RADAR TARGET GENERATION AND DETECTION

Student: Hiep Truong Cong

```
clear all
clc;
```

Radar Specifications

Frequency of operation = 77GHz

Max Range = 200m

Range Resolution = 1 m

Max Velocity = 100 m/s

```
RangeResolution = 1;
Rmax = 200;
Vmax = 100;
c = 3e8;

%speed of light = 3e8
%% User Defined Range and Velocity of target
% *%TODO* :
% define the target's initial position and velocity. Note : Velocity
% remains contant
R = 110
```

```
R = 110
```

```
v = -20;
```

FMCW Waveform Generation

```
% *%TODO* :
%Design the FMCW waveform by giving the specs of each of its parameters.
% Calculate the Bandwidth (B), Chirp Time (Tchirp) and Slope (slope) of the FMCW
% chirp using the requirements above.

B = c / (2 * RangeResolution);
Tchirp = 5.5 * 2 * Rmax / c;
Slope = B / Tchirp
```

```
Slope = 2.0455e+13
```

```

%Operating carrier frequency of Radar
fc= 77e9;             %carrier freq

%The number of chirps in one sequence. Its ideal to have 2^ value for the ease of running the
```

```
%for Doppler Estimation.
Nd=128;                   % #of doppler cells OR #of sent periods % number of chirps

%The number of samples on each chirp.
Nr=1024;                  %for length of time OR # of range cells

% Timestamp for running the displacement scenario for every sample on each
% chirp
t=linspace(0,Nd*Tchirp,Nr*Nd); %total time for samples


%Creating the vectors for Tx, Rx and Mix based on the total samples input.
Tx=zeros(1,length(t)); %transmitted signal
Rx=zeros(1,length(t)); %received signal
Mix = zeros(1,length(t)); %beat signal

%Similar vectors for range_covered and time delay.
r_t=zeros(1,length(t));
td=zeros(1,length(t));
```

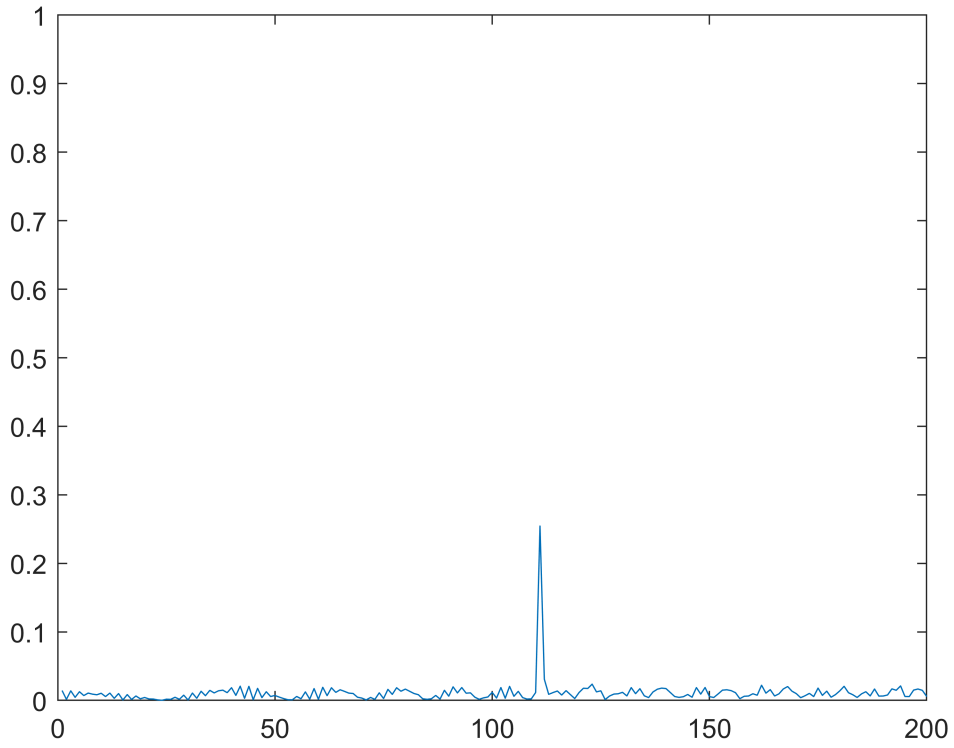
Signal generation and Moving Target simulation

```
%%
% Running the radar scenario over the time.
%
% *%TODO* :
%For each time stamp update the Range of the Target for constant velocity.
r_t = R + (v * t);
td = 2 * r_t / c;
% *%TODO* :
%For each time sample we need update the transmitted and
%received signal.
Tx = cos(2*pi*(fc.*t + (0.5*Slope.*t.*t)));
Rx = cos(2*pi*(fc.*(t - td) + (Slope*(t-td).*(t-td)/2)));

% *%TODO* :
%Now by mixing the Transmit and Receive generate the beat signal
%This is done by element wise matrix multiplication of Transmit and
%Receiver Signal
Mix = Tx.*Rx;
```

RANGE MEASUREMENT

```
% *%TODO* :
%reshape the vector into Nr*Nd array. Nr and Nd here would also define the size of
%Range and Doppler FFT respectively.
Mix = reshape(Mix,[Nr,Nd]);

 % *%TODO* :
%run the FFT on the beat signal along the range bins dimension (Nr) and
%normalize.
fft1 = fft(Mix,Nr);
```

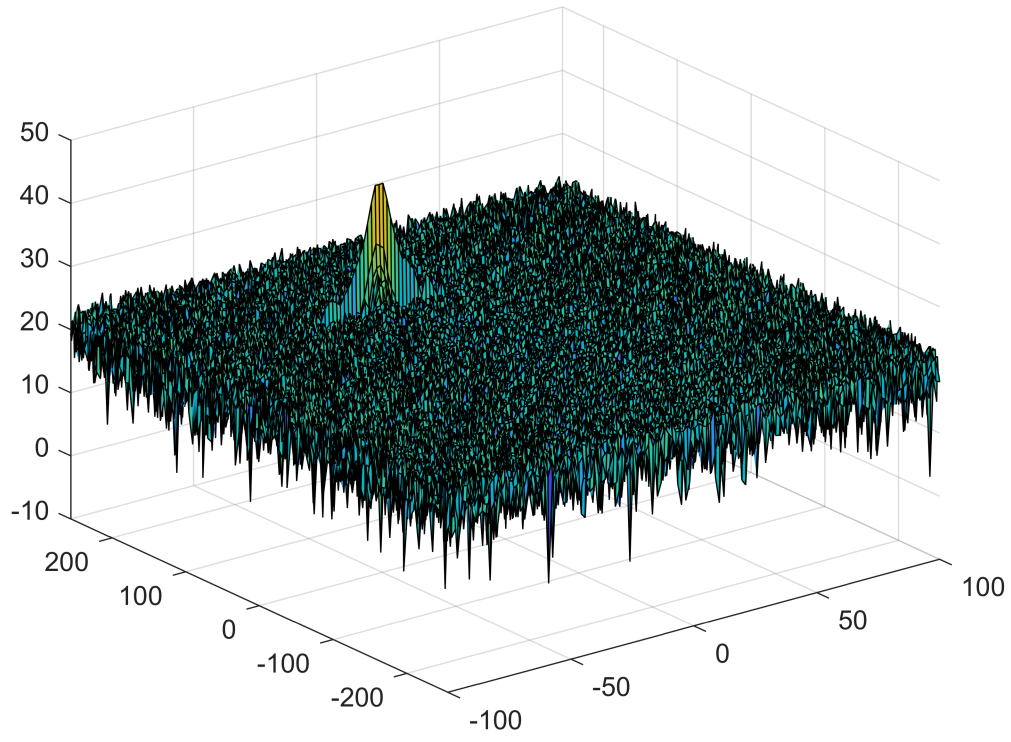
```
fft1 = fft1./Nr;

 % *%TODO* :
% Take the absolute value of FFT output
fft1 = abs(fft1);

 % *%TODO* :
% Output of FFT is double sided signal, but we are interested in only one side of the spectrum
% Hence we throw out half of the samples.
one_side_sig_fft1 = fft1(1:Nr/2);

%plotting the range
figure ('Name','Range from First FFT')
%subplot(2,1,1)
plot(one_side_sig_fft1);

 % *%TODO* :
 % plot FFT output
axis ([0 200 0 1]);
```



RANGE DOPPLER RESPONSE

```
% The 2D FFT implementation is already provided here. This will run a 2DFFT
% on the mixed signal (beat signal) output and generate a range doppler
```

```
% map.You will implement CFAR on the generated RDM

% Range Doppler Map Generation.

% The output of the 2D FFT is an image that has reponse in the range and
% doppler FFT bins. So, it is important to convert the axis from bin sizes
% to range and doppler based on their Max values.

Mix=reshape(Mix,[Nr,Nd]);

% 2D FFT using the FFT size for both dimensions.
fft2 = fft2(Mix,Nr,Nd);

% Taking just one side of signal from Range dimension.
fft2 = fft2(1:Nr/2,1:Nd);
fft2 = fftshift (fft2);
RDM = abs(fft2);
RDM = 10*log10(RDM) ;

%use the surf function to plot the output of 2DFFT and to show axis in both
%dimensions
doppler_axis = linspace(-100,100,Nd);
range_axis = linspace(-200,200,Nr/2)*((Nr/2)/400);
figure,surf(doppler_axis,range_axis,RDM);
```



CFAR implementation

```
%Slide Window through the complete Range Doppler Map

% *%TODO* :
```

Select the number of Training Cells in both the dimensions

```
Tr = 8;
Td = 8;
```

Select the number of Guard Cells in both dimensions around the Cell under test (CUT) for accurate estimation

```
Gr = 4;
Gd = 4;
```

offset the threshold by SNR value in dB

```
offset = 1.2;
```

Create a vector to store noise_level for each iteration on training cells

```
noise_level = zeros(1,1);

% *%TODO* :
```

- Design a loop such that it slides the CUT across range doppler map by giving margins at the edges for Training and Guard Cells.
- For every iteration sum the signal level within all the training cells. To sum convert the value from logarithmic to linear using db2pow function.
- Average the summed values for all of the training cells used.
- After averaging convert it back to logarithimic using pow2db.
- Further add the offset to it to determine the threshold. Next, compare the signal under CUT with this threshold. If the CUT level > threshold assign it a value of 1, else equate it to 0.

```
% Use RDM[x,y] as the matrix from the output of 2D FFT for implementing
% CFAR
RDM = RDM/max(max(RDM));

for i = Tr+Gr+1:(Nr/2)-(Gr+Tr)
    for j = Td+Gd+1:Nd-(Gd+Td)

       %reset noise level
        noise_level = zeros(1,1);

        % Calculate noise
        for l = i-(Tr+Gr) : i+(Tr+Gr)
            for m = j-(Td+Gd) : j+(Td+Gd)
                if (abs(i-l) > Gr || abs(j-m) > Gd)
```

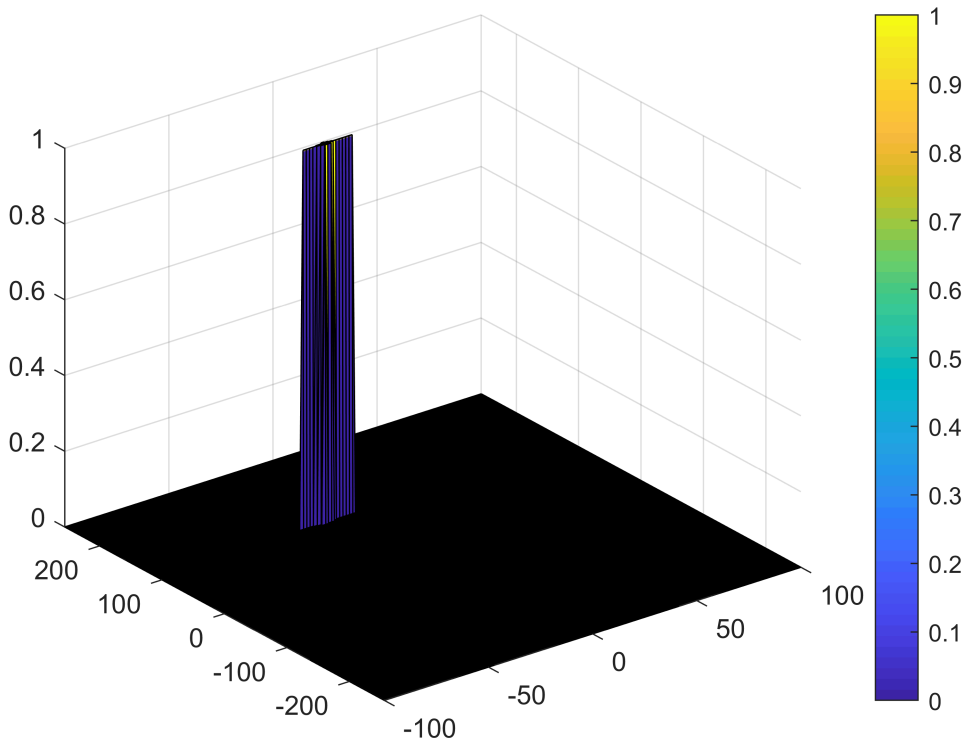
```
                    noise_level = noise_level + db2pow(RDM(l,m));
                end
            end
        end

        % Calculate threshould
        threshold = pow2db(noise_level/(2*(Td+Gd+1)*2*(Tr+Gr+1)-(Gr*Gd)-1)) + offset;

        if (RDM(i,j) < threshold)
            RDM(i,j) = 0;
        else
            RDM(i,j) = 1;
        end

    end
end

% *%TODO* :
```

The process above will generate a thresholded block, which is smaller than the Range Doppler Map as the CUT cannot be located at the edges ofmatrix. Hence,few cells will not be thresholded. To keep the map size same set those values to 0.

```
RDM(union(1:(Tr+Gr),end-(Tr+Gr-1):end),:) = 0;
RDM(:,union(1:(Td+Gd),end-(Td+Gd-1):end)) = 0;

% *%TODO* :
%
```

Display the CFAR output using the Surf function like we did for Range

```
%Doppler Response output.
figure('Name',' 2D CA-CFAR')
surf(doppler_axis,range_axis,RDM);
colorbar
```


1
0.8
0.6
0.4
0.2
0
200
100
0
-100
-200
-100
-50
0
50
100
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0