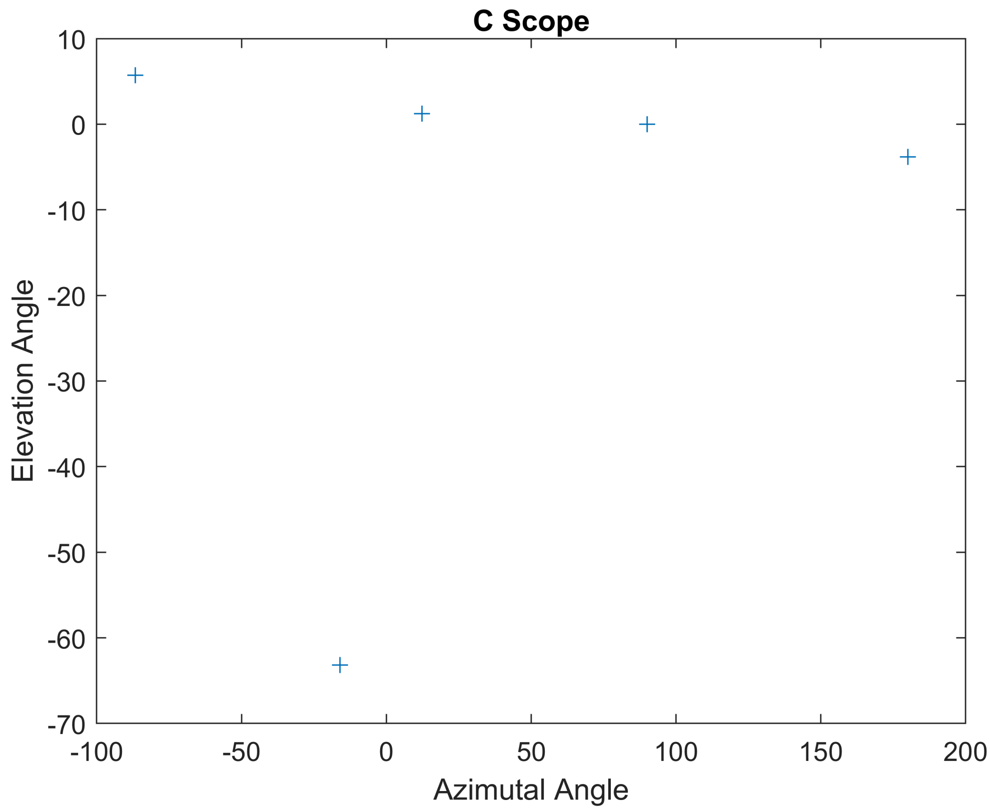
```
tgtazi= tgtang(1, :);
tgtele= tgtang(2, :);


radarplot(range_estimates, tgtazi);
```

```
  Name              Size             Bytes  Class                Attributes

  direction         1x5                 40  double
  errText           1x35                70  char
  gid               1x1                  8  matlab.ui.Figure
  speed             1x5                 40  double
  style             1x1                  2  char
  stylePicker       1x1                  8  double
  styleS            1x1                  2  char
  varargin          1x2                288  cell
```

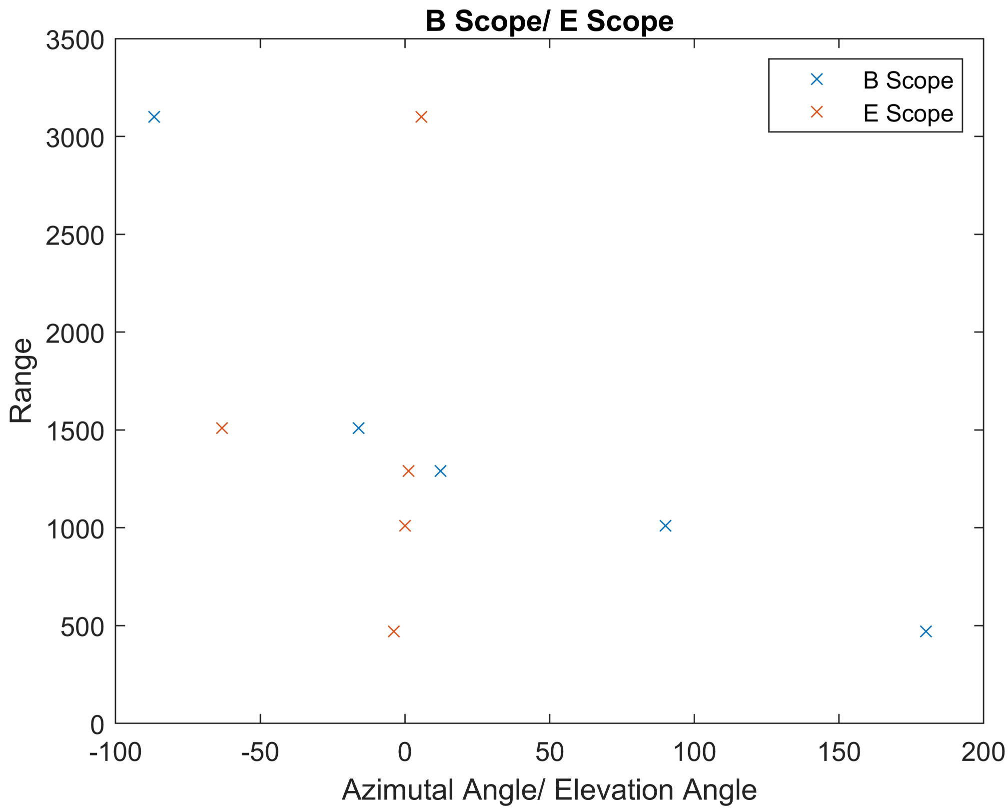
```
title("PPI Display (Azimutal Angle - Range)");
```



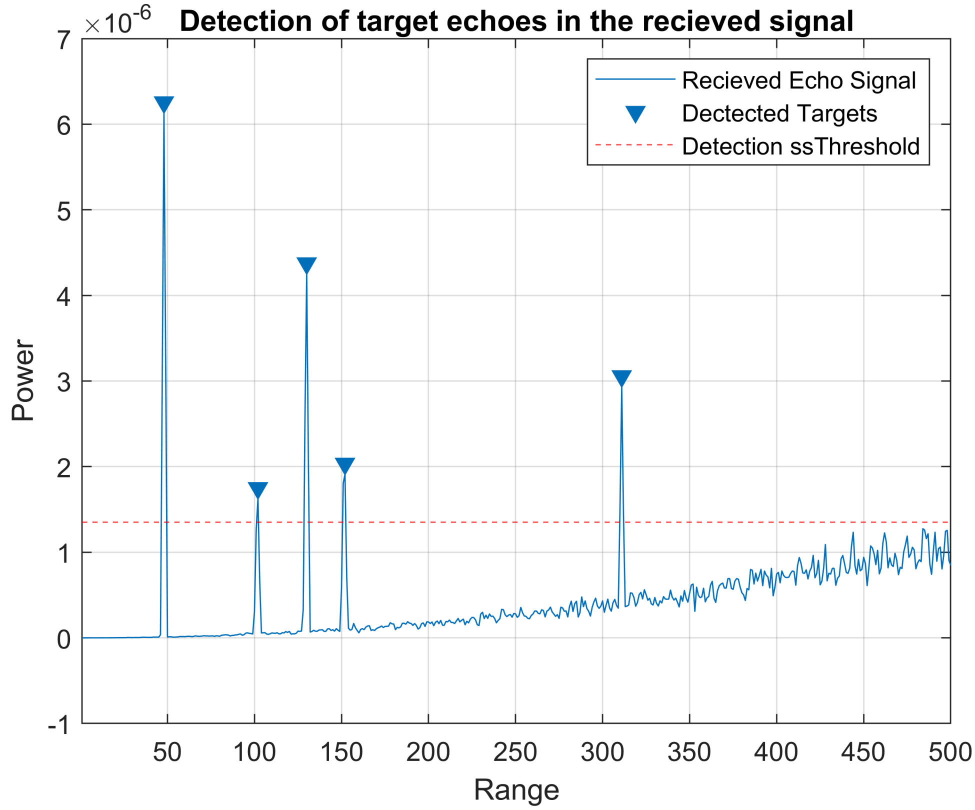

```
% radarplot(range_estimates, tgtele);
% title("Ele - Range");


plot(tgtazi, tgtele, "Marker","+", "LineStyle","none");
title("C Scope");
xlabel("Azimutal Angle");
ylabel("Elevation Angle");
```




```
plot(tgtazi,range_estimates, tgtele, range_estimates, "Marker","x", "LineStyle","none");
title("B Scope/ E Scope");
xlabel("Azimutal Angle/ Elevation Angle");
ylabel("Range");
legend("B Scope", "E Scope", "AutoUpdate","on")
```




```
findpeaks(rxpulses, "MinPeakHeight",1.5e-6);
th= yline(1.35e-6, 'r--');
title("Detection of target echoes in the recieved signal");
xlabel("Range");
ylabel("Power");
legend("Recieved Echo Signal", "Dectected Targets", "Detection ssThreshold");
```


Detection of target echoes in the recieved signal
×10^-6
Power
Range
Recieved Echo Signal
Dectected Targets
Detection ssThreshold
7
6
5
4
3
2
1
0
-1
50
100
150
200
250
300
350
400
450
500