Homework 1a: Radar Imaging

Prof. Andrea Virgilio Monti Guarnieri & Dr. Marco Manzoni

Amini 10707670

Group members: Vahid Rajabi, Hiva Amiri, Kimia Kiyan, Thibault Fievez

2020

MIMO Radar - Direction of Arrival (DoA) estimation

1-Introduction

Multiple Input Multiple Output (MIMO) radar employs multiple antennas in both transmission and reception. Employing the virtual array, MIMO radar can have a great degree of freedom. In this homework a MIMO radar is design to be able to estimate the Direction of Arrivals (DoA) of a specific target in the field of view (FoV). Two method could be used to estimate the DoV: FFT based method, and Backprojection method.

2-Designing the MIMO radar

In this section, designing the MIMO radar is shown step by step.

2-1 Designing the equivalent virtual array

Considering the Phase Center Approximation (PCA), it could be seen that we can form MN virtual array, using just M+N physical antennas. Maximum frequency should be known to be able to design the virtual array. Generally, maximum frequency is equal to $2/\lambda$. In the following case, which we limit x between -35, 35 and y between 100, 150, the angle would be between $(-\theta max, \theta max)$ where θmax can be obtained geometrically as follow:

$$\theta max = \tan^{-1}(x/y) = \tan^{-1}(35/100) = 0.3367 rad = 19.29 \text{ degree}$$

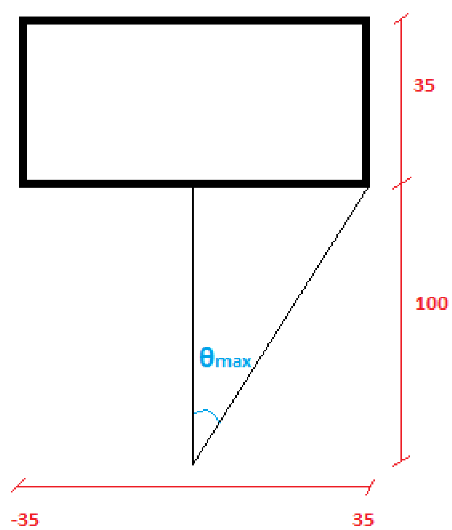

Figure 1: FoV and the Maximum ϑ

The maximum frequency could be calculated as below:

$$f_{max} = \frac{2 \times \sin(\theta_{max})}{\lambda}$$

Also, we can calculate the distance between two consecutive virtual antennas using the following formula:

$$dx = \frac{1}{f_{max}} = \frac{\lambda}{2 \times \sin(\theta_{max})}$$

Considering the resolution of 2m, the minimum number of antennas could be obtained as bellow:

$$\rho x = R \times \rho\theta$$

where $\rho\theta = \lambda/2L$ and $L = N_{tot} \times dx$ and finally:

$$\rho_x = R \times \frac{\lambda}{2 \times N_{tot} \times dx}$$

For this homework, ρx should equal or greater than 2, so ρx >2. R is the maximum distance from the radar which could be calculated by:

$R = \sqrt{(35^2+150^2)} = =154.0292$

So λ could be calculate as follow:

$$\lambda = \frac{c}{f} = \frac{3\times10^8}{9.6\times10^9} = 0.0313$$

$$dx = \frac{\lambda}{2 \times \sin(\theta_{max})} = \frac{0.313}{2 \times \sin(19.29)} = 0.047298259152717$$

$$\rho_x > 154.0292 \times \frac{0.0313}{2N_{tot} \times 0.047298259152717}$$

Finally, $N_{tot} > 50.8836$.

Round up Ntot to 51 which is the 17×3=51 so Ntx = 3, Nrx= 17.

2-2 Designing the Multiple Input Multiple Output Radar

We should locate the transmitters and receivers in a way that the positions of the corresponding virtual antennas have equal distance to dx. It is easy to see that is tx1 is located in x1 and rx1 is located in x2, their corresponding virtual antenna is in the position of x1+x2. Therefore, the distances between the receivers are dx and the distance between transmitters are Nrx × dx.
The virtual antennas must be located symmetrically according to the FoV, so the real transmitters and receivers must be symmetrical in (0,0). Receiver positions and transmitter positions are shown in the figure.

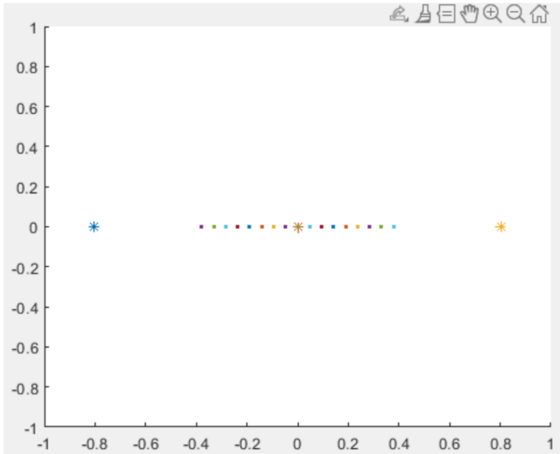

Figure 2: Dots represents the receivers, stars represent the transmitters

The corresponding virtual antenna is symmetrically placed with respect to the (0,0) point, with the distance of dx which is shown in the figure below.

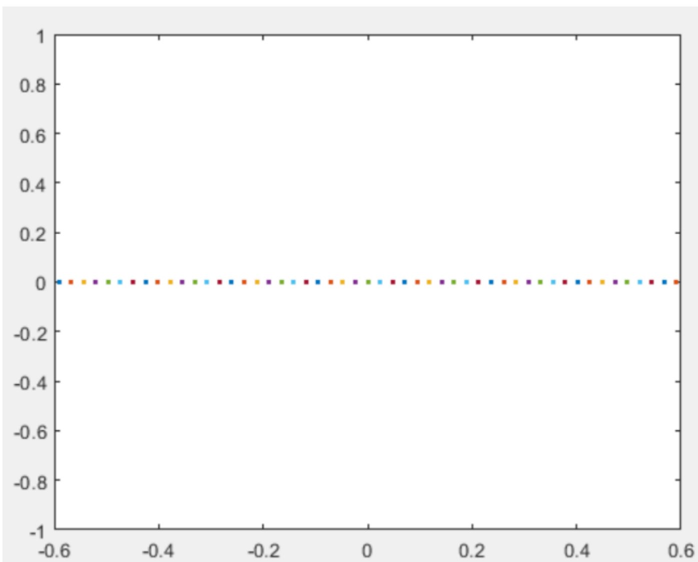

Figure 3Position of virtual antennas

The code related to this part could be seen in the below:

```
c = 3*10^8; % light speed
f = 9.6*10^9; % carrier frequency
lambda = c / f;
number_of_tx = 3;
number_of_rx = 17;
max_theta = atan(0.35);
dx = lambda/(2*sin(max_theta));
dx_rx = dx ; % distance between tow consecutive receivers
dx_tx =number_of_rx*dx_rx ; % distance between two consecutive transmitters
mode_tx = mod(number_of_tx,2);
mode_rx = mod(number_of_rx,2);
if mode_rx == 0
xn = dx_rx*((number_of_rx+1)/2)
rx_position = (-xn+dx_rx : dx_rx : xn-dx_rx );
else
xn = ((number_of_rx-1)/2)*dx_rx;
rx_position = (-xn : dx_rx : xn);
end
if mode_tx == 0
xn = dx_tx*((number_of_tx+1)/2);
tx_position = (-xn+dx_tx : dx_tx : xn-dx_tx );
else
xn = ((number_of_tx-1)/2)*dx_tx;
tx_position = (-xn : dx_tx : xn);
end
```

3-DoA estimation using FFT

In this section, DoA is done for single target and multiple targets.

3.1 Single target without noise

The easiest way to estimate the DoA is Fourier Transform.

In this section, firstly, a single target is located in a random position in the FOV which was defined in previous section. The code related to this part could be seen in the figure.

```
%% part 3.1.1 : place the target in a random position
xBox= [-35,-35,35,35,-35];
yBox= [100,150,150,100,100];
figure(1)
plot(xBox,yBox)
hold on

target_x1 = -35+70*rand; %In general, you can generate N random numbers in the
interval (a,b) with the formula r = a + (b-a).*rand(N,1).
target_y1 = 100+50*rand;
plot (target_x1, target_y1, '^' )
xlim ([-80 80])
ylim ([-10 160])
```

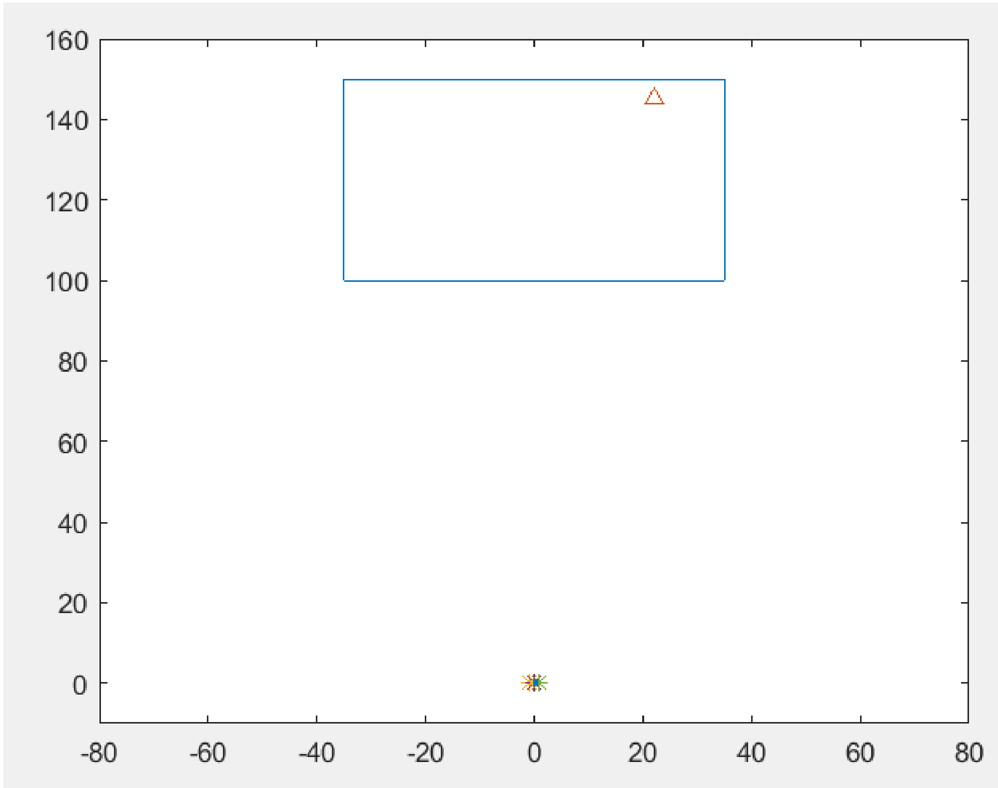

Figure 4The target in a random position

Secondly, Tx and Rx elements are placed in the 2D space by determining their coordinates.

```
%% part 3.1.2 placing TX and RX elements in 2D space

mode_tx = mod(number_of_tx,2);
mode_rx = mod(number_of_rx,2);

if mode_rx == 0
    xn = dx_rx*((number_of_rx+1)/2)
    rx_position = (-xn+dx_rx : dx_rx : xn-dx_rx );

else
    xn = ((number_of_rx-1)/2)*dx_rx;
    rx_position = (-xn : dx_rx : xn);
   % rx_position = rx_position + dx_rx/2;

end


if mode_tx == 0
    xn = dx_tx*((number_of_tx+1)/2);
    tx_position = (-xn+dx_tx : dx_tx : xn-dx_tx );

else
   xn = ((number_of_tx-1)/2)*dx_tx;
    tx_position = (-xn : dx_tx : xn);
end

tx_position;
rx_position;

 plot(tx_position,0,'*')
 plot(rx_position,0,'.')
hold off
```

Thirdly, the angle of arrival is computed using geometry.

```
%% part 3.1.3 : theta1 calculation

theta1 =  rad2deg(atan(target_x1 / target_y1))
```

At the next step, the signal is propagated from the first transmitter to all the Rx receiver elements using the following equation.

$$s_m^n = \frac{\rho_p}{R_0 + R_1} e^{-j\frac{2\pi}{\lambda}(R_0+R_1)}$$

And for all of the other transmitters. The code is shown here:

```
%% 3.1.4 : propagating the signal

R0_1 = zeros(1,number_of_tx);
R1_1= zeros(1,number_of_rx);
for m = 1:1:number_of_tx
   R0_1(1,m) = sqrt((target_x1 -(tx_position(1,m)))^2 + target_y1^2); % calculating R1
and store it in an array
end

for n = 1:1:number_of_rx
     R1_1(1,n) = sqrt((target_x1 -rx_position(1,n))^2 + target_y1^2);% calculating R2
and store it in an array
end
w=rand;
phi=rand*2*pi;
row_p = w*exp(-j*phi) ; ;
Smn1 = zeros(number_of_tx, number_of_rx);
R1 = zeros(1,number_of_tx*number_of_rx);
temp = 1;
for m = 1:1:number_of_tx
   for n = 1:1: number_of_rx
       R1(1,temp) =  R0_1(1,m)+R1_1(1,n);
       temp = temp + 1;
   end
end
received_vector1 = (row_p./R1).*exp(-(j*2*pi.*R1)/lambda); %% received signal stored
in an array
```

The next step is dedicated to calculate the FFT using fft() function. But in order to have more accuracy and better visualization, zero padding and shift are done. The code is shown below:

```
%% part 3.1.5 fft calculation

max_theta =(atan(0.35));
fs = 4 * sin(max_theta) / lambda ; % sampling frequency
dx = 1/fs ;
sample_number = 2000*length(received_vector1);% zero padding
ff1 = fftshift(fft(received_vector1,sample_number)); % do the fft shift it on the
center

power1 = abs(ff1).^2 / sample_number; % calculating the PSD
x = (-fs/2 : fs/sample_number : fs/2-fs/sample_number);
figure(2)
plot(x , power1) % plot the PSD
```

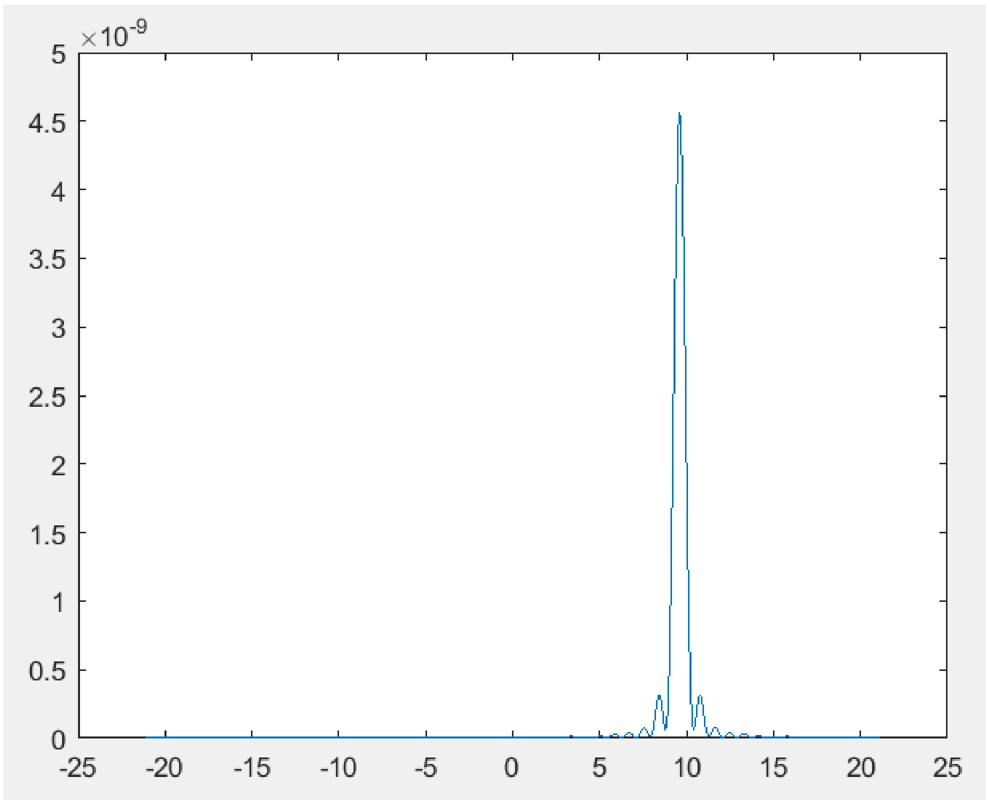

Figure 5: The received signal PSD

Finally, in the 6th step, the frequency is converted in angles and the estimated angle of arrival is calculated.

```
%% part 3.1.6 angle estimation
[M1,I1] = max(power1);
fx1 = -fs/2 + I1*fs/sample_number;% calculating the fx of the maxmumim
sine_theta1 = fx1 * lambda / (2) ;
estimated_theta1 = rad2deg(asin(sine_theta1))% estimate the theta
```

theta1 =   8.6222

estimated_theta1 =   8.6223

3.2 Validation of the nominal resolution - Multiple targets

Repeating the previous section with two targets is done in this step. Receiving signals from two random targets, DOV estimated and the result could be seen at the following figures:

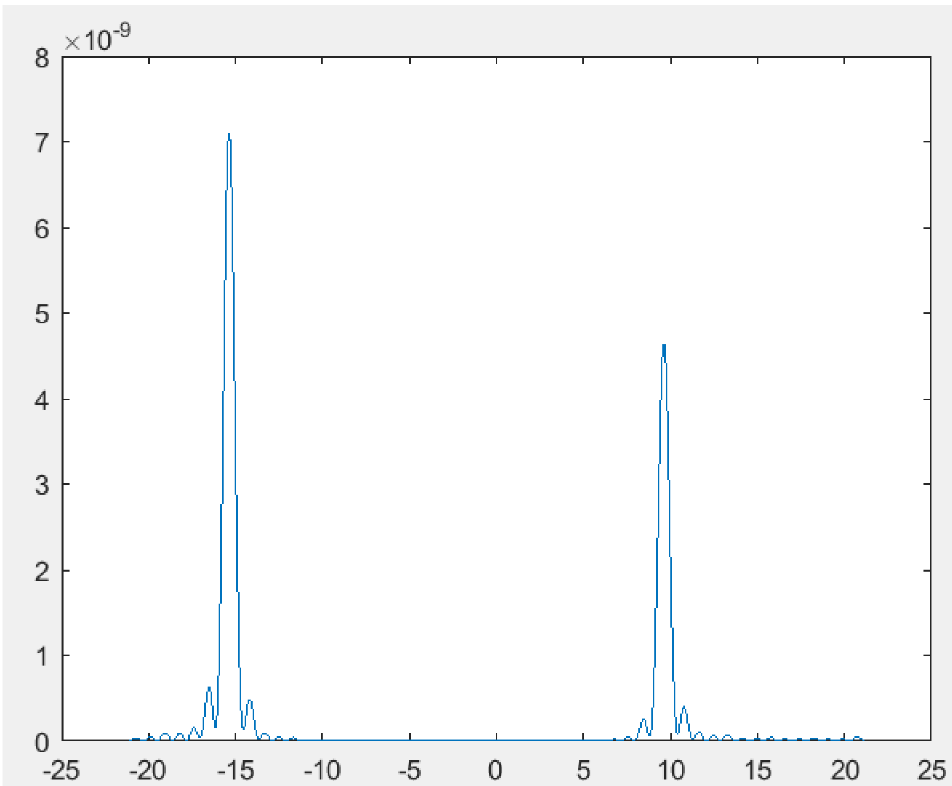

Figure 6: PSD of two random points

two_obj_theta =   8.6222 -13.8899

two_object_estimated_theta = -13.9001   8.6392

The code of this part could be seen below:

```matlab
%% part 3.2 tow object detection

xBox= [-35,-35,35,35,-35];
yBox= [100,150,150,100,100];
figure(3)
 plot(xBox,yBox)
 hold on
a=rand;
target_x2 = -35+70*rand; %In general, you can generate N random numbers in the interval (a,b) 
with the formula r = a + (b-a).*rand(N,1).
target_y2 = 100+50*rand;
plot (target_x1, target_y1, '+' )
plot (target_x2, target_y2, '*' )
xlim ([- 80 80])
ylim ([-10 150])

hold off

theta2 =  rad2deg(atan(target_x2 / target_y2));


R0_2 = zeros(1,number_of_tx);
R1_2= zeros(1,number_of_rx);
for m = 1:1:number_of_tx
   R0_2(1,m) = sqrt((target_x2 -(tx_position(1,m)))^2 + target_y2^2); % calculate the R0
end

for n = 1:1:number_of_rx
     R1_2(1,n) = sqrt((target_x2 -rx_position(1,n))^2 + target_y2^2); % calculate the R2
end
R2 = zeros(1,number_of_tx*number_of_rx);
temp = 1;
for m = 1:1:number_of_tx
   for n = 1:1: number_of_rx
       R2(1,temp) =  R0_2(1,m)+R1_2(1,n);
       temp = temp + 1;
   end
end

received_vector2 = (row_p./R2).*exp(-(j*2*pi.*R2)/lambda);%

ff2 = fftshift(fft(received_vector2,sample_number));
power2 = abs(ff2.^2)/sample_number;
x = (-fs/2 : fs/sample_number : fs/2-fs/sample_number);
figure(4);plot(x,power2)

received_vector = received_vector1 + received_vector2; % received signal of 2 object
ff = fftshift(fft(received_vector,sample_number));
power = abs(ff.^2)/sample_number;
figure(5)
plot(x,power)
% finding 2 peaks and their cooresponding fx
[local_peaks location] = findpeaks(power);

[max1 , index1] = max(local_peaks);
location1 = location(index1);
local_peaks(index1) = [];
location(index1) = [];

[max2 , index2] = max(local_peaks);

location2 = location(index2);
fx_1 = -fs/2 + location1*fs/sample_number;
sine_theta_1 = fx_1 * lambda / (2) ;
estimated_theta_1 = rad2deg(asin(sine_theta_1));

fx_2 = -fs/2 + location2*fs/sample_number;
sine_theta_2 = fx_2 * lambda / (2) ;
estimated_theta_2 = rad2deg(asin(sine_theta_2));
two_obj_theta = [theta1 theta2]
two_object_estimated_theta = [  estimated_theta_1 estimated_theta_2]
```

4- DoA with Backprojection

Another way to estimate the direction of arrival is back projection method. The received signal is back projected into a FOV which is defined before. The meshgrid is used to divide the FoV into pixels within the area of $dx \times dx$. Taking advantage of imagesec, the calculated formula was plotted shown in the following figure. As it is shown in figure below, the maximum of the signal is in the direction of FoV.

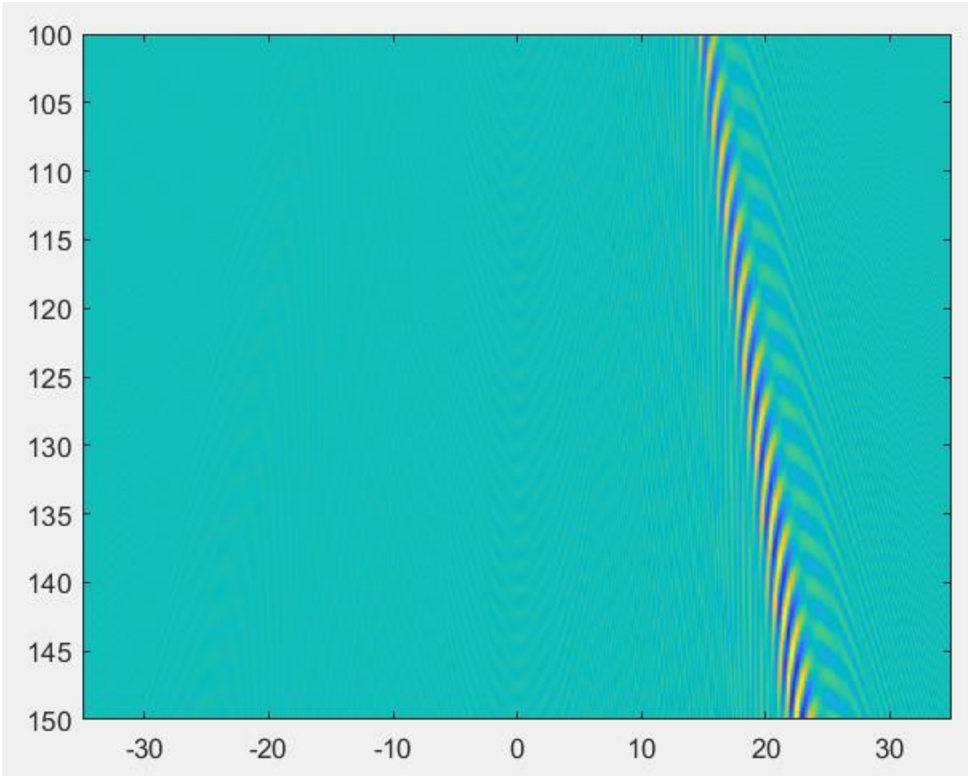

Figure 7: *Backprojection signals (FoV)*

The code of implementing back projection is shown below. Also, the estimated angle could be seen with this method.

```matlab
%%Back projection

x_mesh = -35 :  dx : 35;
y_mesh = 100 : dx : 150;
[x,y] = meshgrid(x_mesh,y_mesh);% creat the pixel
Ntot = number_of_rx * number_of_tx;

if mode(Ntot,2) == 0
    xn = dx*((Ntot+1)/2)
    virtual_antenna_position = (-xn+dx : dx : xn-dx );

else
    xn = ((Ntot-1)/2)*dx;
    virtual_antenna_position = (-xn : dx : xn);

end


Sp = zeros(length(y_mesh),length(x_mesh));

for l = 1:1:length(received_vector1)
    D=sqrt((x-virtual_antenna_position(l)).^2+(y).^2);
    S = received_vector1(1,l)*D.*exp(j*4*pi*D/lambda); % propagate the signal of each
virtual antena
    Sp = Sp + S ; % sum the signals propagated by each virtual antennas

end
figure(6)
imagesc(x_mesh,y_mesh,real(Sp))


K = abs(Sp);
maximum = max(max(K));
[alfa,beta]=find(K==maximum);
rad2deg(atan(x(alfa,beta)/y(alfa,beta))) % estimate the theta
```

Estimated Theta in Backprojection case:

ans =

8.6249