Report Radar Target Generation and Detection

1. Range and Velocity of target and radar specifications

```
Max_Range=200;
Max_Velocity=100;
Range_Resolution=1;
c=3e8;

Range_target=110;
Velocity_target=50;
```

2. FMCW Waveform

- Calculate the Bandwidth (B), Chirp Time (Tc) and Slope (slope) of the FMCW chirp using the requirements above and carrier frequency of Radar
- Design the FMCW waveform by giving the specs of each of its parameters.

```
fc=77e9;
sweep_time=5.5;
B=c/(2*Range_Resolution);
Tc=(sweep_time*2*Max_Range)/c;
slope=B/Tc;
Nd=128;
Nr=1024;
t=linspace(0,Nd*Tc,Nr*Nd);
```

- Creating the vectors Mix, Tx, Rx rely on the total samples

```
Tx=zeros(1,length(t));
Rx=zeros(1,length(t));
Mix=zeros(1,length(t));
```

- Similar vectors for range covered and time delay.

```
rt=zeros(1,length(t));
td=zeros(1,length(t));
```

3. Generate signal and target simulation

```
for i=1:length(t)
    rt(i) = Range_target + (Velocity_target*t(i));
    td(i) = (2*rt(i)) / c;
    Tx(i) = cos( 2*pi*( fc*(t(i)) + ( 0.5 * slope * t(i)*t(i)))
);
    Rx(i) = cos( 2*pi*( fc*(t(i)-td(i)) + ( 0.5 * slope *
(t(i)-td(i))^2) ) );
    Mix(i) = Tx(i).*Rx(i);

End
```

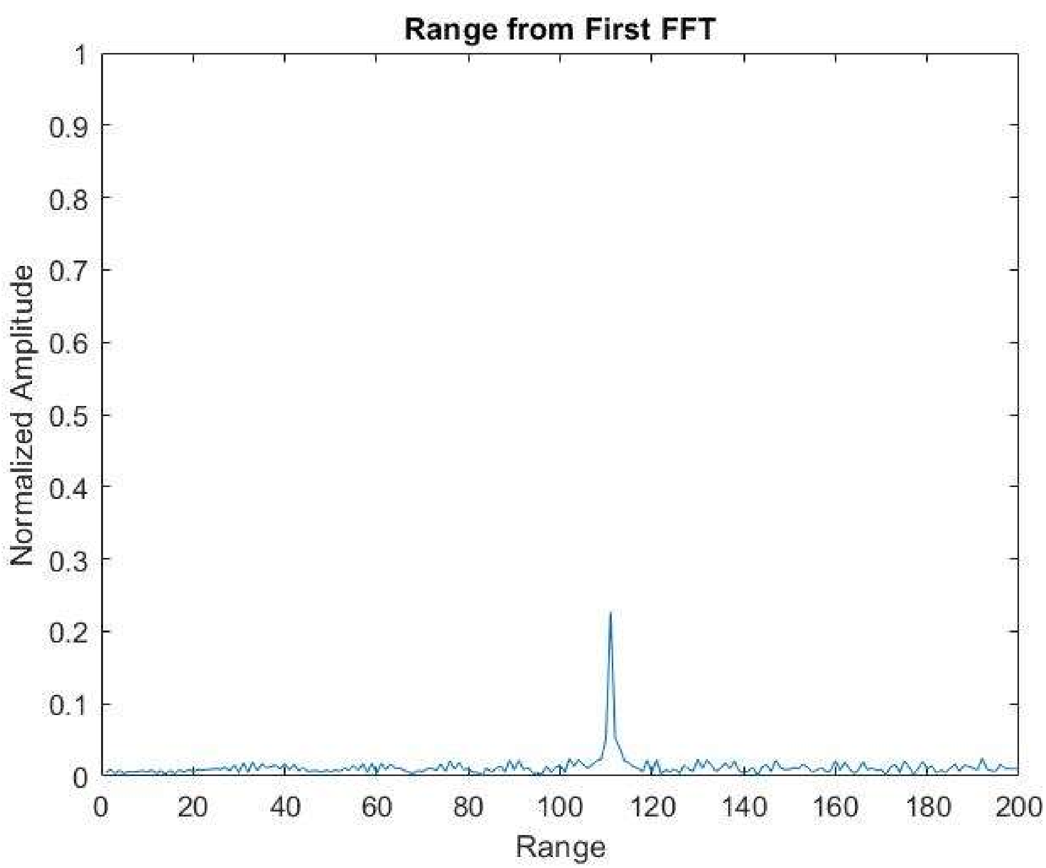
4. Range Measurement

- Reshape the vector into Nr*Nd array. Nr and Nd would also define the size of range and Doppler FFT respectively.
- Run the FFT on the beat signal along the range bins dimension (Nr) and normalize.
- Take the absolute value of FFT output. Output of FFT is double sided signal, but we are interested in only one side of the spectrum.
- We keep the half of the samples. Plotting the range, plot FFT output

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Mix=reshape(Mix,[Nr,Nd]);
sig_fft1=fft(Mix,Nr);
sig_fft1=sig_fft1./Nr;
sig_fft1=abs(sig_fft1);

single_fft1=sig_fft1(1:Nr/2);
figure('Name','Range From FFT');
plot(single_fft1);
axis([0 200 0 1]);
title('Range from First FFT');
ylabel('Normalized Amplitude');
xlabel('Range');
```

Simulation Result



5. Range Doppler response

- The 2D FFT implementation is already provided here.

- This will run a 2D-FFT on the mixed signal (beat signal) output and generate a range doppler map.
- Implement CFAR on the generated RDM Range Doppler Map Generation.
- The output of the 2D FFT is an image that has response 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
- 2D FFT using the FFT size for both dimensions. Taking one side of signal from Range dimension.

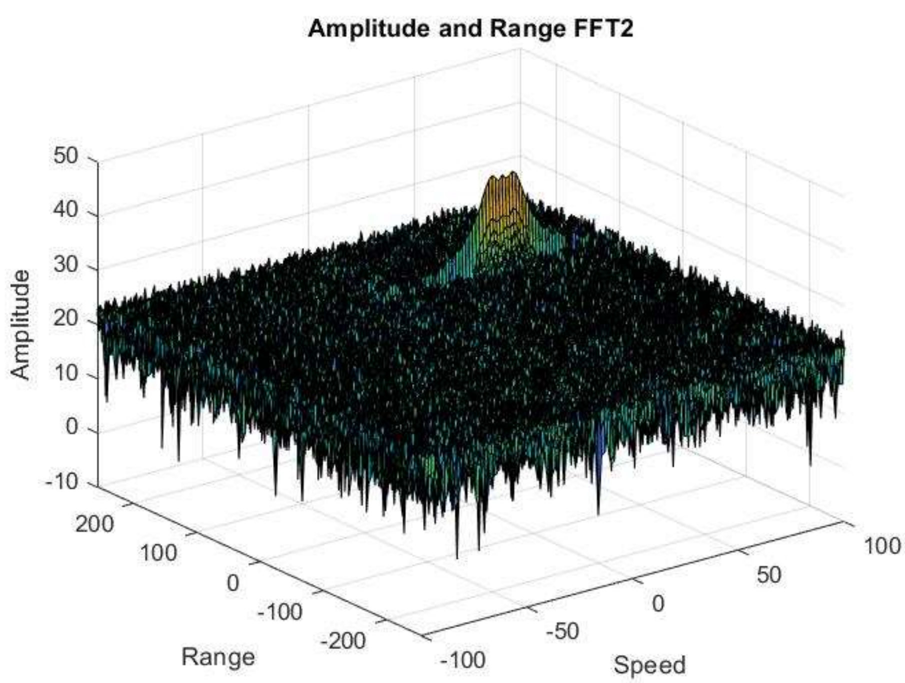
```
Mix=reshape(Mix,[Nr,Nd]);
sig_fft2=fft2(Mix,Nr,Nd);
sig_fft2=sig_fft2(1:Nr/2,1:Nd);
sig_fft2=fftshift(sig_fft2);

rdm=abs(sig_fft2);
rdm=10*log10(rdm);
```

- Use the surf function to plot the output of 2D-FFT and to show axis in both dimensions

```
dopller=linspace(-100,100,Nd);
range_axis=linspace(-200,200,Nr/2)*((Nr/2)/400);
figure('Name','Range from FFT2')
surf(dopller,range_axis,rdm);
title('Amplitude and Range From FFT2');
xlabel('Speed');
ylabel('Range');
zlabel('Amplitude');
```

Simulation Result



6. CFAR implementation

- Slide Window through the complete Range Doppler Map
- Select the number of Training Cells in both the dimensions.
- Select the number of Guard Cells in both dimensions around the Cell under test (CUT) for accurate estimation
- Offset the threshold by SNR value in dB

  Tr=10; Td=8; Gr=4; Gd=4; offset=1.4;

- Create a vector to store noise_level for each iteration on training cells 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)
        noise=zeros(1,1);
        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=noise+db2pow(rdm(p,q));
                end
            end
        end

        threshold=pow2db(noise/(2*(Td+Gd+1)*2*(Tr+Gr+1)-
(Gr*Gd)-1));
        threshold=threshold+offset;
        CUT=rdm(i,j);
        if(CUT<threshold)
            rdm(i,j)=0;
        else
            rdm(i,j)=1;
        end
    end
end
```

- The process above will generate a threshold block, which is smaller than the Range Doppler Map as the CUT cannot be located at the edges of matrix.
- Few cells will not be thresholded.
- To keep the map size same set those values to 0.
- Display the CFAR output using the Surf function
- Doppler Response output.

```
rdm(1:(Tr+Gr), :) = 0;
rdm(end-Tr-Gr:end, :) = 0;
rdm(:, 1:(Td+Gd)) = 0;
rdm(:, end-Td-Gd:end) = 0;
figure('Name','CA-CFAR RDM');
surf(doplller,range_axis,rdm);
colorbar;
title( 'CA-CFAR RDM');
xlabel('Speed');
ylabel('Range');
zlabel('Normalized Amplitude');
view(315, 45);
```

Simulation Result