Radar Target Generation and Detection

1. Radar Specifications

- frequency = 77e9
- maxRange = 200
- rangeResolution = 1
- maxVelocity = 100
- speedOfLight = 3e8

2. User Defined Range and Velocity of target

- targetRange = 110
- targetVelocity = -20

3.FMCW Waveform Generation – Calculation of Bandwidth, Chirp Time & Slope

Calculated Bandwidth, Chirp Time and Slope of the FMCW as given below.

Bandwidth = speedOfLight/2*rangeResolution
Tchirp = 5.5*2*maxRange/speedOfLight
slope = Bandwidth/Tchirp

4. Signal generation and Moving Target simulation – Simulation Loop

For each time stamp updated the Range of the Target for constant velocity and updated the transmitted and received signal. Then beat signal was generated by mixing transmitted and received signal using matrix multiplication. Below is the code used for this process.

```
for i=1:length(t)

    % *%TODO* :
   %For each time stamp update the Range of the Target for constant velocity.
   r_t(i) = targetRange+(targetVelocity*t(i)); % distance = speed * time
   td(i) = 2*r_t(i)/speedOfLight; %twice the distnace to and fro

   % *%TODO* :
   %For each time sample we need update the transmitted and
   %received signal.
   Tx(i) = cos(2*pi*(fc*t(i)+(slope*t(i)*t(i))/2));
   Rx(i)  = cos(2*pi*(fc*(t(i)-td(i))+(slope*(t(i)-td(i))*(t(i)-td(i)))/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(i) = Tx(i)*Rx(i);
end
```

5. Implementation for Range FFT on the Beat or Mix signal and plotting the result

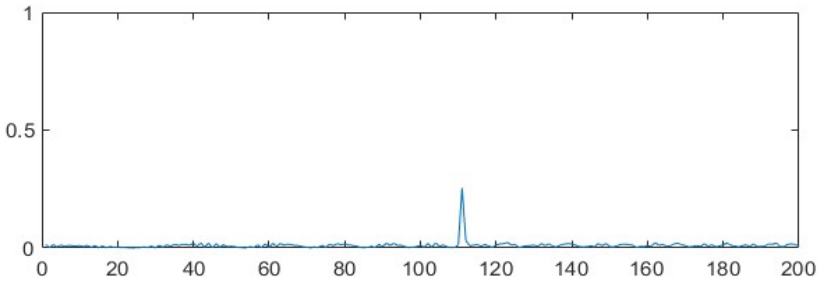
a. reshape the vector into Nr*Nd array
```
Mix = reshape(Mix, [Nr, Nd])
```

b. run the FFT on the beat signal along the range bins dimension Nr and then Normalize
```
Mix_fft = fft(Mix, Nr)
Mix_fft = Mix_fft/Nr
```

c. Take the absolute value of the FFT output and since we are interested in only one side of the spectrum, half of the samples were thrown out
```
Mix_fft = abs(Mix_fft)
Mix_fft = Mix_fft(1:Nr/2)
```

d. Plot the FFT Output
```
plot(Mix_fft)
axis ([0 200 0 1])
```

Output:



6. Implementation for 2D FFT on the Mixed signal output and generate range Doppler map

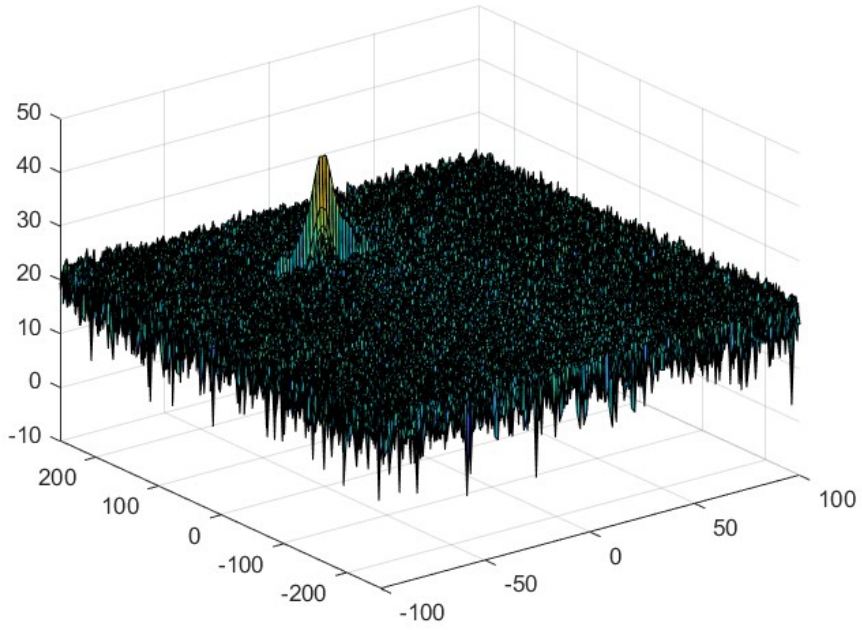
a. Convert the axis from bin sizes to range and Doppler bases on their max values
```
Mix=reshape(Mix,[Nr,Nd])
```

b. 2D FFT using the FFT size for both dimensions
    sig_fft2 = fft2(Mix,Nr,Nd)

c. Taking just one side of signal from Range dimension
    sig_fft2 = sig_fft2(1:Nr/2,1:Nd)
    sig_fft2 = fftshift (sig_fft2)
    RDM = abs(sig_fft2)
    RDM = 10*log10(RDM)

d. 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('Name','2D FFT OUTPUT'),surf(doppler_axis,range_axis,RDM)

    Output:



7. Implementation for 2D CFAR process

a. The number of Training Cells selected in both the dimensions
    Tr = 10
    Td = 8

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

Gr = 4
Gd = 4

c. offset the threshold by SNR value in dB by using the below value
offset = 14

d. Looping over the cells in both range and Doppler dimensions by leaving the margins using the start and end indices. For every iteration sum the signal level within all the training cells. To sum, converted the value from logarithmic to linear using db2pow function. Average the summed values for all of the training cells used. After averaging converted it back to logarithimic using pow2db and added the offset to it to determine the threshold. Next, compared the signal under CUT with this threshold. If the CUT level > threshold assigned it a value of 1, else equated it to 0.

```
for i = Tr+Gr+1:(Nr/2)-(Gr+Tr)
   for j = Td+Gd+1:Nd-(Gd+Td)
      %Create a vector to store noise_level for each iteration on training cells
      noise_level = zeros(1,1);

      % Calculate noise SUM in the area around CUT
     for p = i-(Tr+Gr) : i+(Tr+Gr)
        for q = j-(Td+Gd) : j+(Td+Gd)
           if (abs(i-p) > Gr || abs(j-q) > Gd)
              noise_level = noise_level + db2pow(RDM(p,q));
           end
        end
      end
%  Calculate threshould from noise average then add the offset
      threshold = offset+pow2db(noise_level/(2*(Td+Gd+1)*2*(Tr+Gr+1)-(Gr*Gd)-1));

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

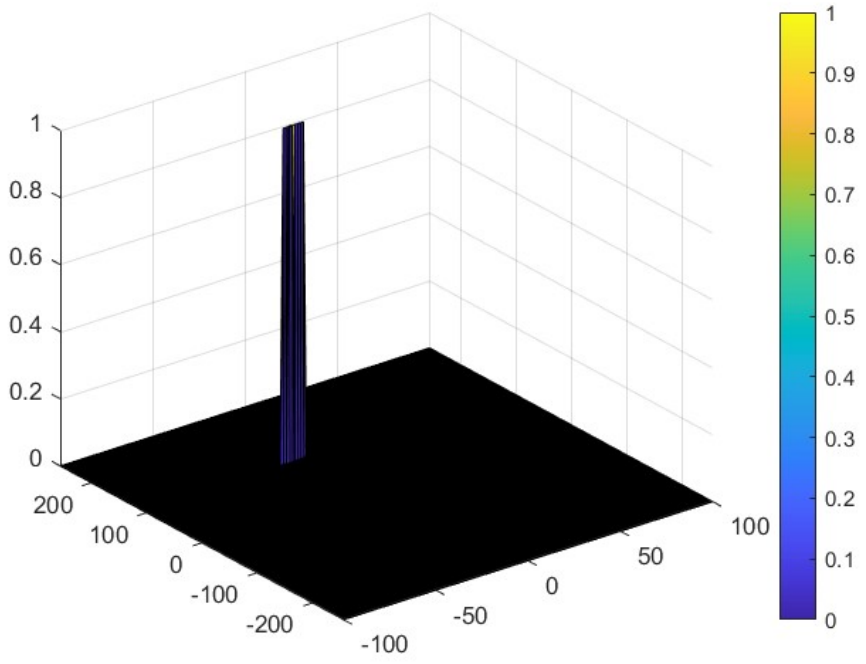
e. The above generated thresholded block is smaller than the Range Doppler Map as the CUT cannot be located at the edges of matrix. Hence, few cells not thresholded set to) to keep the map size same.

```
RDM(RDM~=0 & RDM~=1) = 0
```

f. Displayed the CFAR output using the Surf function

```
figure('Name','2D CFAR OUTPUT'),surf(doppler_axis,range_axis,RDM)
colorbar
```

Output :


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