Project 2 – Range Doppler Map

ECE538

John Claus

12/18/2019

Table of Contents

Problem Statement

Part 1

Model a radar system that employs stretch processing on a linear frequency waveform (LFM) chirp to create a range doppler map (RDM) from the following parameters for both center frequencies of 10GHz and 1GHz:

Radar Properties	
Tau	50 msec
BW	150 MHz
Range Swath	1 km
PRF	7.5 kHz
CPI	40 msec
RCS	15 dbsm
Noise	3 db
Gain	20 db
Aperature Time	40 msec
Transmit Power	10 dbW

Additionally, the radar is on a moving platform that has an initial location and velocity below:

Initial Radar Position		
-3	1000	304.8
Radar Velocity		
150	0	0

A collection of targets locations are also listed with the matrix:

Target Locations		
-100	-100	0
-100	100	0
100	-100	0
100	100	0
0	0	0

For part 1 of the project the center reference point (CRP A) below is also incorporated into the model.

Center Reference Point A		
10000	0	0

Plots of the received signal and RDM should for both center frequencies also are to be included.

Part 2

Using the same radar and platform, model the radar systems using the alternative CRP B below with center frequencies of 10Ghz and 1GHz and again plot the received signal and RDM for both.

Center Reference Point B		
0	0	0

Model Description

Part 1

The initial values were defined as follows:

```
% Initial Values
C = physconst('LightSpeed');    %mps
Fc_1 = 1e9;     %Hz
Fc_10 = 10e9;   %Hz
tau = 50e-6;    %s
BW = 150e6;     %Hz
Range_Swath = 1e3;  %m
PRF = 7500;     %Hz
PRI = 1/PRF;    %s
CPI = 40e-3;    %s
RCS_db = 15;    %dbsm
Noise_db = 3;   %db
Temp = 290;     %K
G_db = 20;      %db
Lsys_db = 0;    %db
Latm_db = 0;    %db
Aper_t = 40e-3; %s
Pt_db = 10;     %dbW
```

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.

```
% 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);
```

The initial platform velocities/location, CRP matrices, and target location matrix were also initialized here.

```
% Initial Radar Position
P_RDR = [-3; 10000; 304.8];
% Radar Velocity
V_RDR = [150; 0; 0];
% Center Reference Point
CRP_A = [10000, 0, 0];
CRP_B = [0, 0, 0];
% Scatter Matrix
L = [-100, -100, 0; -100, 100, 0; 100, -100, 0; 100, 100, 0; 0, 0, 0];
```

Further calculations necessary for the RDM plots and received signal creation were written here:

```
% Chirp Rate Calculation
Chirp_Rate = BW/tau;
% Sampling Frequency Calculation
Fs = 2*BW*Range_Swath/(C*tau);
% Samples per pulse
NS = round(Fs*tau);
% Time within a pulse
Pulse_Time = linspace(-tau/2,tau/2,NS);
% Number of Pulses
NP = round(Aper_t*PRF);
% Slow time vector
Slow_Time = linspace(0,Aper_t,NP);
% V vector calculation
V = norm(V_RDR)
% Aperature length calculation
Aper_len = Aper_t*V;  %m
% Phase History Initialization
Phase_Hist= zeros(NS,NP);
% number of targets
N_TGT = size(L,1);
```

The center frequency and CRP type were initialized before the signal processing loop:

```
%%% Fc = 1GHz CRP = A %%%%
% Set Fc to be 1GHz
Fc = Fc_1;
lambda = C/Fc;
% Set CRP to CRP A
CRP = CRP_A;
% Adjust for senter reference point
L_TGT = L;
for i=1:5
    L_TGT(i,:,:) = L(i,:,:) + CRP;
end
```

The received signal was then created and processed for all the cumulative positions

```
for np = 1:NP
    % Platform position at pulse "np"
    Current_P = P_RDR + V_RDR.*Slow_Time(np);
    % Motion Compensation/Chirp Reference Range
    R_MC(np) = norm(Current_P);
    % Initialize the received signal
    ReceivedSignal = zeros(NS,1);
    % Compute return signal for each target and sum
    for nt = 1:N_TGT
        % Range to target calculations
        R_Current = norm(Current_P-L_TGT(nt,:));
        Delay_Current    = 2*R_Current/C;
        R_TGT(nt,np) = R_MC(np)-R_Current;
        % Calculate Received Power
        Pr = (Pt*(G^2)*(lambda^2)*RCS)/(((4*pi)^3)*((R_TGT(nt,np))^4)*Lsys*Latm);
        sigma_sq = Pr*(10^(Noise_db/10));
        % Chirp Received Waveform/Signal;
        CurrentSignal = Pr*exp(-1i.*(2.*pi.*Fc.*(Pulse_Time.'+Slow_Time(np)-Delay_Current)+pi.*Chirp_Rate.*(Pulse_Time.'-Delay_Current).^2));
        Noise = 0;
        %Noise = sqrt(0.5.*sigma_sq).*(randn(size(CurrentSignal))+1j.*randn(size(CurrentSignal)));
        ReceivedSignal = ReceivedSignal + CurrentSignal + Noise;
    end
    % Reference Chirp (for stretch processing)
    Chirp_Ref = exp(1i.*(2.*pi.*Fc.*(Pulse_Time.'+Slow_Time(np)-2*(R_MC(np))/C)+pi.*Chirp_Rate.*(Pulse_Time.'-2*R_MC(np)./C).^2));
    % De-ramping
    Phase_Hist(:,np) = Chirp_Ref.*ReceivedSignal;
end
```

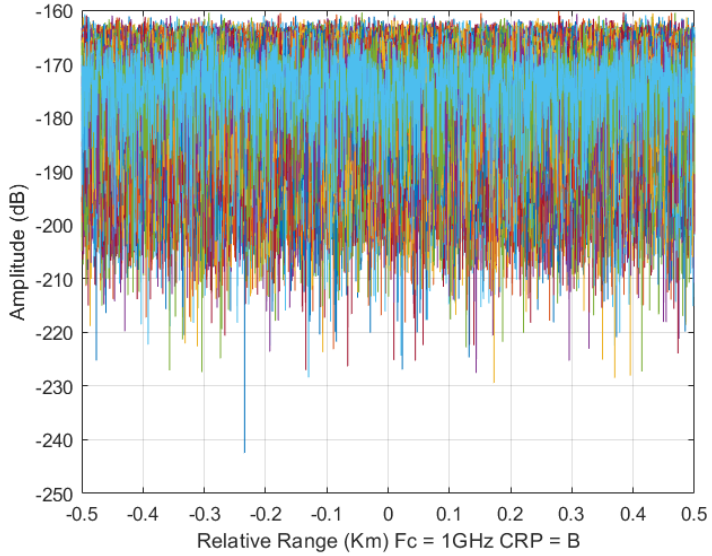
Part 2

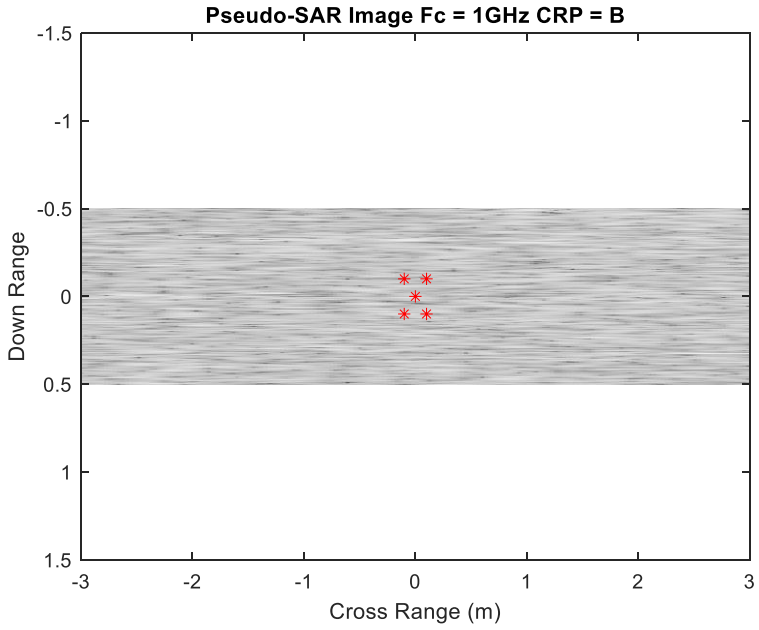
The above was repeated using CRP B

Results

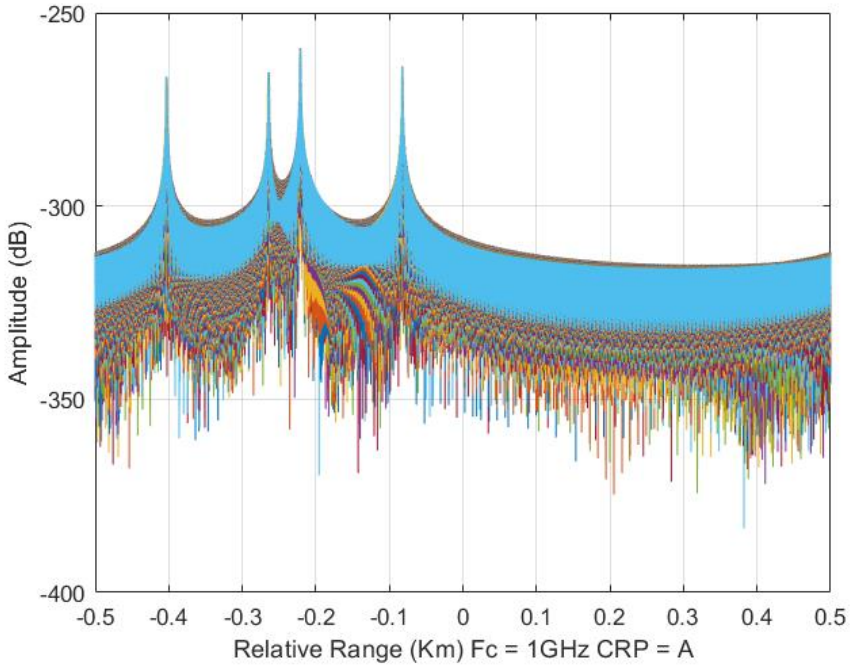
Part 1

The results of the project were not ideal, but due to deadline constraints I was unable to create a an RDM that accurately correlated target locations to the received pulse. Furthermore, as can be seen in Figure 1, my noise injection yielded no discernable signal whatsoever. Therefore, I was forced to disable it for troubleshooting purposes of the processing the clean signal to begin with.

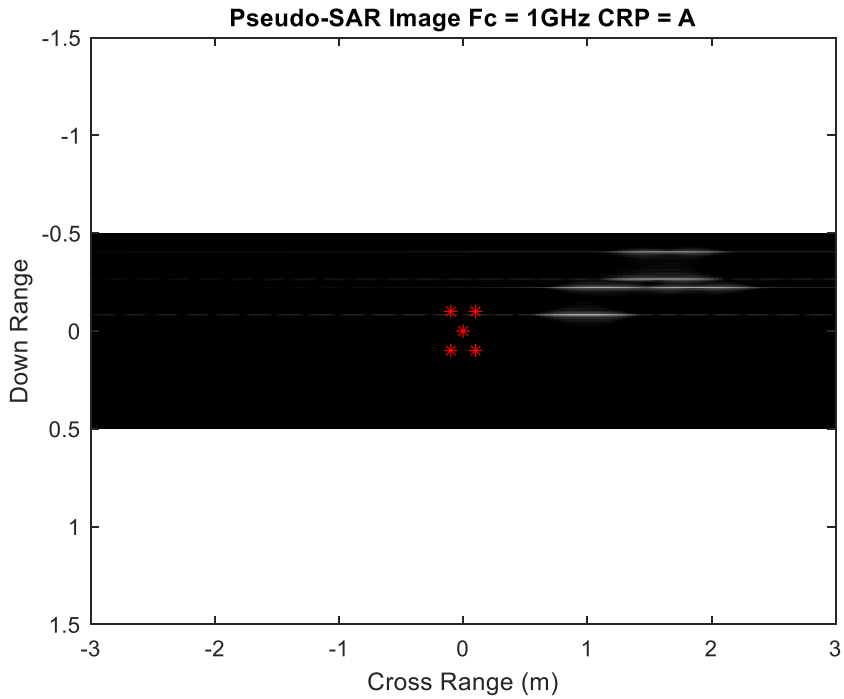




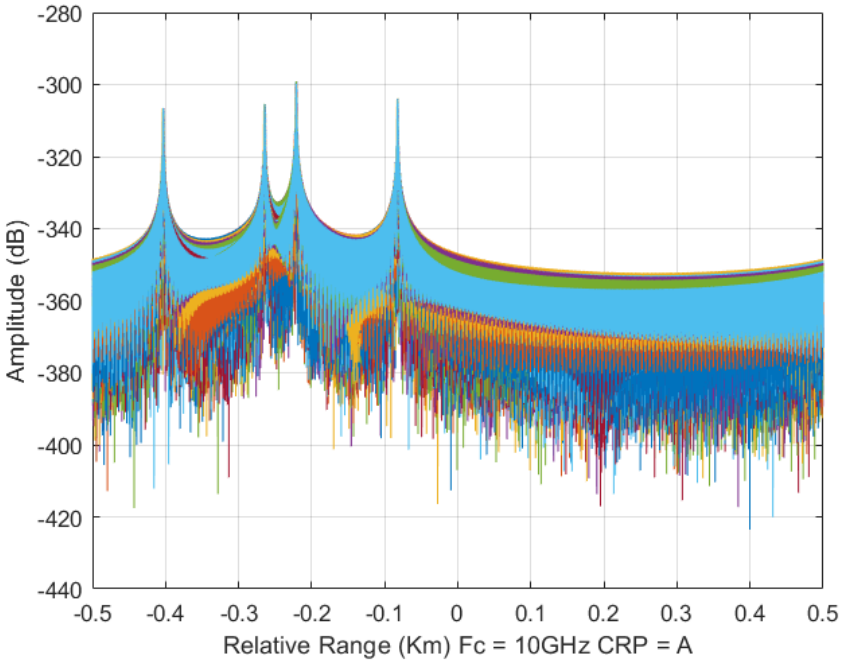
The received signal graph using the CRP A at a center frequency of 1GHz can be seen in Figure 2. Clearly the targets can be identified in the graph as the platform changes locations as indicated in the graph by a color change.



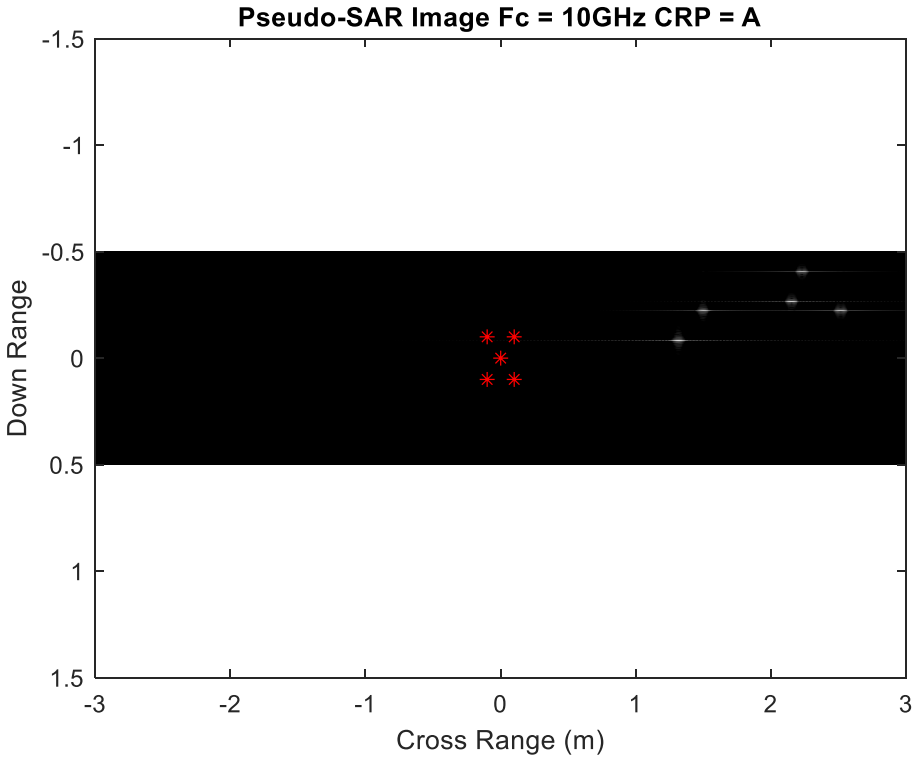
The RDM seen in Figure 3 however does not line up with the target positions. The returns can be seen as white lines but are not discernable or aligned as specific target location marked with an x.



Similar to the 1GHz returns, the 10GHz returns could be seen in the signal plots in Figure 4

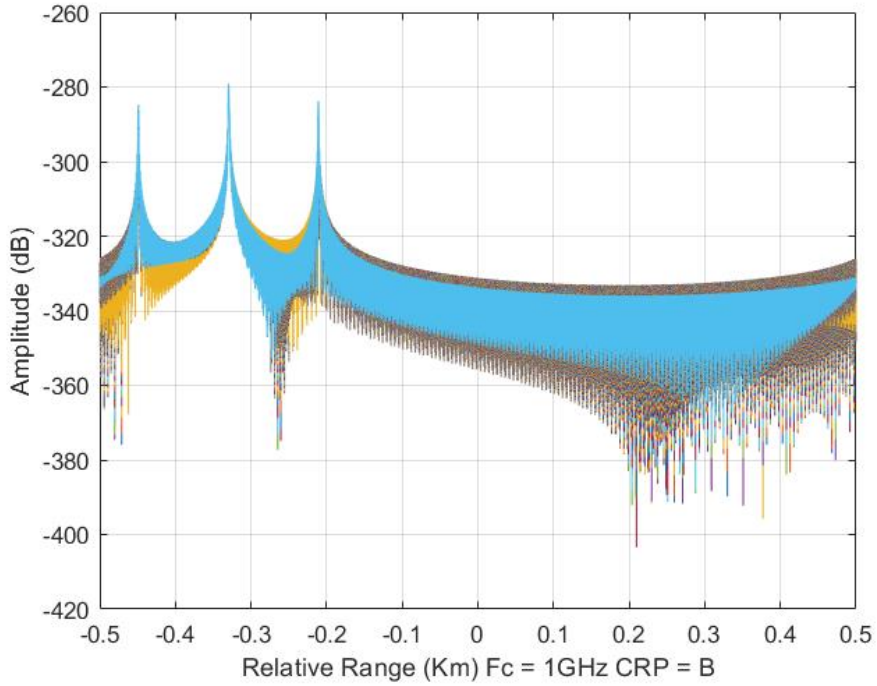


The 10GHz signal performed much better than did the 1GHz for the RDM but was still not aligned with the expected positions as can be seen in Figure 5

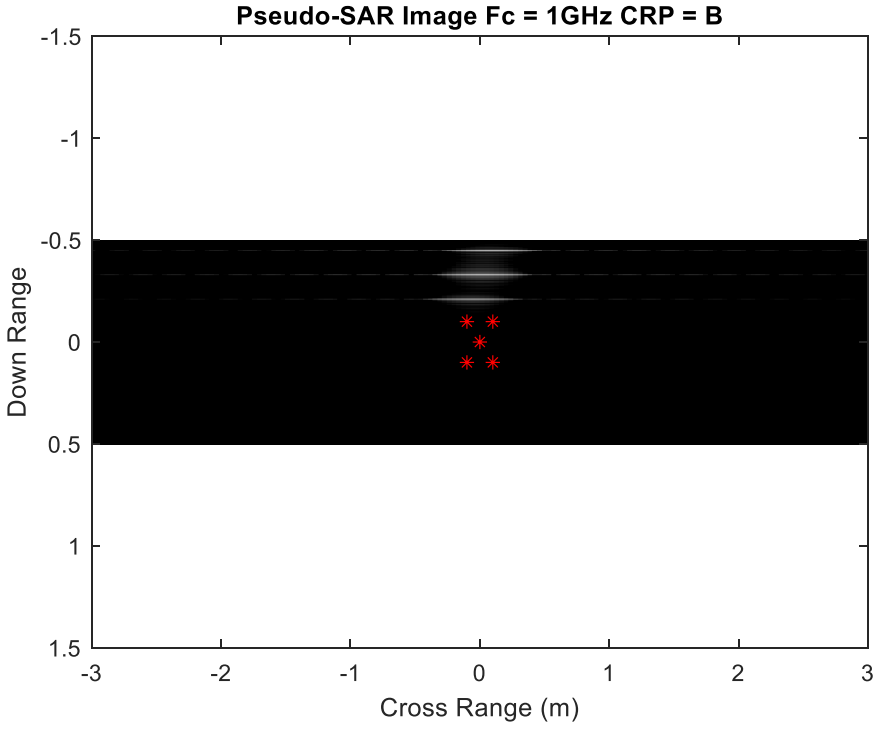


Part 2

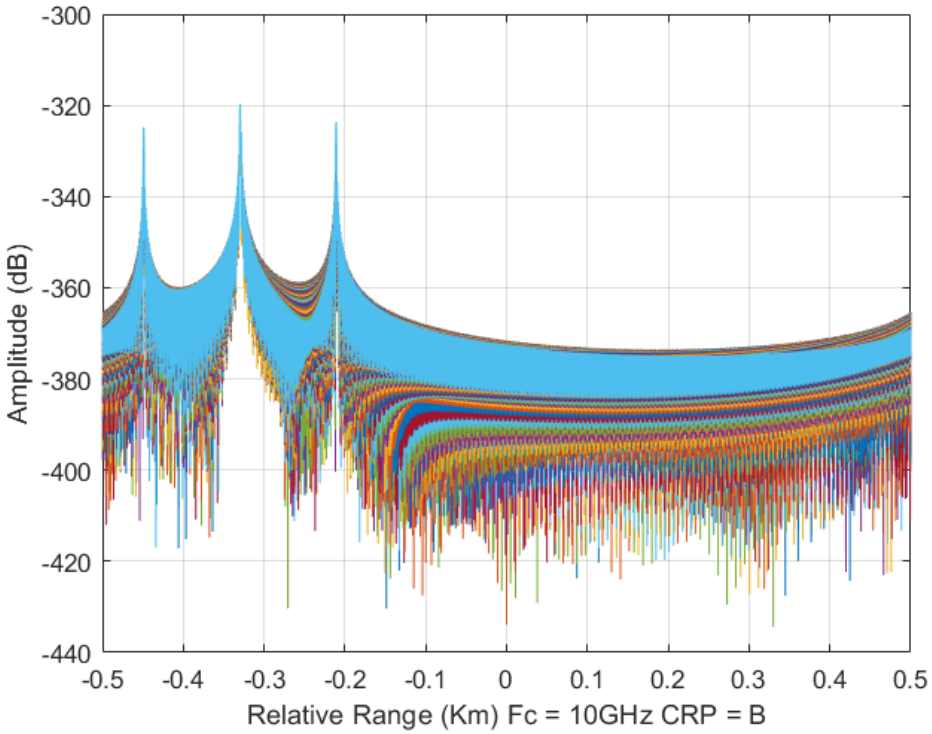
The CRP B performed almost identically to the CRP A with the following results. The 1 GHz received signal in Figure 6:



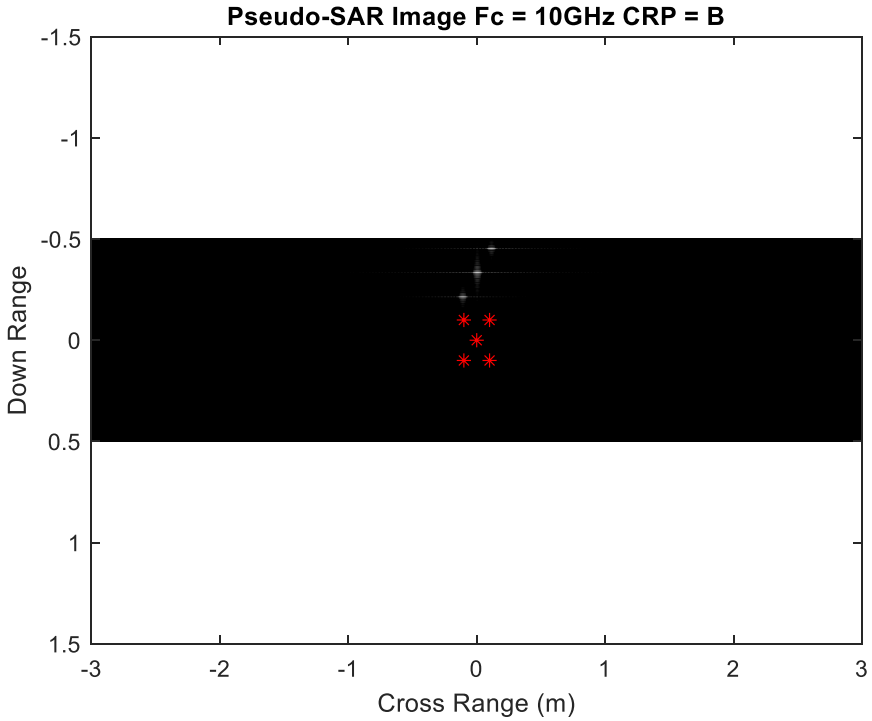
The 1Ghz RDM in Figure 7:



The 10GHz received signal in Figure 8:



The 10GHz RDM in Figure 9:



Discussion

The results of this model were a disaster. I think had I more time to develop the model, I might have been able to iron out some of these bugs and actually integrate the effects of noise into the signals as well as calculate squint angle. I did learn a great deal while trying to get the project to at least be somewhat presentable in the stretch processing methods used for a scenario where multiple layers of data are collected for the received signal at varying time steps and a moving platform. To be honest I am a little ashamed of this report, but am already three days overdue on it and therefore cannot devote any more time on the quality of the results.

Appendix – MATLAB Code

```
%%%% John Claus %%%%
%%%% ECE538     %%%%
%%%% Project 2  %%%%
%%%% 12/18/2019 %%%%

clear all
close all

% Initial Values
C = physconst('LightSpeed');    %mps
Fc_1 = 1e9;     %Hz
Fc_10 = 10e9;   %Hz
tau = 50e-6;    %s
BW = 150e6;     %Hz
Range_Swath = 1e3;  %m
PRF = 7500;     %Hz
PRI = 1/PRF;    %s
CPI = 40e-3;    %s
RCS_db = 15;    %dbsm
Noise_db = 3;   %db
Temp = 290;     %K
G_db = 20;      %db
Lsys_db = 0;    %db
Latm_db = 0;    %db
Aper_t = 40e-3; %s
Pt_db = 10;     %dbW

% 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);

% Initial Radar Position
P_RDR = [-3; 10000; 304.8];
% Radar Velocity
V_RDR = [150; 0; 0];
% Center Reference Point
CRP_A = [10000, 0, 0];
CRP_B = [0, 0, 0];
% Scatter Matrix
L = [-100, -100, 0; -100, 100, 0; 100, -100, 0; 100, 100, 0; 0, 0, 0];


% Chirp Rate Calculation
Chirp_Rate = BW/tau;
% Sampling Frequency Calculation
Fs = 2*BW*Range_Swath/(C*tau);
% Samples per pulse
NS = round(Fs*tau);
% Time within a pulse
Pulse_Time = linspace(-tau/2,tau/2,NS);
% Number of Pulses
```

```
NP = round(Aper_t*PRF);
% Slow time vector
Slow_Time = linspace(0,Aper_t,NP);
% V vector calculation
V = norm(V_RDR)
% Aperature length calculation
Aper_len = Aper_t*V;  %m
% Phase History Initialization
Phase_Hist= zeros(NS,NP);
% number of targets
N_TGT = size(L,1);

% Display Chirp Rate and Sampling Frequency calculations
disp(['Chirp Parameters: Chirp Rate ',num2str(Chirp_Rate),' Hz/s'])
disp(['Chirp Parameters: Sampling Frequency',num2str(Fs/1e6),' MHz'])


%%% Fc = 1GHz CRP = A %%%%
% Set Fc to be 1GHz
Fc = Fc_1;
lambda = C/Fc;
% Set CRP to CRP A
CRP = CRP_A;
% Adjust for senter reference point
L_TGT = L;
for i=1:5
    L_TGT(i,:,:) = L(i,:,:) + CRP;
end
for np = 1:NP
    % Platform position at pulse "np"
    Current_P = P_RDR + V_RDR.*Slow_Time(np);
    % Motion Compensation/Chirp Reference Range
    R_MC(np) = norm(Current_P);
    % Initialize the received signal
    ReceivedSignal = zeros(NS,1);
    % Compute return signal for each target and sum
    for nt = 1:N_TGT
        % Range to target calculations
        R_Current = norm(Current_P-L_TGT(nt,:));
        Delay_Current    = 2*R_Current/C;
        R_TGT(nt,np) = R_MC(np)-R_Current;
        % Calculate Received Power
        Pr = (Pt*(G^2)*(lambda^2)*RCS)/(((4*pi)^3)*((R_TGT(nt,np))^4)*Lsys*Latm);
        sigma_sq = Pr*(10^(Noise_db/10));
        % Chirp Received Waveform/Signal;
        CurrentSignal = Pr*exp(-1i.*(2.*pi.*Fc.*(Pulse_Time.'+Slow_Time(np)-
Delay_Current)+pi.*Chirp_Rate.*(Pulse_Time.'-Delay_Current).^2));
        Noise = 0;
        %Noise =
sqrt(0.5.*sigma_sq).*(randn(size(CurrentSignal))+1j.*randn(size(CurrentSignal)));
        ReceivedSignal = ReceivedSignal + CurrentSignal + Noise;
    end
    % Reference Chirp (for stretch processing)
    Chirp_Ref = exp(1i.*(2.*pi.*Fc.*(Pulse_Time.'+Slow_Time(np)-
2*(R_MC(np))/C)+pi.*Chirp_Rate.*(Pulse_Time.'-2*R_MC(np)./C).^2));
    % De-ramping
    Phase_Hist(:,np) = Chirp_Ref.*ReceivedSignal;
end
% Range Profile
NFFT = 8192;
rngProfile = fftshift(ifft(Phase_Hist,NFFT));

% Map Frequency to Range
fftFreq = linspace(-Fs/2,Fs/2,NFFT);
timeVec = fftFreq.*tau./BW;
rngVec  = C.*timeVec./2;
figure,plot(rngVec./1e3,20.*log10(abs(rngProfile)))
xlabel('Relative Range (Km) Fc = 1GHz CRP = A')
ylabel('Amplitude (dB)')
grid on;
```

```
R0 = min(R_MC);
% Setting up image sixze
NFFT = 2048;
NFFT2 = 4096;
% Frequency to range conversion for the deramped chirp
freqVec = linspace(-Fs/2,Fs/2,NFFT);
tVec = freqVec./Chirp_Rate;
rngVec = C.*tVec./2;
% Applying fast and slow time windows to the data (so it looks nice)
phW = repmat(taylorwin(size(Phase_Hist,1),4,-
35),1,size(Phase_Hist,2)).*repmat(taylorwin(size(Phase_Hist,2),4,-
35).',size(Phase_Hist,1),1).*Phase_Hist;
% Compress the data in range
rngComp = fftshift(ifft(phW,NFFT,1),1);
% Form the initial RDM
im = fftshift(fft(rngComp,NFFT,2),2);
% Doppler Vector
dopVec = linspace(-PRF/2,PRF/2,NFFT);
% Cross Range map at scene center
crMap = lambda.*R0.*dopVec./(2.*V);
% figure,imagesc(1:NP,rngVec,20.*log10(abs(rngComp)))
% % hold on;plot(1:NP,tgtRng,'k--')
% xlabel('Pulse Number')
% ylabel('Slant Range (m)')
% title('Range Compressed Data - After Motion Compensation Fc = 1GHz CRP = A')

figure,imagesc(crMap./1e3,rngVec./1e3,20.*log10(abs(im)))
hold on;plot(L(:,1)./1e3,L(:,2)./1e3,'r*')
ylim([-1.5 1.5])
xlim([-1.5 1.5].*2)
colormap gray
caxis([20.*log10(max(abs(im(:))))-60 20.*log10(max(abs(im(:))))])
title('Pseudo-SAR Image Fc = 1GHz CRP = A')
xlabel('Cross Range (m)')
ylabel('Down Range')

%%% Fc = 1GHz CRP = B %%%%
% Set Fc to be 1GHz
Fc = Fc_1;
lambda = C/Fc;
% Set CRP to CRP B
CRP = CRP_B;
% Adjust for senter reference point
L_TGT = L;
for i=1:5
    L_TGT(i,:,:) = L(i,:,:) + CRP;
end
for np = 1:NP
    % Platform position at pulse "np"
    Current_P = P_RDR + V_RDR.*Slow_Time(np);
    % Motion Compensation/Chirp Reference Range
    R_MC(np) = norm(Current_P);
    % Initialize the received signal
    ReceivedSignal = zeros(NS,1);
    % Compute return signal for each target and sum
    for nt = 1:N_TGT
        % Range to target calculations
        R_Current = norm(Current_P-L_TGT(nt,:));
        Delay_Current    = 2*R_Current/C;
        R_TGT(nt,np) = R_MC(np)-R_Current;
        % Calculate Received Power
        Pr = (Pt*(G^2)*(lambda^2)*RCS)/(((4*pi)^3)*((R_TGT(nt,np))^4)*Lsys*Latm);
        sigma_sq = Pr*(10^(Noise_db/10));
        % Chirp Received Waveform/Signal;
        CurrentSignal = Pr*exp(-1i.*(2.*pi.*Fc.*(Pulse_Time.'+Slow_Time(np)-
Delay_Current)+pi.*Chirp_Rate.*(Pulse_Time.'-Delay_Current).^2));
        Noise = 0;
        %Noise =
sqrt(0.5.*sigma_sq).*(randn(size(CurrentSignal))+1j.*randn(size(CurrentSignal)));
        ReceivedSignal = ReceivedSignal + CurrentSignal + Noise;
    end
```

```
    % Reference Chirp (for stretch processing)
    Chirp_Ref = exp(1i.*(2.*pi.*Fc.*(Pulse_Time.'+Slow_Time(np)-
2*(R_MC(np))/C)+pi.*Chirp_Rate.*(Pulse_Time.'-2*R_MC(np)./C).^2));
    % De-ramping
    Phase_Hist(:,np) = Chirp_Ref.*ReceivedSignal;
end
% Range Profile
NFFT = 8192;
rngProfile = fftshift(ifft(Phase_Hist,NFFT));

% Map Frequency to Range
fftFreq = linspace(-Fs/2,Fs/2,NFFT);
timeVec = fftFreq.*tau./BW;
rngVec  = C.*timeVec./2;
figure,plot(rngVec./1e3,20.*log10(abs(rngProfile)))
xlabel('Relative Range (Km) Fc = 1GHz CRP = B')
ylabel('Amplitude (dB)')
grid on;

R0 = min(R_MC);
% Setting up image sixze
NFFT = 2048;
NFFT2 = 4096;
% Frequency to range conversion for the deramped chirp
freqVec = linspace(-Fs/2,Fs/2,NFFT);
tVec = freqVec./Chirp_Rate;
rngVec = C.*tVec./2;
% Applying fast and slow time windows to the data (so it looks nice)
phW = repmat(taylorwin(size(Phase_Hist,1),4,-
35),1,size(Phase_Hist,2)).*repmat(taylorwin(size(Phase_Hist,2),4,-
35).',size(Phase_Hist,1),1).*Phase_Hist;
% Compress the data in range
rngComp = fftshift(ifft(phW,NFFT,1),1);
% Form the initial RDM
im = fftshift(fft(rngComp,NFFT,2),2);
% Doppler Vector
dopVec = linspace(-PRF/2,PRF/2,NFFT);
% Cross Range map at scene center
crMap = lambda.*R0.*dopVec./(2.*V);
% figure,imagesc(1:NP,rngVec,20.*log10(abs(rngComp)))
% % hold on;plot(1:NP,tgtRng,'k--')
% xlabel('Pulse Number')
% ylabel('Slant Range (m)')
% title('Range Compressed Data - After Motion Compensation Fc = 1GHz CRP = B')

figure,imagesc(crMap./1e3,rngVec./1e3,20.*log10(abs(im)))
hold on;plot(L(:,1)./1e3,L(:,2)./1e3,'r*')
ylim([-1.5 1.5])
xlim([-1.5 1.5].*2)
colormap gray
caxis([20.*log10(max(abs(im(:))))-60 20.*log10(max(abs(im(:))))])
title('Pseudo-SAR Image Fc = 1GHz CRP = B')
xlabel('Cross Range (m)')
ylabel('Down Range')

%%% Fc = 10GHz CRP = A %%%%
% Set Fc to be 1GHz
Fc = Fc_10;
lambda = C/Fc;
% Set CRP to CRP A
CRP = CRP_A;
% Adjust for senter reference point
L_TGT = L;
for i=1:5
    L_TGT(i,:,:) = L(i,:,:) + CRP;
end
for np = 1:NP
    % Platform position at pulse "np"
    Current_P = P_RDR + V_RDR.*Slow_Time(np);
    % Motion Compensation/Chirp Reference Range
    R_MC(np) = norm(Current_P);
```

```
    % Initialize the received signal
    ReceivedSignal = zeros(NS,1);
    % Compute return signal for each target and sum
    for nt = 1:N_TGT
        % Range to target calculations
        R_Current = norm(Current_P-L_TGT(nt,:));
        Delay_Current    = 2*R_Current/C;
        R_TGT(nt,np) = R_MC(np)-R_Current;
        % Calculate Received Power
        Pr = (Pt*(G^2)*(lambda^2)*RCS)/(((4*pi)^3)*((R_TGT(nt,np))^4)*Lsys*Latm);
        sigma_sq = Pr*(10^(Noise_db/10));
        % Chirp Received Waveform/Signal;
        CurrentSignal = Pr*exp(-1i.*(2.*pi.*Fc.*(Pulse_Time.'+Slow_Time(np)-
Delay_Current)+pi.*Chirp_Rate.*(Pulse_Time.'-Delay_Current).^2));
        Noise = 0;
        %Noise =
sqrt(0.5.*sigma_sq).*(randn(size(CurrentSignal))+1j.*randn(size(CurrentSignal)));
        ReceivedSignal = ReceivedSignal + CurrentSignal + Noise;
    end
    % Reference Chirp (for stretch processing)
    Chirp_Ref = exp(1i.*(2.*pi.*Fc.*(Pulse_Time.'+Slow_Time(np)-
2*(R_MC(np))/C)+pi.*Chirp_Rate.*(Pulse_Time.'-2*R_MC(np)./C).^2));
    % De-ramping
    Phase_Hist(:,np) = Chirp_Ref.*ReceivedSignal;
end
% Range Profile
NFFT = 8192;
rngProfile = fftshift(ifft(Phase_Hist,NFFT));

% Map Frequency to Range
fftFreq = linspace(-Fs/2,Fs/2,NFFT);
timeVec = fftFreq.*tau./BW;
rngVec  = C.*timeVec./2;
figure,plot(rngVec./1e3,20.*log10(abs(rngProfile)))
xlabel('Relative Range (Km) Fc = 10GHz CRP = A')
ylabel('Amplitude (dB)')
grid on;

R0 = min(R_MC);
% Setting up image sixze
NFFT = 2048;
NFFT2 = 4096;
% Frequency to range conversion for the deramped chirp
freqVec = linspace(-Fs/2,Fs/2,NFFT);
tVec = freqVec./Chirp_Rate;
rngVec = C.*tVec./2;
% Applying fast and slow time windows to the data (so it looks nice)
phW = repmat(taylorwin(size(Phase_Hist,1),4,-
35),1,size(Phase_Hist,2)).*repmat(taylorwin(size(Phase_Hist,2),4,-
35).',size(Phase_Hist,1),1).*Phase_Hist;
% Compress the data in range
rngComp = fftshift(ifft(phW,NFFT,1),1);
% Form the initial RDM
im = fftshift(fft(rngComp,NFFT,2),2);
% Doppler Vector
dopVec = linspace(-PRF/2,PRF/2,NFFT);
% Cross Range map at scene center
crMap = lambda.*R0.*dopVec./(2.*V);
% figure,imagesc(1:NP,rngVec,20.*log10(abs(rngComp)))
% % hold on;plot(1:NP,tgtRng,'k--')
% xlabel('Pulse Number')
% ylabel('Slant Range (m)')
% title('Range Compressed Data - After Motion Compensation Fc = 1GHz CRP = A')

figure,imagesc(crMap./1e3,rngVec./1e3,20.*log10(abs(im)))
hold on;plot(L(:,1)./1e3,L(:,2)./1e3,'r*')
ylim([-1.5 1.5])
xlim([-1.5 1.5].*2)
colormap gray
caxis([20.*log10(max(abs(im(:))))-60 20.*log10(max(abs(im(:))))])
title('Pseudo-SAR Image Fc = 10GHz CRP = A')
```

```
xlabel('Cross Range (m)')
ylabel('Down Range')

%%% Fc = 1GHz CRP = B %%%%
% Set Fc to be 1GHz
Fc = Fc_10;
lambda = C/Fc;
% Set CRP to CRP B
CRP = CRP_B;
% Adjust for senter reference point
L_TGT = L;
for i=1:5
    L_TGT(i,:,:) = L(i,:,:) + CRP;
end
for np = 1:NP
    % Platform position at pulse "np"
    Current_P = P_RDR + V_RDR.*Slow_Time(np);
    % Motion Compensation/Chirp Reference Range
    R_MC(np) = norm(Current_P);
    % Initialize the received signal
    ReceivedSignal = zeros(NS,1);
    % Compute return signal for each target and sum
    for nt = 1:N_TGT
        % Range to target calculations
        R_Current = norm(Current_P-L_TGT(nt,:));
        Delay_Current    = 2*R_Current/C;
        R_TGT(nt,np) = R_MC(np)-R_Current;
        % Calculate Received Power
        Pr = (Pt*(G^2)*(lambda^2)*RCS)/(((4*pi)^3)*((R_TGT(nt,np))^4)*Lsys*Latm);
        sigma_sq = Pr*(10^(Noise_db/10));
        % Chirp Received Waveform/Signal;
        CurrentSignal = Pr*exp(-1i.*(2.*pi.*Fc.*(Pulse_Time.'+Slow_Time(np)-
Delay_Current)+pi.*Chirp_Rate.*(Pulse_Time.'-Delay_Current).^2));
        Noise = 0;
        %Noise =
sqrt(0.5.*sigma_sq).*(randn(size(CurrentSignal))+1j.*randn(size(CurrentSignal)));
        ReceivedSignal = ReceivedSignal + CurrentSignal + Noise;
    end
    % Reference Chirp (for stretch processing)
    Chirp_Ref = exp(1i.*(2.*pi.*Fc.*(Pulse_Time.'+Slow_Time(np)-
2*(R_MC(np))/C)+pi.*Chirp_Rate.*(Pulse_Time.'-2*R_MC(np)./C).^2));
    % De-ramping
    Phase_Hist(:,np) = Chirp_Ref.*ReceivedSignal;
end
% Range Profile
NFFT = 8192;
rngProfile = fftshift(ifft(Phase_Hist,NFFT));

% Map Frequency to Range
fftFreq = linspace(-Fs/2,Fs/2,NFFT);
timeVec = fftFreq.*tau./BW;
rngVec  = C.*timeVec./2;
figure,plot(rngVec./1e3,20.*log10(abs(rngProfile)))
xlabel('Relative Range (Km) Fc = 10GHz CRP = B')
ylabel('Amplitude (dB)')
grid on;

R0 = min(R_MC);
% Setting up image sixze
NFFT = 2048;
NFFT2 = 4096;
% Frequency to range conversion for the deramped chirp
freqVec = linspace(-Fs/2,Fs/2,NFFT);
tVec = freqVec./Chirp_Rate;
rngVec = C.*tVec./2;
% Applying fast and slow time windows to the data (so it looks nice)
phW = repmat(taylorwin(size(Phase_Hist,1),4,-
35),1,size(Phase_Hist,2)).*repmat(taylorwin(size(Phase_Hist,2),4,-
35).',size(Phase_Hist,1),1).*Phase_Hist;
% Compress the data in range
rngComp = fftshift(ifft(phW,NFFT,1),1);
```

```
% Form the initial RDM
im = fftshift(fft(rngComp,NFFT,2),2);
% Doppler Vector
dopVec = linspace(-PRF/2,PRF/2,NFFT);
% Cross Range map at scene center
crMap = lambda.*R0.*dopVec./(2.*V);
% figure,imagesc(1:NP,rngVec,20.*log10(abs(rngComp)))
% % hold on;plot(1:NP,tgtRng,'k--')
% xlabel('Pulse Number')
% ylabel('Slant Range (m)')
% title('Range Compressed Data - After Motion Compensation Fc = 10GHz CRP = B')

figure,imagesc(crMap./1e3,rngVec./1e3,20.*log10(abs(im)))
hold on;plot(L(:,1)./1e3,L(:,2)./1e3,'r*')
ylim([-1.5 1.5])
xlim([-1.5 1.5].*2)
colormap gray
caxis([20.*log10(max(abs(im(:))))-60 20.*log10(max(abs(im(:))))])
title('Pseudo-SAR Image Fc = 10GHz CRP = B')
xlabel('Cross Range (m)')
ylabel('Down Range')
```