PROJECT SPECIFICATION
Radar Target Generation and Detection

1. Given Radar Specifications -

```
Max Range = 200m
Range Resolution = 1 m
Max Velocity = 100 m/s
speed of light = 3e8
```

2. User Defined Range and Velocity of target -

```
Rtarget = 100;(in m)
Vtarget = 50;(in m/s)
```

3. FMCW Waveform Design
3.1 Calculating FMCW chirp parameters- Bandwidth, Tchirp, Slope of the chirp

```
Bsweep = c/(2*Rr); %Bandwidth for each chirp at given resolution
Tchirp  = 5.5*2*Rmax/c; %Chirp time for each chirp
slope = Bsweep/Tchirp; %Slope of the chirp signal
```

4. Modeling Signal propogation for moving targets - Simulation Loop
Includes defining the transmitted signal, recieved signal and creating a mix signal which provides details for range and doppler frequency shift (beat frequency).

```
for i=1:length(t)
    % *%TODO* :
    %For each time stamp update the Range of the Target for constant velocity.
    r_t(i) = Rtarget+(Vtarget*t(i));% distance = speed * time
    td(i) = 2*r_t(i)/c; %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. Range FFT / 1-D FFT - Determine the FFT of the mixed signal calculated over range. For this the Nr i.e. range bins dimension is considered.

```
 % *%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.
Mix_fft = fft(Mix,Nr);
 % *%TODO* :
% Take the absolute value of FFT output and %normalize
Mix_fft = abs(Mix_fft/Nr);
 % *%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.
Mix_fft = Mix_fft(1:(Nr/2)+1);
```

Result

6. 2D FFT - Doppler FFT - Determine the 2D-FFT of the mixed signal calculated along range-velocity grids were used to generate range-velocity-map.

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

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

% 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) ;% Linear to logarithmic
```

Result:

7. 2D - CFAR - The above Range Doppler signal contains a lot of noise which is called as clutters, Those clutters could create false alarms. Thus, it is very important to suppress the same. Thus we implement a 2D CFAR on the above signal.

7.1 Selected TRaining cells:

```
Tr = 10; Td = 8;
```

7.2 Selected Guard cells around the CUT:

```
Gr = 5;  Gd = 5;
```

7.3 Selected Offset in dB

```
Offset = 6;
```

7.4 2D-CFAR implementation and Steps taken to suppress the non-thresholded cells at the edges:

```
trainingcells = (2*Tr+2*Gr+1)*(2*Td+2*Gd+1) - (2*Gr+1)*(2*Gd+1);%total training cells

for i = 1:(Nr/2 - (2*Gr+2*Tr))%outerloop of range
   for j = 1:(Nd - (2*Gd+2*Td))%inner loops of doppler

       %sig1-sig2 is done to only calculate training cells noise signal
       %(Tr and Td)- note we just considering the leading training 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.

       sig1 = sum(db2pow(RDM(i:i+2*Tr+2*Gr, j:j+2*Td+2*Gd)),'all');
       sig2 = sum(db2pow(RDM(i+Tr:i+Tr+2*Gr, j+Td:j+Td+2*Gd)),'all');
       noise_level = sig1 - sig2;

       %Average the summed values for all of the training cells used for this - convert back to
the logarithmic using pow2db function.

       threshold = pow2db(noise_level/trainingcells);

       %Further add the offset to it to determine the threshold.
       threshold = threshold + Offset;
       %Measuring the signal within the CUT
       signal = RDM(i+Tr+Gr, j+Td+Gd);
```

```
        % Filter the signal above the threshold for that -  compare the signal under CUT with this
threshold. If the CUT level > threshold assign it a value of 1.          % *%TODO* :

        if (db2pow(signal) > db2pow(threshold))
            signal = 1;
        else
            signal = 0;
        end

            % 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 of matrix. Hence,few cells will
not be thresholded. To keep the map size same set those values to 0.

         if(signal ~= 0 & signal ~= 1)
             signal = 0;
         end

        signal_cfar(i+Tr+Gr,j+Td+Gd) = signal;

    end
end
```

Result: