```
pd = 0.9;            % Probability of detection
pfa = 1e-6;          % Probability of false alarm
MaxRange = 10e3;    % Maximum unambiguous range
res = 10;      % Required range resolution
rcs = 1;         % Required target radar cross section
NumPulses= 10;
%% Monostatic Radar System Design

prop_speed = 3e8;
c= prop_speed;
pulseWidth = 100e-6
```

```
pulseWidth = 1.0000e-04
```

```
pulseBw = 1/pulseWidth
```

```
pulseBw = 10000
```

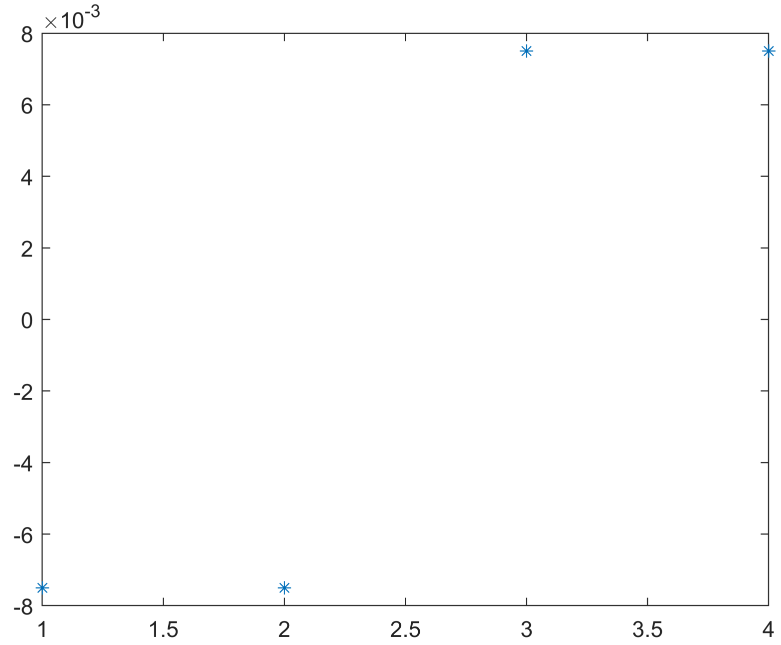
```
prf = 2e3;
fc = 10e9;
lambda= c/fc;
txGain= 20;
```

```
%Waveform Model
waveform= phased.RectangularWaveform('PRF', prf, 'PulseWidth', pulseWidth, ...
    'NumPulses', 1, 'OutputFormat', "Pulses");
fs= waveform.SampleRate
```

```
fs = 1000000
```

```
arraySize=[2 2];

antenna= phased.URA('Size', arraySize,'ElementSpacing', lambda/2, ...
    'ArrayNormal',"z");
%
% patternAzimuth(antenna, fc, 0);
% patternElevation(antenna, fc, 0);



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

angle= [0; 0];

response= antenna(fc, angle);


% patternAzimuth(antenna, fc, [-90:90]);

% patternElevation(antenna, fc, [-180:180]);
```

```
antennaPos= getElementPosition(antenna);
posRec= antennaPos(2:3,:);

axis([-1.5 1.5 -2 2]);
xlabel("x");
ylabel("y");
plot(antennaPos(1,:), 'LineStyle',"none", "Marker","*");
```



```
antennaPlatform= phased.Platform('InitialPosition', [0; 0; 0], 'Velocity',[0; 0; 0], 'ScanMode'

tgtPos = [1500; -400; 2000]
```

```
tgtPos = 3×1
        1500
        -400
        2000
```

```
tgtVel =  [0; 0; 0];
targetPlatform= phased.Platform('InitialPosition', tgtPos, 'Velocity', tgtVel );
tgtRcs= 15;
target= phased.RadarTarget("Model", "Nonfluctuating", "MeanRCS", tgtRcs,"PropagationSpeed", c,


[tgtRange, tgtAngle]= rangeangle(targetPlatform.InitialPosition, antennaPlatform.InitialPositio
```

```
tgtRange = 2.5318e+03
tgtAngle = 2×1
  -14.9314
   52.1811
```

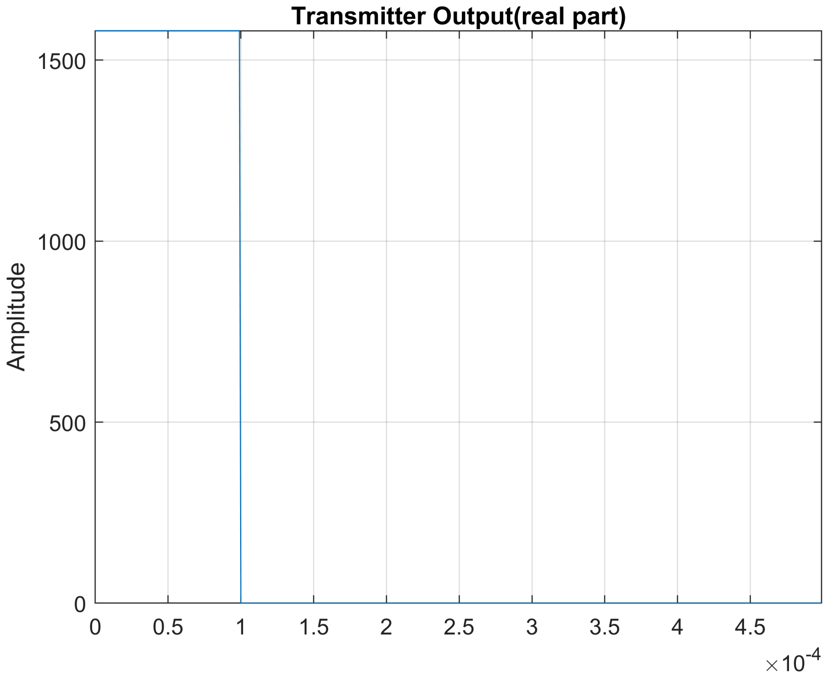
```
SNR= albersheim(pd, pfa, NumPulses)
```

```
SNR = 4.9904
```

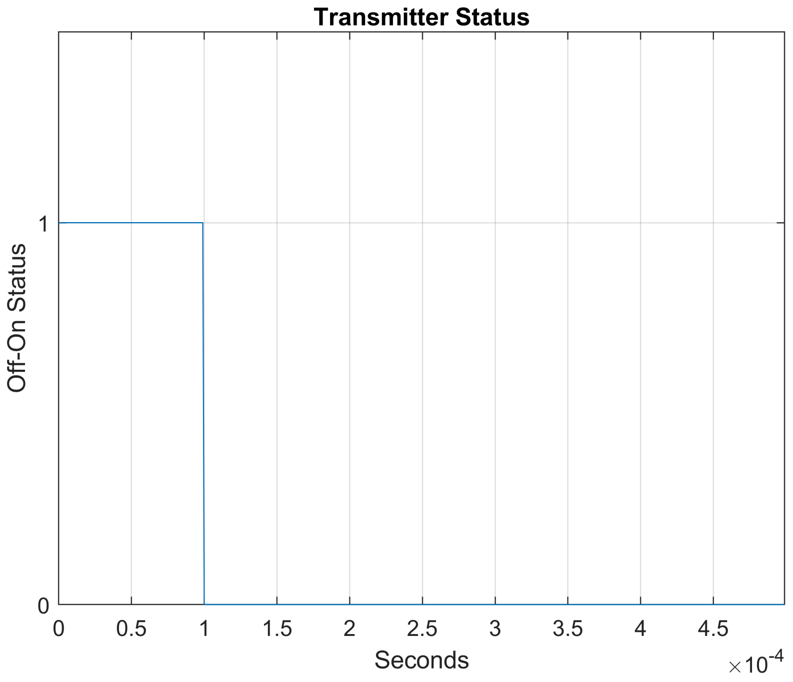
```

peakPower= radareqpow(lambda,MaxRange,SNR,pulseWidth,...
    'RCS',tgtRcs,'Gain',txGain)
```

```
peakPower = 18.5702
```

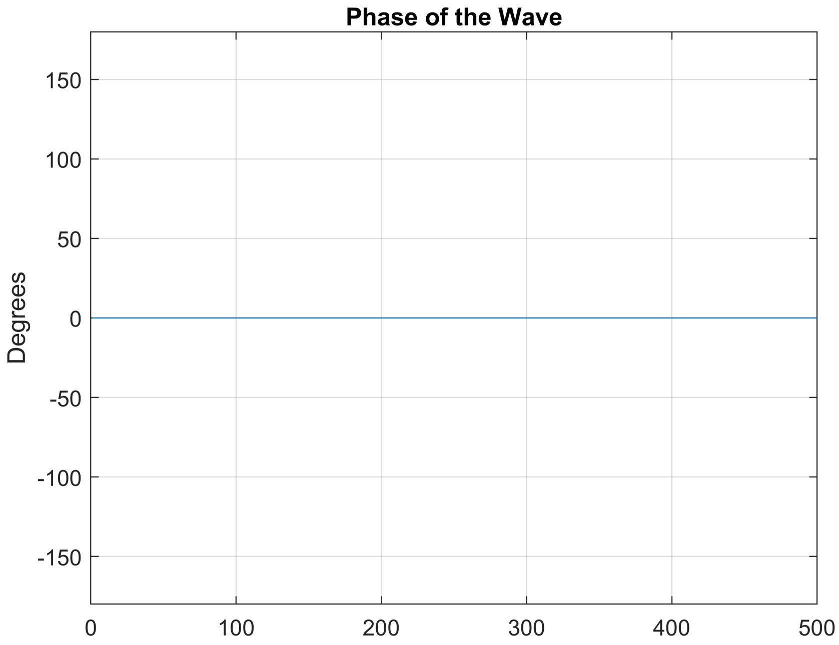
```
transmitter= phased.Transmitter("PeakPower", 25e3,'Gain', txGain, 'LossFactor', 0, ...
    'InUseOutputPort', true, 'CoherentOnTransmit', true);

wf=waveform();
[txOutput,txStatus] = transmitter(wf);
t = unigrid(0,1/waveform.SampleRate,1/waveform.PRF,'[)');
plot(t,real(txOutput))
axis tight
grid on
ylabel('Amplitude')
title('Transmitter Output(real part)')
```



```
plot(t,txStatus)
axis([0 t(end) 0 1.5])
xlabel('Seconds')
grid on
ylabel('Off-On Status')
set(gca,'ytick',[0 1])
title('Transmitter Status')
```



```
plot(rad2deg(atan2(imag(txOutput), real(txOutput))));
axis([0 length(wf) -180 180]);
ylabel('Degrees')
title('Phase of the Wave');
grid on;
```



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

collector= phased.Collector('Sensor', antenna, 'OperatingFrequency', ...
    fc, 'PropagationSpeed', c, ...
    'Wavefront', 'Plane');

receiver= phased.ReceiverPreamp('Gain', 20, 'NoiseFigure', 2, ...
    'LossFactor', 0, 'SampleRate', fs, 'EnableInputPort', ...
    true, 'SeedSource', 'Property', 'Seed', 2010, ...
    'PhaseNoiseInputPort', false, ...
    "NoiseComplexity", "Complex", "ReferenceTemperature", 290);
```

```
channel= phased.FreeSpace('OperatingFrequency', fc, ...
    'PropagationSpeed', c, 'SampleRate', fs, ...
    'MaximumDistance', 10e3, 'TwoWayPropagation', true);
```

```
Warning: The MaximumDistance property is not relevant in this configuration of the System object.
```

```
%End of Design
```

```
%Simulation and Implementation
T= 1/waveform.PRF;

txPos= antennaPlatform.InitialPosition;
txVel= antennaPlatform.InitialVelocity;
rxSig= zeros(waveform.SampleRate*T, NumPulses);




for n = 1:NumPulses
    % Update the target position
    [tgtPos,tgtVel] = targetPlatform(T);
    % Get the range and angle to the target
    [tgtRange,tgtAng] = rangeangle(tgtPos,txPos);
    % Generate the pulse
    sig = waveform();
    % Transmit the pulse. Output transmitter status
    [sig,txStatus] = transmitter(sig);
    % Radiate the pulse toward the target
    sig = radiator(sig,tgtAng);
    % Propagate the pulse to the target in free space
    sig = channel(sig,txPos,tgtPos,[0;0;0],tgtVel);
    % Reflect the pulse off the target
    sig = target(sig);
    % Propagate the echo to the antenna in free space
    sig = channel(sig,tgtPos,txPos,tgtVel,[0;0;0]);
    % Collect the echo from the incident angle at the antenna
    sig = collector(sig,tgtAng);
    % Receive the echo at the antenna when not transmitting
    rxSig(:,n) = receiver(sig,~txStatus);
```

```
  end
```

```
Unable to perform assignment because the size of the left side is 500-by-1 and the size of the right side is
 500-by-4.
```

```
rxSig= pulsint(rxSig, 'noncoherent');

t= unigrid(0, 1/receiver.SampleRate, T, '[)');
rangeGates= (c*t)/2;
%  plot(rangeGates/1000, rxSig);

threshold= noisePower*db2pow(npwgnthresh(pfa, NumPulses,'noncoherent'));
findpeaks(rxSig, 'MinPeakHeight',3.7e-7);
xlabel('range (km)');
ylabel('Power');

plot(rangeGates/1e3, rxSig);
```

```
noiseBw= pulseBw;
noisePower= noisepow(noiseBw, receiver.NoiseFigure, receiver.ReferenceTemperature);
```