

Wireless Communications

Project for the exam

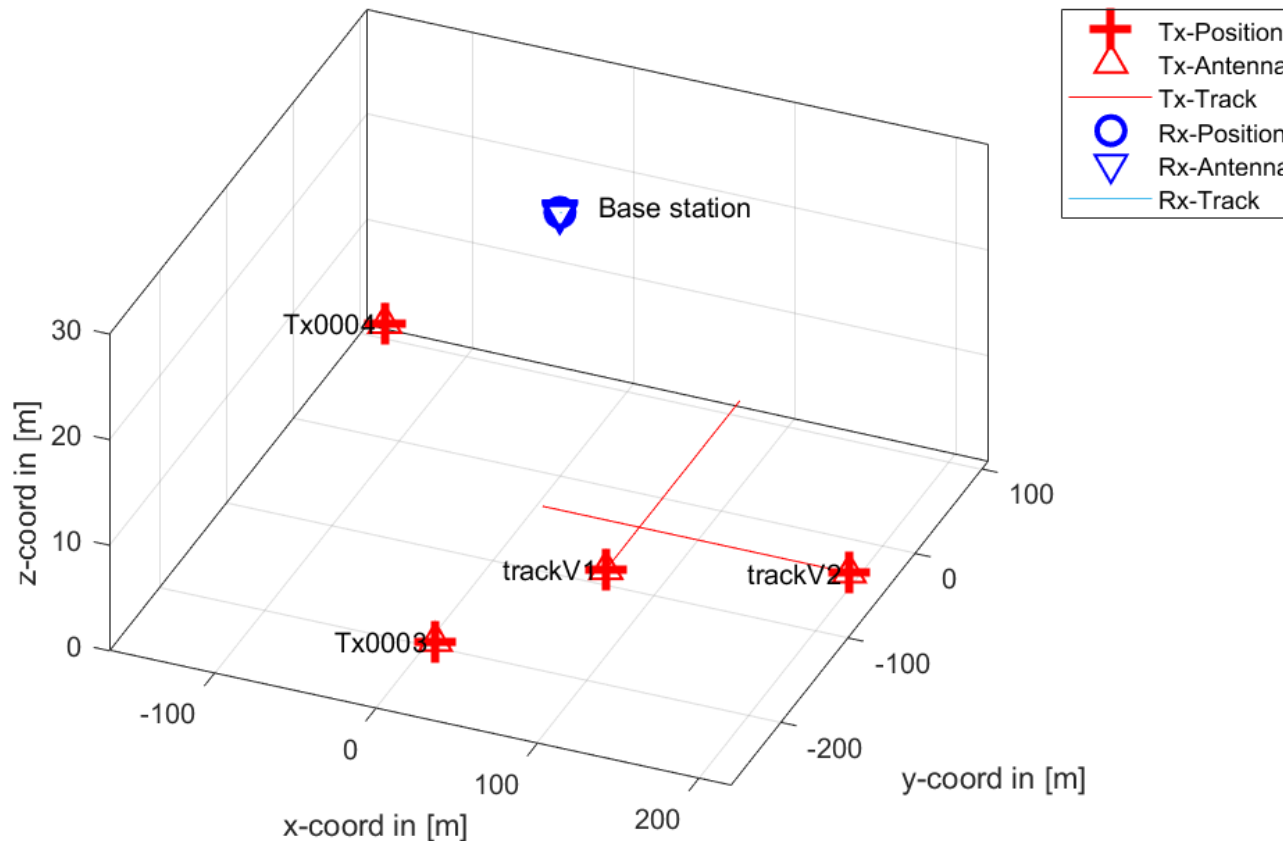
Lab Assistant: Davide Scazzoli
(davide.scazzoli@polimi.it)



POLITECNICO
MILANO 1863

Course Project

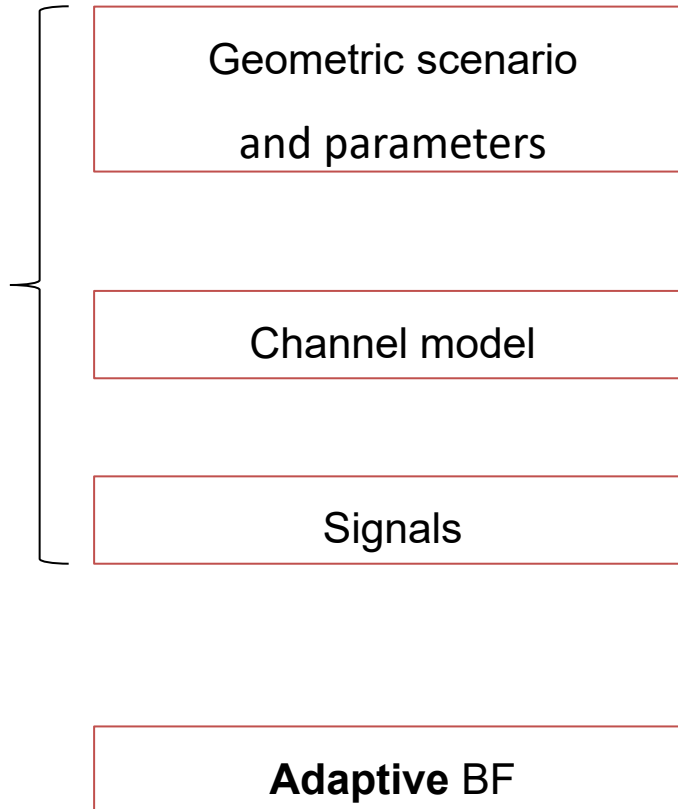
Simulate an environment with a BS equipped with an antenna array (rectangular)



The idea is to track, with adaptive BF, two mobile terminals in presence of one or more interferers.

Course Project

Project blocks



Main options

2D / 3D


LoS / 3GPP / ...

NB / OFDM / 3GPP

DoA based / LMS / ...

Course Project

Project results and levels (D is the minimum level to be achieved)



Levels	Geometry	Channel	Signal	BF
D (2 pnt)	2D	LoS, free space	Narrow-band	DoA based
C (3 pnt)	3D	LoS, free space	OFDM	DoA based
B (5 pnt)	3D	LoS + simplified multipath	OFDM	LMS (time or frequency domain)
A (6 pnt)	3D	3GPP	OFDM / 3GPP / NR (> 6 GHz)	LMS (time and frequency domain)

- Any feature in the code that is parameterized (e.g. the code works for any number of interferers, terminals, etc ...) constitutes an added value for the project evaluation.
- The project can be done in a team (till to 3 persons) but the exam discussion of the project is always personal.
- I expect a final brief report or presentation + commented (Matlab or Simulink) code.
- The type of numerical results in the report and the related comments are crucial for the project evaluation.
- If a team divides the tasks, please specify how you have organized and divided the activity in the final report.

Scenario and Channel



Outline

- Getting started, defining a geometry of the problem
- Simple LOS channel
- Two Ray Ground Reflection Channel
- Quadriga Channel Model
- F.A.Q.



Defining a Geometry

- Simply define some relevant parameters and object positions as, for example, vectors of xyz coordinates

```
Pars.fc=1e9; %Carrier frequency
Pars.c = physconst('LightSpeed');
Pars.lambda = Pars.c/Pars.fc;
%Define geometry of the problem (xyz coordinates)
Geometry.BSPos=[0,0,25]; % 25m is a typical height for a macrocell BS
Geometry.V1PosStart=[70,-100,1.5]; %Start position fo Vehicle 1
Geometry.V1PosEnd=[70,100,1.5]; %End position for vehicle 1
Geometry.V2PosStart=[200,-50,1.5];
Geometry.V2PosEnd=[10,-50,1.5];
Geometry.I1Pos=[10,-210,1.5];
Geometry.I2Pos=[-150,100,1.5];
```

- Putting all values in a struct is not required, but convenient if you want to easily pass them to functions:

```
CreateScenarioAndVisualize(Geometry, Pars);
```

Defining a Geometry

- From xyz coordinates we can extract distances and direction of arrival using simple trigonometry:

```
% Calculate distances as sqrt((x1-x2)^2+(y1-y2)^2)
Geometry.T1=sqrt(sum((Geometry.V1PosEnd(1,1:2) -...
    Geometry.V1PosStart(1,1:2)).^2));
Geometry.T2=sqrt(sum((Geometry.V2PosEnd(1,1:2) -...
    Geometry.V2PosStart(1,1:2)).^2));
Geometry.DistV1Start=sqrt(sum((Geometry.V1PosStart(1,1:2) - ...
    Geometry.BSPos(1,1:2)).^2));
Geometry.DistV2Start=sqrt(sum((Geometry.V2PosStart(1,1:2) - ...
    Geometry.BSPos(1,1:2)).^2));

%Calculate DoA for V1 at start
Geometry.AOAV1Start=atan2(Geometry.BSPos(1,2) - Geometry.V1PosStart(1,2), ...
    Geometry.BSPos(1,1) - Geometry.V1PosStart(1,1))*180/pi;
Geometry.ZOAV1Start=atan2(Geometry.DistV1Start, ...
    Geometry.BSPos(1,3) - Geometry.V1PosStart(1,3))*180/pi;
Geometry.DOAV1S=[Geometry.AOAV1Start Geometry.ZOAV1Start]; %DoA = [ AoA ZoA]
```



Defining a Geometry

- Define the Base Station Array with MATLAB built in functions:

```
% Defining a 4x4 antenna array can be done with phased.URA
Geometry.BSarray = phased.URA('Size',[4 4],...
    'ElementSpacing',[Pars.lambda/2 Pars.lambda/2],'ArrayNormal','x');
%To get position of elements of the array simply call
Geometry.BSAntennaPos=getElementPosition(Geometry.BSarray);
```

- In this case we have defined a 4x4 array with spacing $\lambda / 2$, we can get the position of each element using the getElementPosition function.



Simple LOS Channel Model

- Calculate path loss from the distance:

$$\text{FSPL} = \frac{P_t}{P_r} = \left(\frac{4\pi d}{\lambda} \right)^2$$

- Apply it to the waveform (Remember waveforms are voltages, not power!)
- Collect the waveform(s) at the receiving BS array. (in this case we are considering the UPLINK)

```
%Calculation on waveform
```

```
receivedW = collectPlaneWave(Geometry.BSarray, [waveform1 waveform2], ...  
    [Geometry.DOAV1S' Geometry.DOAV2S'], Pars.fc);
```



Simple LOS Channel Model

- Add AWGN:

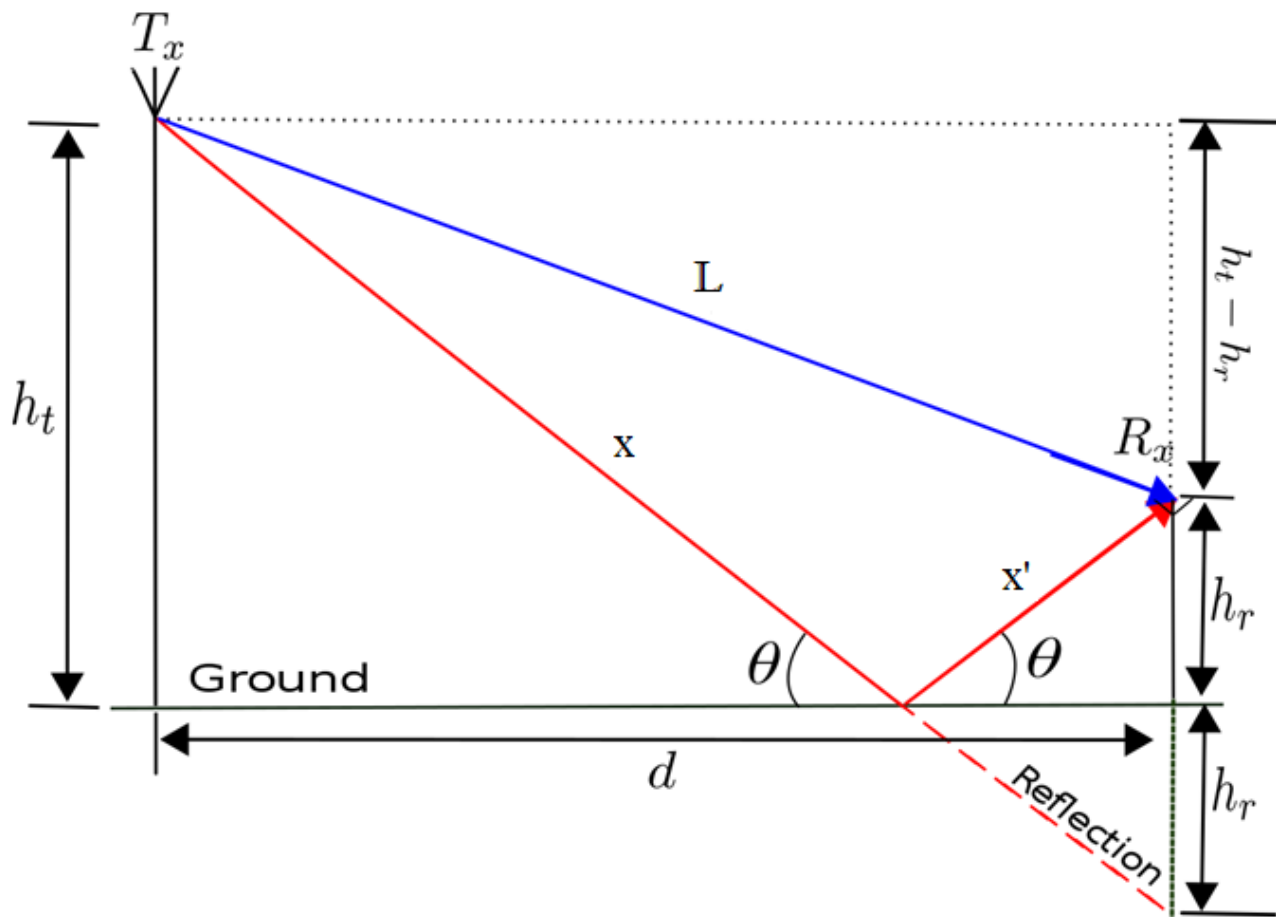
```
Pars.SNR=20; %20db  
%add AWGN  
chOut=awgn(receivedW,Pars.SNR,'measured');
```

- Done!

Two ray reflection

A slightly more complex channel model:

L : LOS path
 $x + x'$: Reflected Path
 θ : Angle of Reflection



Two ray reflection

- At the receiver we will have the sum of the two fields.
How to calculate them? Let's simplify the problem with the following assumptions:

1. Reflection coefficient: $\Gamma(\theta) = -1$

2. Similar Antenna Gain for both paths:

$$G_{LOS} \approx G_{Reflected} = G$$

3. Large d approximation:

$$\Delta d = x + x' - L \approx \frac{2h_t h_r}{d}$$



Two ray reflection

- Given our assumptions we have

$$P_r \approx P_t G \left(\frac{\lambda}{4\pi d} \right)^2 |1 - e^{-j\Delta\phi}|^2$$

where

$$\Delta\phi = \frac{2\pi\Delta d}{\lambda} \approx \frac{4\pi h_t h_r}{\lambda d}$$

Two ray reflection

- Let's try this with a sinusoid

```
Fsin=600;    %Sinusoid Frequency
Ts=1e-5;    %Sampling time for the sinusoid
Fsample=1/Ts; %10khz
TsVect=0:Ts:5/Fsin; %Vector of sampling times for 5 periods
sinusoid_waveform=sin(2*pi*Fsin*TsVect); %generate the waveform
txPos=Geometry.BSPos;
htx=txPos(3);
rxPos=Geometry.V1PosStart;
hrx=rxPos(3);
rxPosR=rxPos;
rxPosR(3)=-rxPosR(3); %invert the Z coordinate
```

- Calculate the distances

```
%LOS path distance
l= sqrt(sum((rxPos-txPos).^2));
%Reflected path distance (just use negative height!)
x=sqrt(sum((rxPosR-txPos).^2));
%2D distance (d)
d=sqrt(sum((rxPos(1:2)-txPos(1:2)).^2));
DeltaD=x-l;
```



Two ray reflection

- Calculate theoretical results with isotropic antennas

```
%Remember power of a sinusoid= a^2/2, a=1
%suppose both isotropic antennas for now (G=1)
PrTheory=0.5 * 1 * (Pars.lambda/(4*pi*d))^2*...
    sqrt(abs(1-exp(1i*(2*pi*DeltaD/(Pars.lambda)))));
PrApprox=0.5 * 1 * (Pars.lambda/(4*pi*d))^2*...
    sqrt(abs(1-exp(1i*(4*pi*htx*hrx/(Pars.lambda*d)))));
```

- Which give... `>> [PrTheory, PrApprox]`

```
ans =
```

```
1.0e-08 *
```

```
0.0044    0.1832
```

...completely
different
numbers!

Two ray reflection

- Let's try with a model available in MATLAB

```
pos1 = txPos'; %Adapting dimensions
pos2 = rxPos';
vel1 = [0;0;0]; % Transmitter and receiver not moving
vel2 = [0;0;0];
swav=sinusoid_waveform';

channel = phased.TwoRayChannel('SampleRate',Fsample,...
    'GroundReflectionCoefficient',-1,'OperatingFrequency',Pars.fc,...
    'CombinedRaysOutput',true);
prop_signal = channel([swav,swav],pos1,pos2,vel1,vel2);
```

- For more info you can look up the help for `phased.TwoRayChannel`



Two ray reflection

- We get an output sinusoid, calculating its power we get:

```
>> RxPow = mean(abs(prop_signal).^2);  
[RxPow PrTheory PrApprox]
```

```
ans =
```

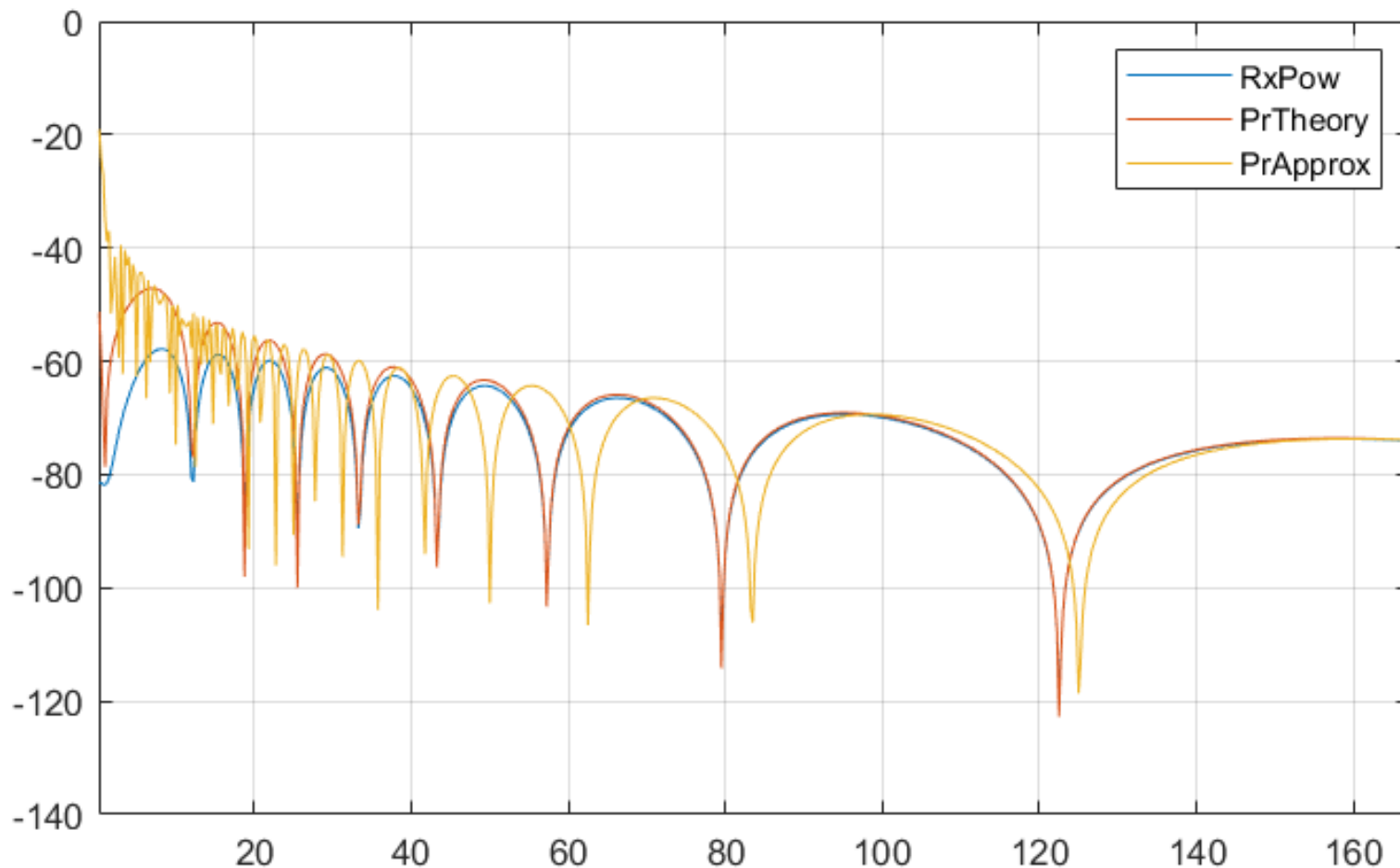
```
1.0e-08 *
```

0.0043	0.0044	0.1832
--------	--------	--------

So we can see there is a problem with the approximation

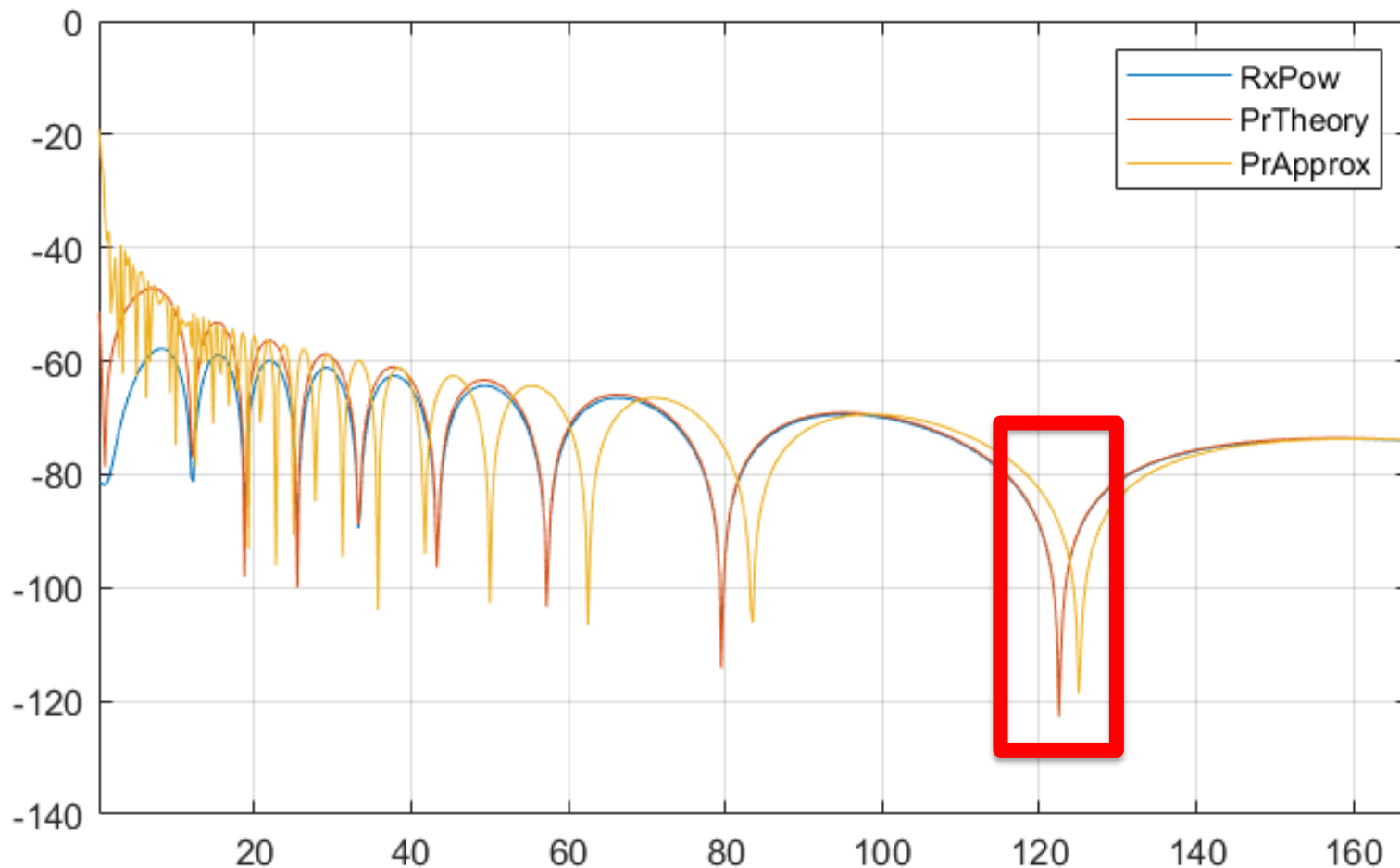
Two ray reflection

- We can put all of that in a for loop and plot some results varying for example $\text{rxPos}=[\text{idx},0,1.5]$;



Two ray reflection

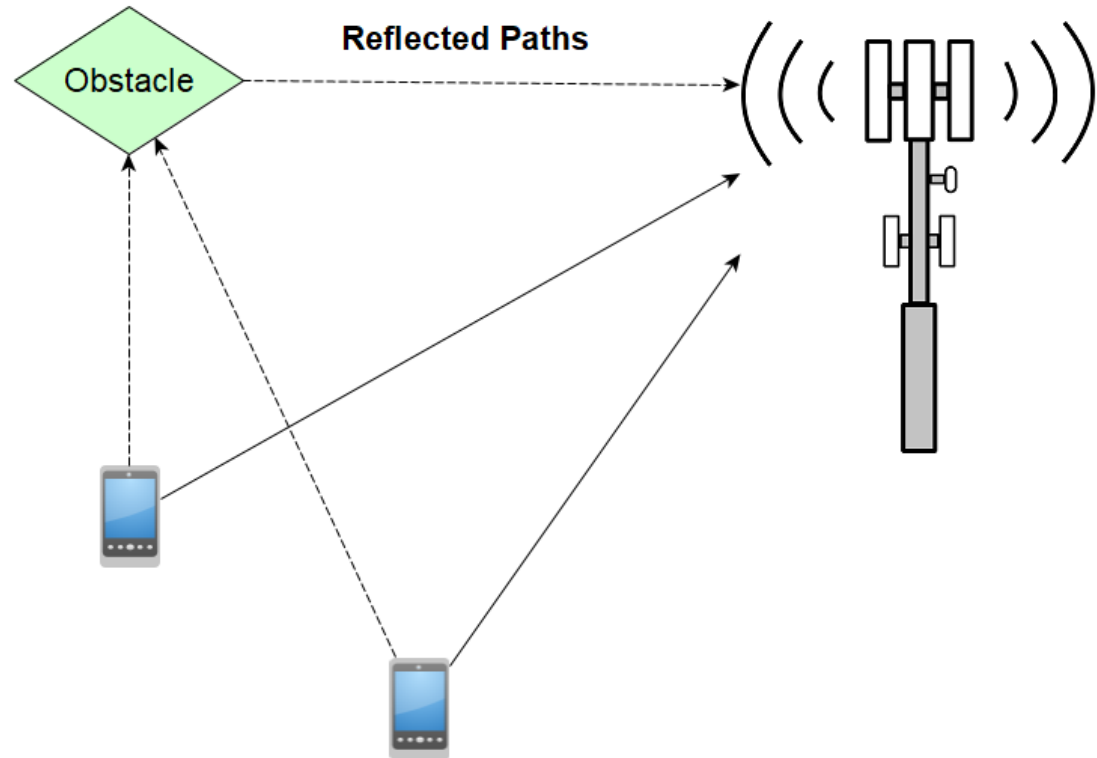
- We can put all of that in a for loop and plot some results varying for example $\text{rxPos}=[\text{idx}, 0, 1.5]$;



Simplified Multipath

How to add a reflection

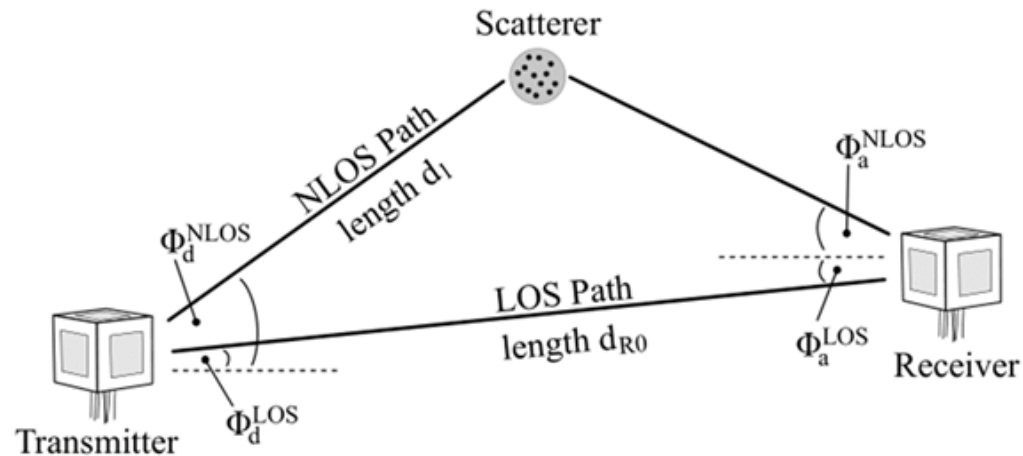
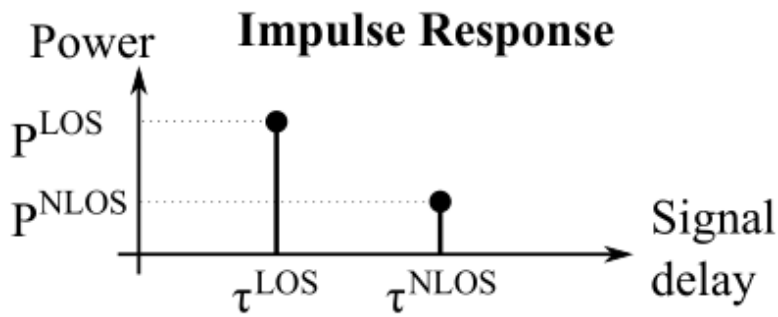
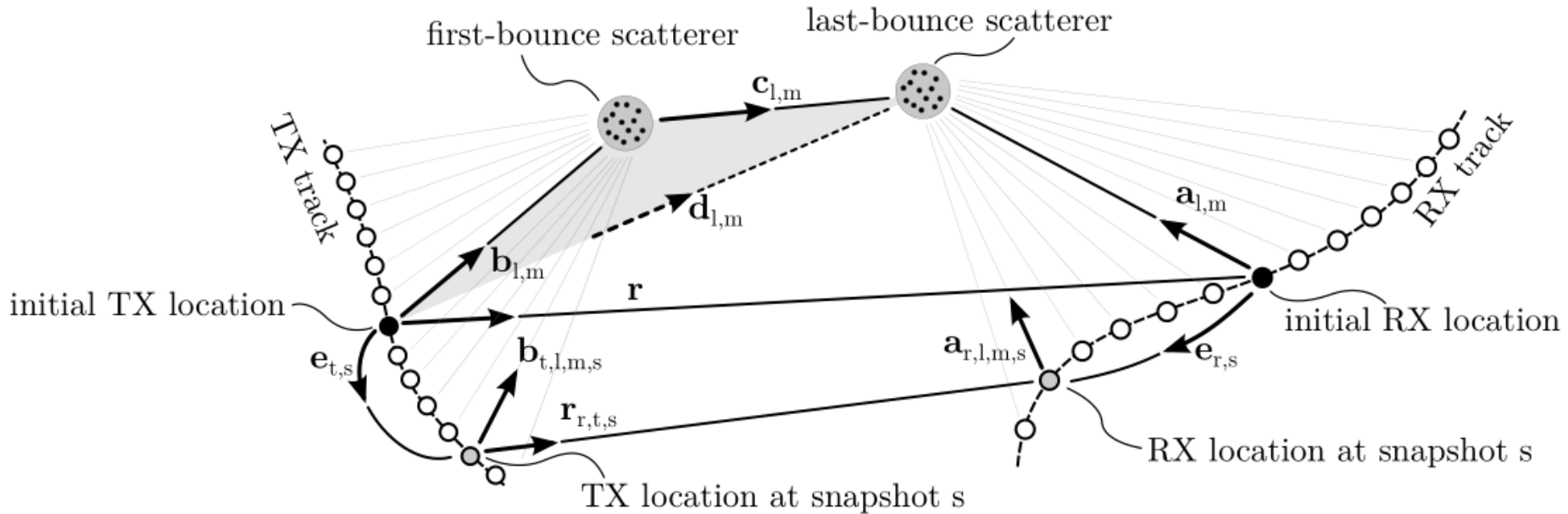
1. Find the signal that arrives at the reflector
2. Retransmit it from the obstacle position with some losses (ex. 10dB)
3. Done!



Be mindful of delays!

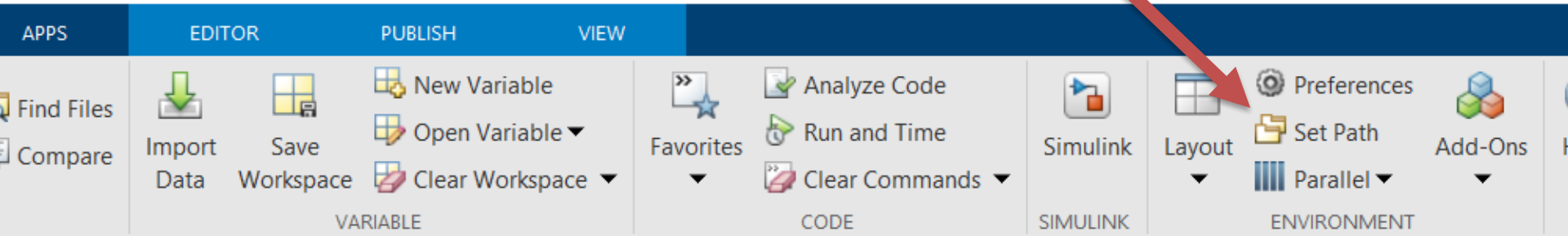
With this channel, **and with every type of channel that introduces phaseshifts**, such as the matlab free space channel, in order to extract a sensible BER a channel equalization must be performed!


Quadriga Channel Model



Quadriga Channel Model

- Downloadable from: <https://quadriga-channel-model.de/#Download>
- Install by adding the source folder to MATLAB path



 Set Path

All changes take effect immediately.

Add Folder...

Add with Subfolders...

MATLAB search path:

- C:\Users\Davide Scazzoli\Documents\MATLAB
- C:\DATA\Ricerca\NATO\MATLAB\QuaDriGa_2019.06.27_v2.2.0\quadriga_src
- C:\DATA\Ricerca\NATO\MATLAB\QuaDriGa_2019.06.27_v2.2.0

Quadriga Channel Model

- In the event you get this error using the newest version of the model:

```
>> qd_builder.supported_scenarios
Index exceeds the number of array elements (0).

Error in qd_builder.supported_scenarios (line 61)
config_folder = fullfile(quadriga_path{1}{1}, 'config');
```

- Open the offending file and replace the following lines of code:

```
59 % Parse files in config-folder
60 - quadriga_path = regexp(path, '?:([^\:]*quadriga_src):', 'tokens');
61 - config_folder = fullfile(quadriga_path{1}{1}, 'config');
62 - files_tmp = dir(fullfile(config_folder, '*.conf'));
```

- With those from the previous version

```
59 % Parse files in config-folder
60 - information = what('quadriga_src');
61 - config_folder = [information.path, filesep, 'config', filesep];
62 - files_tmp = dir([config_folder, '*.conf']);
```



Quadriga Channel Model

- Create a Layout

```
%Create Layout Object  
l = qd_layout;  
% Set scenario parameter:  
l.set_scenario('QuaDRiGa_UD2D_LOS');
```

- Define transmitters and receivers

```
txArr=qd_arrayant('omni');  
rxArr=qd_arrayant('omni');  
rxArr.no_elements=16;  
rxArr.element_position=Geometry.BSAntennaPos;
```



Quadriga Channel Model

- Give the geometry information to the Quadriga Layout:

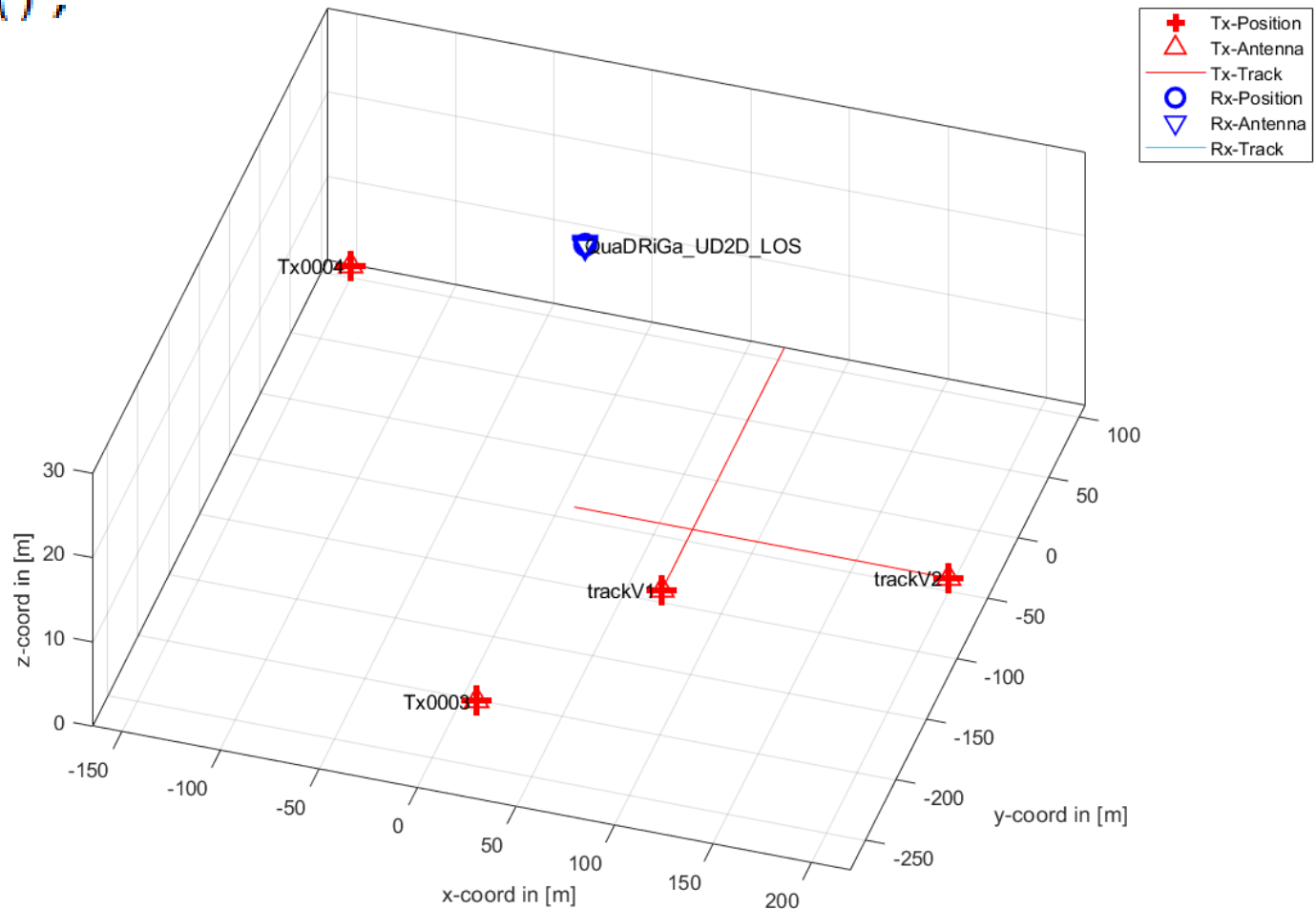
```
l.tx_array = txArr;
l.rx_array = rxArr;
l.no_rx = 1;
l.no_tx = 4;
tx_track1 = qd_track('linear',Geometry.T1,pi/2);
tx_track1.name='trackV1';
tx_track2 = qd_track('linear',Geometry.T2,pi);
tx_track2.name='trackV2';
tx_track1.initial_position=Geometry.V1PosStart';
tx_track2.initial_position=Geometry.V2PosStart';
l.tx_position=[Geometry.V1PosStart',Geometry.V2PosStart',...
    Geometry.I1Pos', Geometry.I2Pos'];
l.tx_track(1,1)=copy(tx_track1);
l.tx_track(1,2)=copy(tx_track2);
l.rx_position=Geometry.BSPos';
```



Quadriga Channel Model

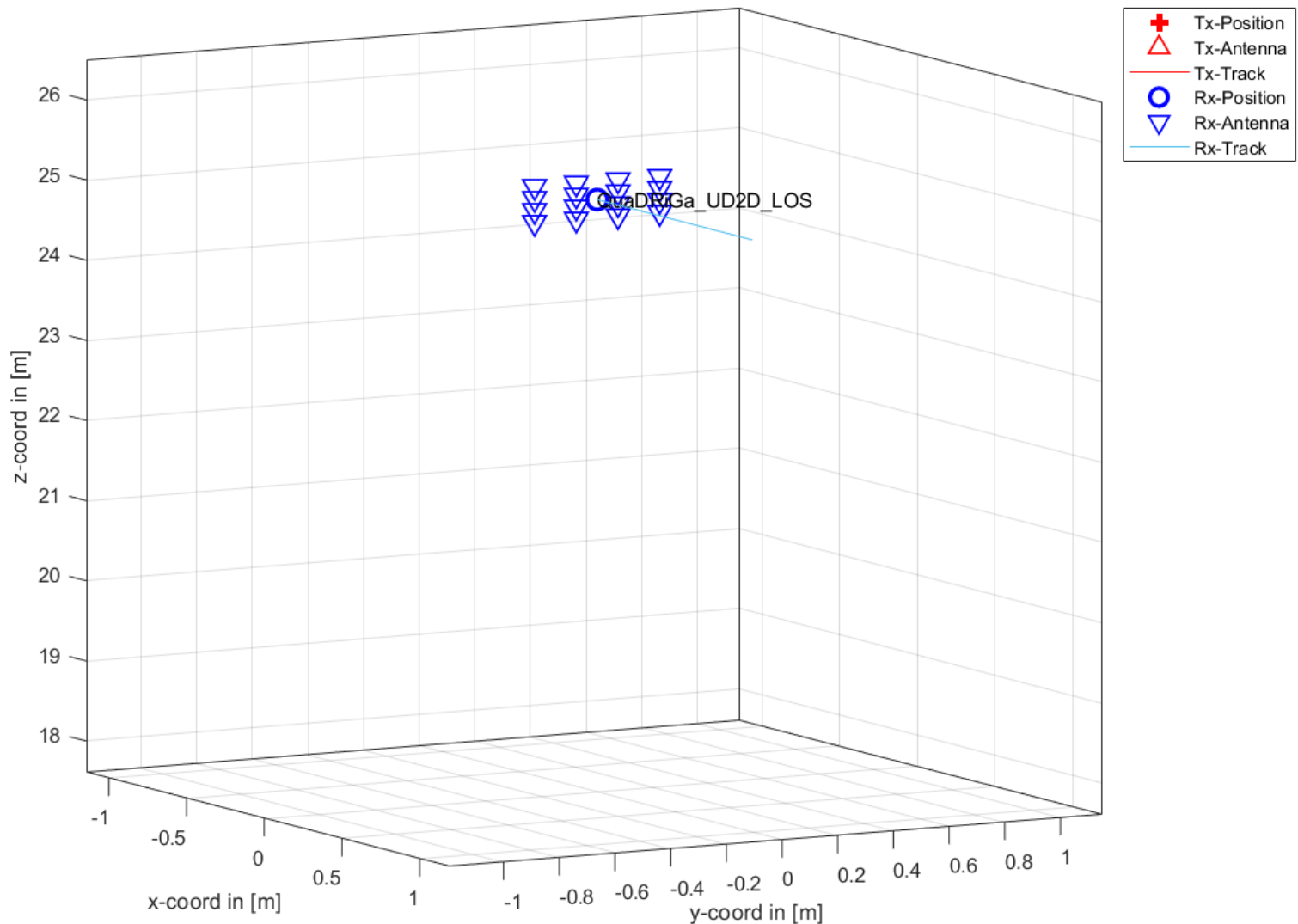
- Visualize the scenario:

```
l.visualize();
```



Quadriga Channel Model

Zooming on
the RX:



Quadriga Channel Model

- To get the channel we must run the model:

```
l.set_pairing;           % Evaluate all links
chan = l.get_channels;    % Generate input for channel_builder
Starting channel generation using QuaDRiGa v2.2.0-0
1 receiver, 4 transmitters, 1 frequency (1.0 GHz)
Warning: Sample density in tracks does not fulfill the sampling
theoreme.
> In gd_layout/get_channels (line 268)
Generating channel builder objects
- 1 builder, 4 channel segments
Initializing random generators
Generating parameters
SSF Corr.      [oooooooooooooooooooooooooooooooooooooooooooooooooooo]
Channels       [oooooooooooooooooooooooooooooooooooooooooooooooooooooooo]
Merging        [oooooooooooooooooooooooooooooooooooooooooooooooooooooooo]
Formatting output channels - 4 channel objects
Total runtime: 1 seconds
```



Quadriga Channel Model

- We get the following struct as an output:

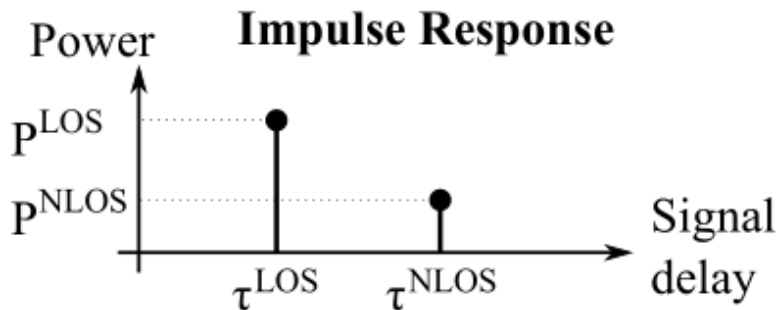
`chan =`

1×4 [`gd_channel`](#) array with properties:

Channel
Coefficients

Channel Delays

name
version
center_frequency
coeff
delay
par
initial_position
tx_position
rx_position
no_rxant
no_txant
no_path
no_snap
individual_delays



Quadriga Channel Model

- Let's look at the details:

```
>> chan(1).name
```

```
ans =
```

```
'Tx0003_Rx0001'
```

- Chan(1) is the channel between Tx 3 and Rx 1, it starts from Tx 3 because Tx 1 & 2 are defined as tracks and are added later. Double check which channel is which!



Quadriga Channel Model

- Let's look at the details:

```
>> coefficients = chan(1).coeff;  
>> whos coefficients
```

Name	Size	Bytes	Class	Attributes
coefficients	16x1x34x2	17408	double	complex

Antenna

Paths (first one is usually LOS path)

- The other two dimensions are the track positions. In order to get the channel for each point in the track you must run the interpolation command. See Example 4.9 in the Quadriga documentation.

Quadrige Channel Model

- To get the output we must perform the convolution:

```
chTaps=size(chan(1).delay);
TS=Ts; %Sampling time, could be different for different waveforms!
WFlength=size(sinusoid_waveform);
chOut=zeros(chTaps(1),WFlength(2));
TsVect=0:TS:TS*(WFlength(2)-1);
for antenna=1:1:chTaps(1)
    for path=1:1:chTaps(3)
        %Get the x for our interpolation
        %we put minus in the delay because we want to do the convolution!
        inX=TsVect-CHAN(1).delay(antenna,1,path,1);
        %Interpolate the waveform
        inY = interp1(TsVect,sinusoid_waveform,inX,'pship');
        %Get the output by multiplying by the coefficient and adding all
        %contributions up!
        chOut(antenna,:)=inY*chan(1).coeff(antenna,1,path,1) ...
            +chOut(antenna,:);
    end
end
```



Quadriga Channel Model

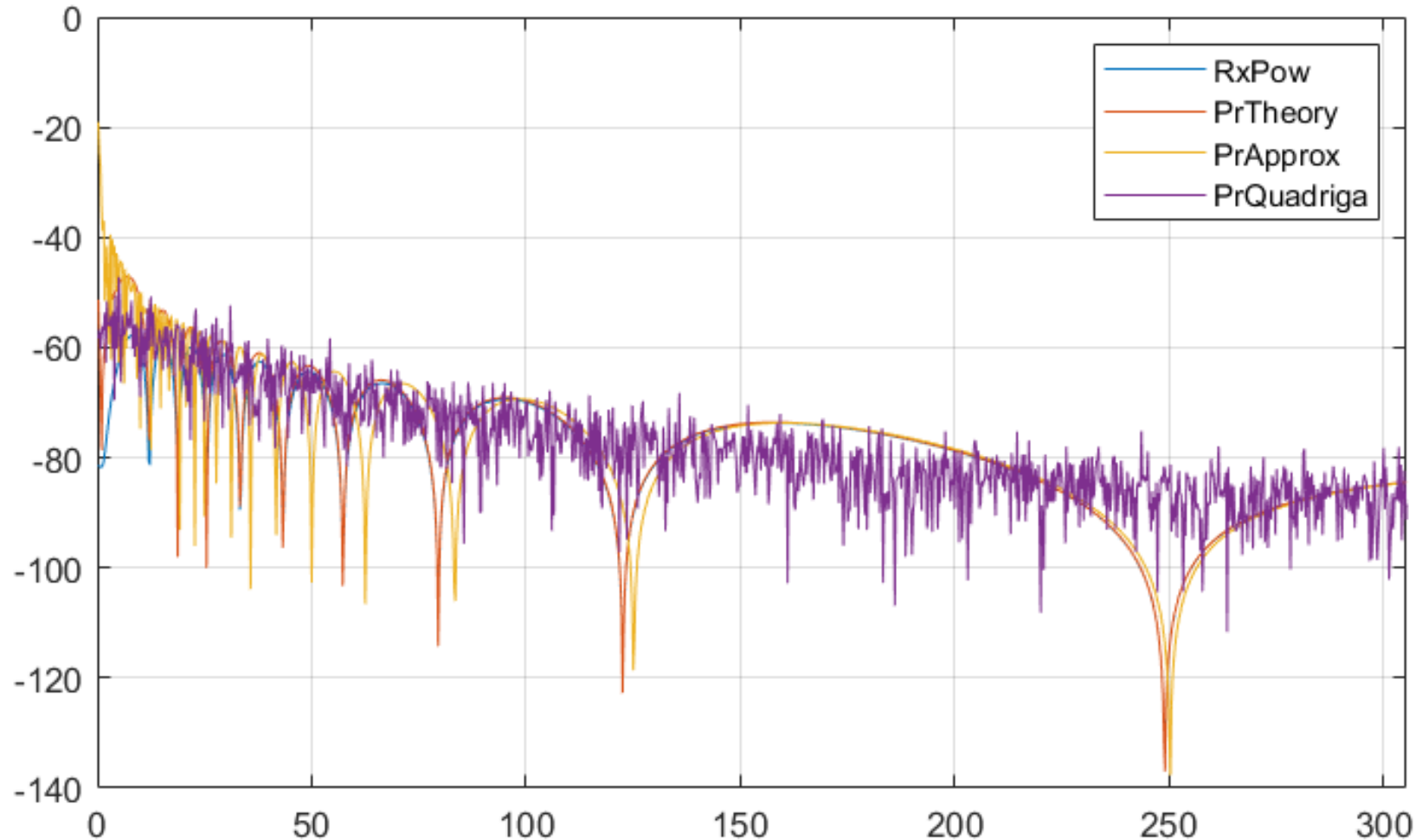
- To get the output we must perform the convolution:

```
chTaps=size(chan(1).delay);
TS=Ts; %Sampling time, could be different for different waveforms!
WFlength=size(sinusoid_waveform);
chOut=zeros(chTaps(1),WFlength(2));
TsVect=0:TS:TS*(WFlength(2)-1);
for antenna=1:1:chTaps(1)
    for path=1:1:chTaps(3)
        %Get the x for our interpolation
        %we put minus in the delay because we want to do the convolution!
        inX=TsVect-CHAN(1).delay(antenna,1,path,1);
        %Interpolate the waveform
        inY = interp1(TsVect,sinusoid_waveform,inX,'pship');
        %Get the output by multiplying by the coefficient and adding all
        %contributions up!
        chOut(antenna,:)=inY*chan(1).coeff(antenna,1,path,1) ...
            +chOut(antenna,:);
    end
end
```

**For simplicity
let's consider
just one position**

Quadriga Channel Model

- We can change the parameters to match those used in the two-ray channel and compare received power:



Quadriga Channel Model

- After all of that we still need to do one more thing...



Quadriga Channel Model

- After all of that we still need to do one more thing...

- Add AWGN:

```
%add AWGN
```

```
chOut=awgn(chOut, Pars.SNR, 'measured') ;
```

- Done!



Questions?



F.A.Q

- Can I use _____ ?

Yes! Provided that:

1. **You know what it is doing.**

2. For Beamforming, you should also have the calculation of the weights using theoretical formula.

You can double check your results with built in formulas such as: `phased.PhaseShiftBeamformer`

- How do I make two separate beams?
 1. Calculate two set of weights
 2. Multiply the [N_{antennas} X W_{waveform}] channel output matrix by the two set of weights [$1 \times N_{\text{antennas}}$] to get the two different outputs. You will get two outputs of size [$1 \times W_{\text{waveform}}$].

Note Generally $A \cdot B \neq B \cdot A$, however the following still applies:

$$(A \cdot B)^t = B^t \cdot A^t$$

Beamforming and Waveforms



Outline

- Generating OFDM signals
- Estimating DOA with MUSIC
- Simple Beamforming Example



Generating OFDM Signals

- Open Waveform Generator Tool



Generating OFDM Signals

The screenshot displays a software interface for generating OFDM signals. The top bar has three tabs: GENERATOR, TRANSMITTER (selected), and a third unlabeled tab. Below the tabs are three main sections: FILE, WAVEFORM TYPE, and GENERATION. The FILE section contains buttons for New Session, Open Session, and Save Session. The WAVEFORM TYPE section contains buttons for OFDM, QAM, and PSK, with a red arrow pointing to the OFDM button. The GENERATION section contains buttons for Impairments, Visualize, Default Layout, Generate, and Export. Below the tabs, there are two main panels: Waveform and Spectrum Analyzer. The Waveform panel shows the OFDM Waveform Configuration with the following parameters: FFT length: 12, Guard band subcarriers: [1;1], Insert DC null (unchecked), Cyclic prefix lengths: [0], Windowing (unchecked), OFDM symbols: 140, Transmit antennas: 1, Subcarrier spacing (Hz): 15000, Pilot input (checked), Pilot indices: [6], and Size of pilot input: [1 140 1]. The Spectrum Analyzer panel shows a plot of dBm vs frequency, with the signal stopped.

TRANSMITTER

FILE

New Session Open Session Save Session

WAVEFORM TYPE

OFDM QAM PSK

GENERATION

Impairments Visualize Default Layout Generate Export

Waveform

▼ OFDM Waveform Configuration

FFT length: 12

Guard band subcarriers: [1;1]

☐ Insert DC null

Cyclic prefix lengths: [0]

☐ Windowing

OFDM symbols: 140

Transmit antennas: 1

Subcarrier spacing (Hz): 15000

☒ Pilot input

Pilot indices: [6]

Size of pilot input: [1 140 1]

Spectrum Analyzer

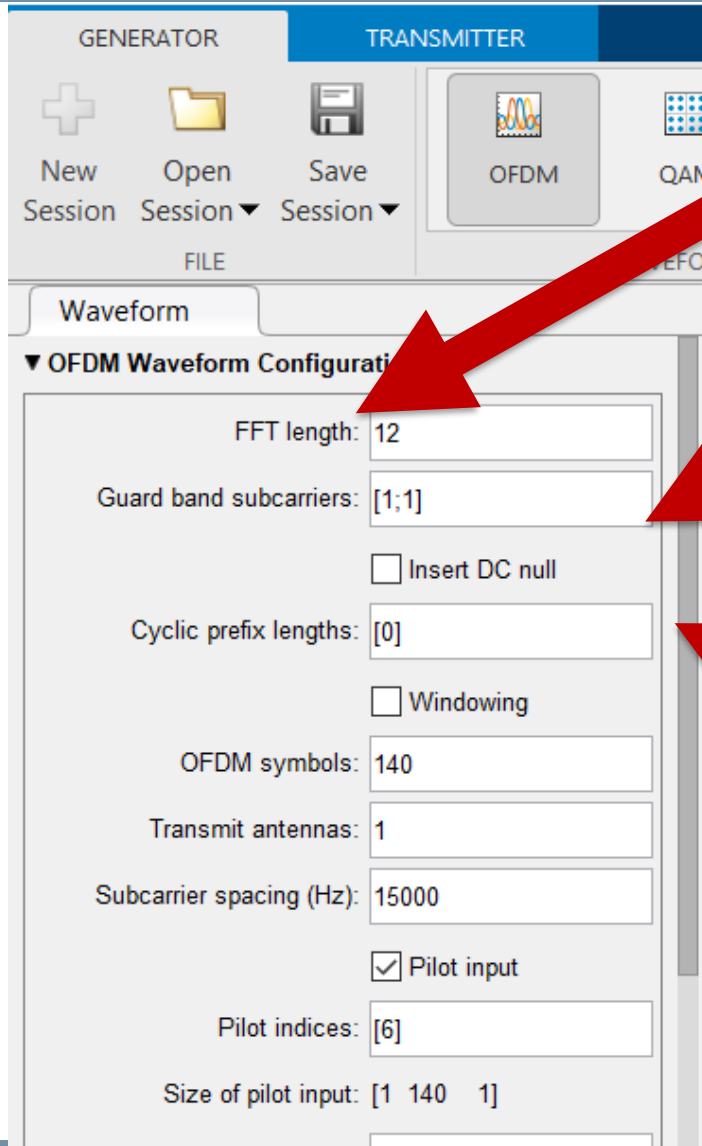
dBm

10 0 -10 -20 -30 -40 -50 -60

-6 -4 -2

Stopped

Generating OFDM Signals



The screenshot shows the 'TRANSMITTER' tab of a software interface. The 'Waveform' section is expanded, showing 'OFDM Waveform Configuration'. The configuration parameters are as follows:

Parameter	Value
FFT length:	12
Guard band subcarriers:	[1;1]
Insert DC null	<input type="checkbox"/>
Cyclic prefix lengths:	[0]
Windowing	<input type="checkbox"/>
OFDM symbols:	140
Transmit antennas:	1
Subcarrier spacing (Hz):	15000
Pilot input	<input checked="" type="checkbox"/>
Pilot indices:	[6]
Size of pilot input:	[1 140 1]

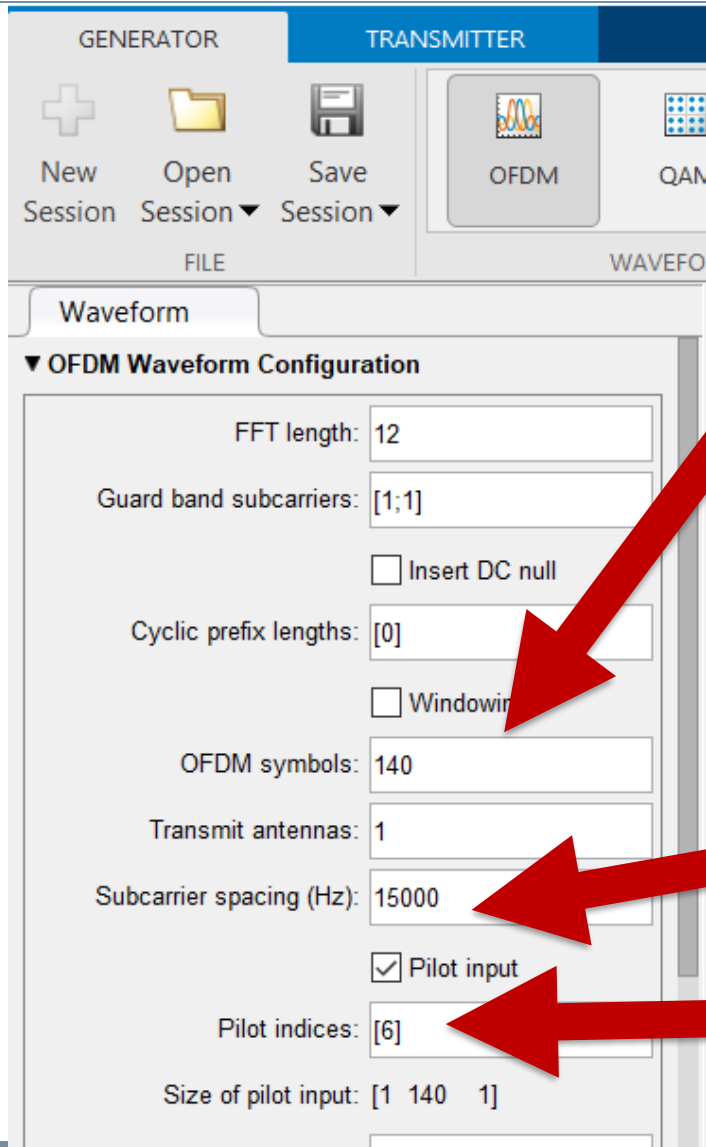
Red arrows point from the explanatory text on the right to the 'FFT length', 'Guard band subcarriers', and 'Cyclic prefix lengths' fields.

FFT length indicates the number of subcarriers

Here you select how many guard band subcarriers there are on each side

Here you choose cyclic prefix length, cannot be greater than OFDM symbols

Generating OFDM Signals



The screenshot shows the 'TRANSMITTER' tab of a software interface. Under the 'Waveform' section, the 'OFDM Waveform Configuration' is expanded. The configuration fields are as follows:

- FFT length: 12
- Guard band subcarriers: [1;1]
- ☐ Insert DC null
- Cyclic prefix lengths: [0]
- ☐ Windowing
- OFDM symbols: 140
- Transmit antennas: 1
- Subcarrier spacing (Hz): 15000
- ☒ Pilot input
- Pilot indices: [6]
- Size of pilot input: [1 140 1]

Three red arrows point from text annotations to the 'Guard band subcarriers' field, the 'Subcarrier spacing (Hz)' field, and the 'Pilot indices' field.

Here you choose the number of symbols to transmit

Subcarrier spacing, for LTE it is 15 khz

Index of pilot inputs if present

Generating OFDM Signals

The screenshot displays a software interface for generating OFDM signals, divided into two main sections: **GENERATOR** and **TRANSMITTER**.

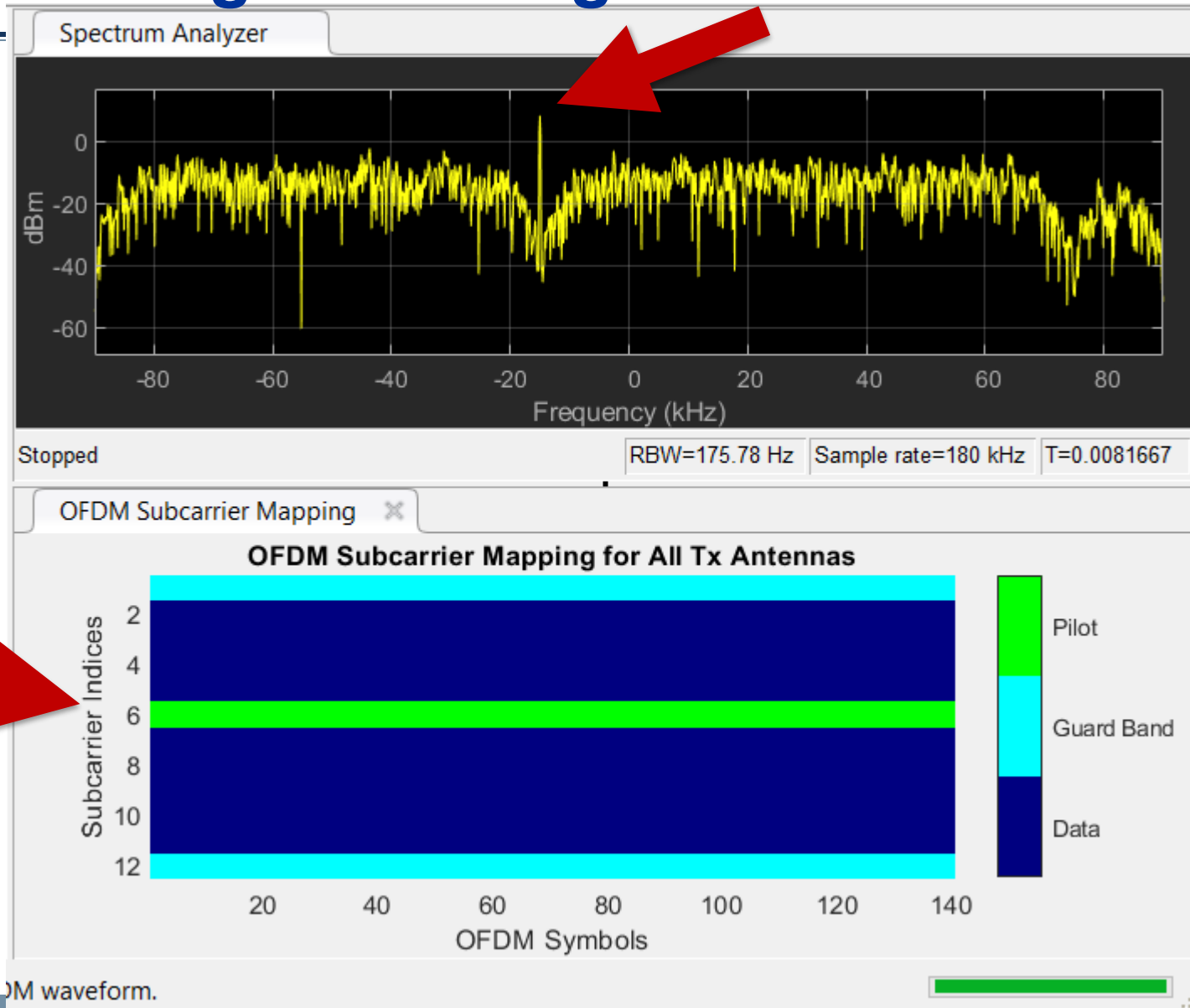
GENERATOR Section:

- FILE:** New Session, Open Session, Save Session.
- WAVEFORM TYPE:** OFDM, QAM, PSK.
- GENERATE:** Impairments, Visualize, Default Layout, **Generate** (highlighted with a red arrow), Export.

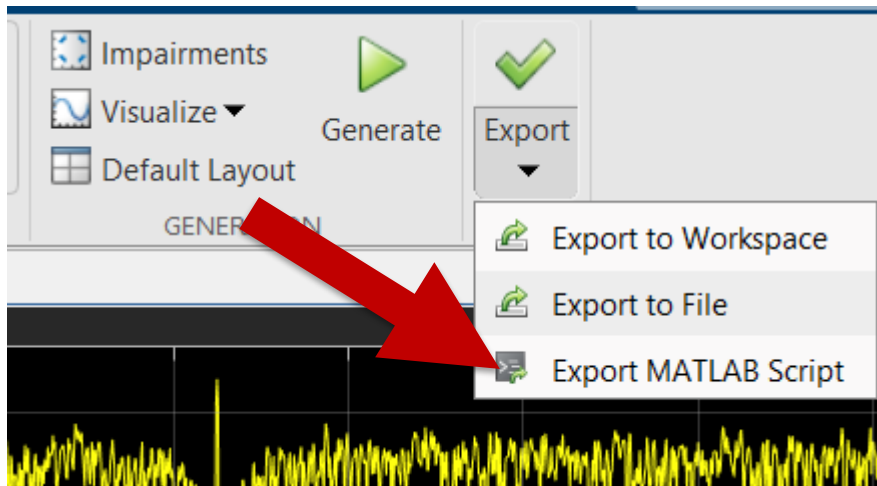
TRANSMITTER Section:

- Waveform Configuration:**
 - FFT length: 12
 - Guard band subcarriers: [1;1]
 - ☐ Insert DC null
 - Cyclic prefix lengths: [0]
 - ☐ Windowing
 - OFDM symbols: 140
 - Transmit antennas: 1
 - Subcarrier spacing (Hz): 15000
 - ☒ Pilot input
 - Pilot indices: [6]
 - Size of pilot input: [1 140 1]
- Spectrum Analyzer:** A plot showing dBm vs. frequency (Hz) with a grid. The plot is currently **Stopped**.

Generating OFDM Signals



Generating OFDM Signals



```
% input bit source:  
in = randi([0 1], 2800, 1);
```

**You will need these bits
to calculate the BER**

```
%% Generate OFDM Waveform  
% OFDM configuration:  
ofdmMod = comm.OFDMModulator('FFTLength', 12, ...  
    'NumGuardBandCarriers', [1;1], ...  
    'InsertDCNull', false, ...  
    'CyclicPrefixLength', [0], ...  
    'Windowing', false, ...  
    'NumSymbols', 140, ...  
    'NumTransmitAntennas', 1, ...  
    'PilotInputPort', true, ...  
    'PilotCarrierIndices', [6]);
```

```
M = 4; % Modulation order  
% input bit source:  
in = randi([0 1], 2800, 1);
```

```
dataInput = qammod(in, M, 'gray', 'InputType', 'bi  
ofdmInfo = info(ofdmMod);  
ofdmSize = ofdmInfo.DataInputSize;  
dataInput = reshape(dataInput, ofdmSize);
```

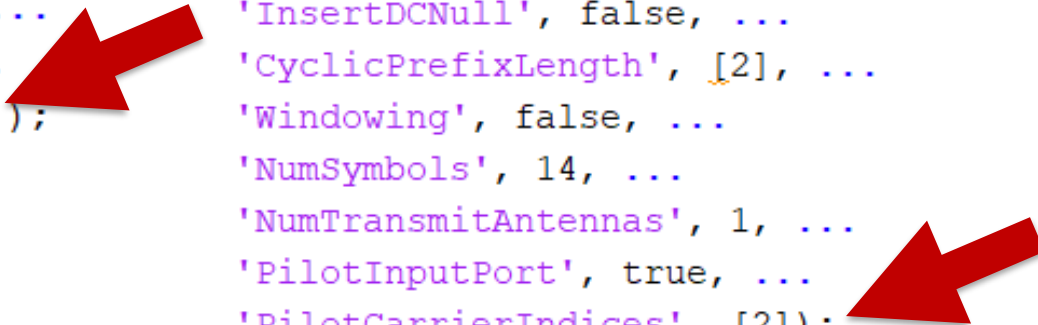
```
% waveform generation:  
pilotInput = ones(1, 140, 1);  
waveform = ofdmMod(dataInput, pilotInput);
```

```
Fs = 180000; % sample rate
```

```
%% Visualize OFDM Waveform
```

Generating OFDM Signals

```
ofdmMod1 = comm.OFDMModulator('FFTLength', 12, ...  
    'NumGuardBandCarriers', [0;0], ...  
    'InsertDCNull', false, ...  
    'CyclicPrefixLength', [2], ...  
    'Windowing', false, ...  
    'NumSymbols', 14, ...  
    'NumTransmitAntennas', 1, ...  
    'PilotInputPort', true, ...  
    'PilotCarrierIndices', [11]);  
  
ofdmMod2 = comm.OFDMModulator('FFTLength', 12, ..  
    'NumGuardBandCarriers', [0;0], ...  
    'InsertDCNull', false, ...  
    'CyclicPrefixLength', [2], ...  
    'Windowing', false, ...  
    'NumSymbols', 14, ...  
    'NumTransmitAntennas', 1, ...  
    'PilotInputPort', true, ...  
    'PilotCarrierIndices', [2]);
```



You can generate the demodulator from the modulator definition very quickly:


```
% Build the companion demod for later  
ofdmDemod1=comm.OFDMDemodulator(ofdmMod1);  
ofdmDemod2=comm.OFDMDemodulator(ofdmMod2);
```

DOA Estimation with MUSIC

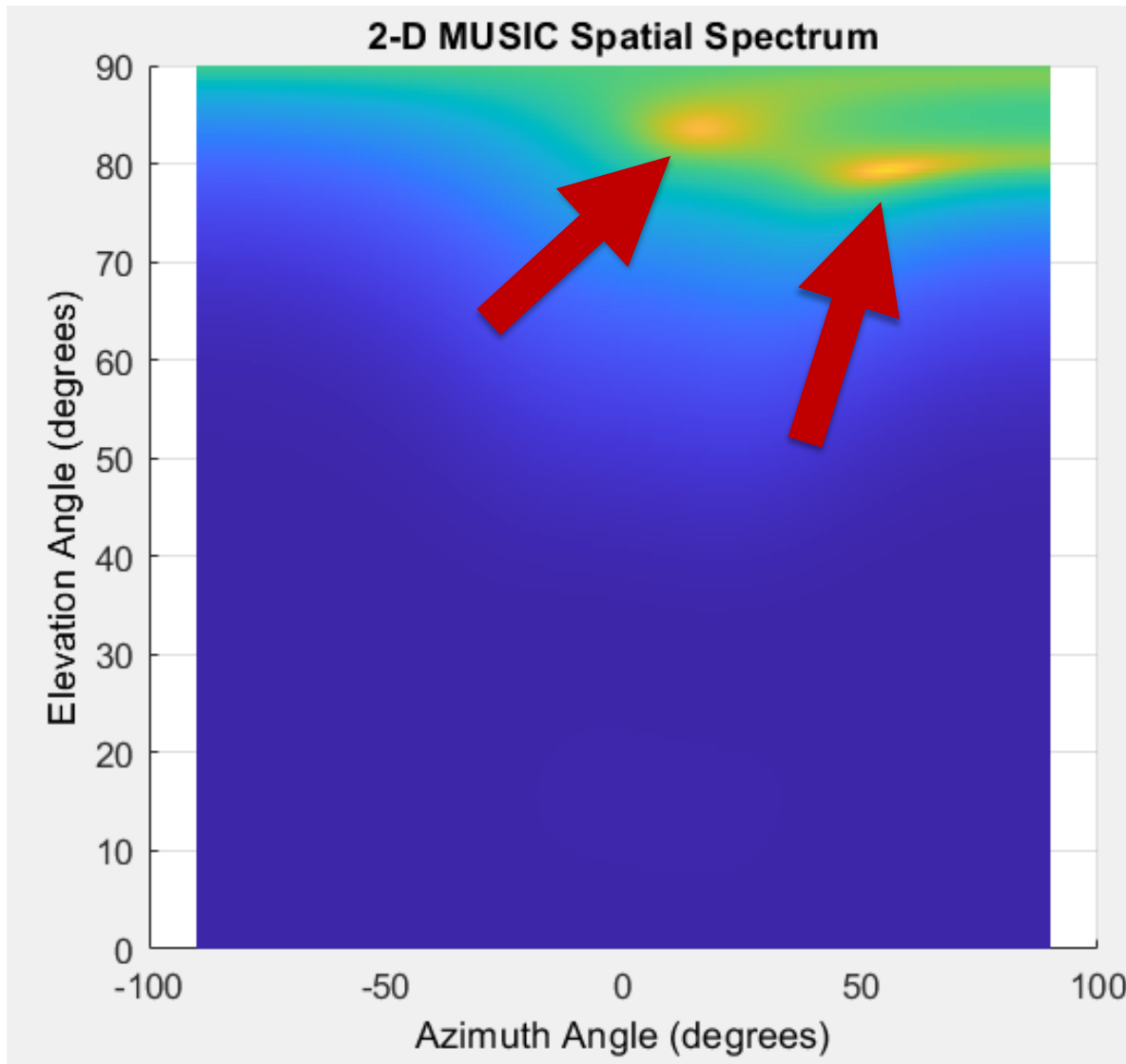
```
% One way to estimate DoA is using
% the MULTiple Signal Classification algorithm:
estimator = phased.MUSICEstimator2D(...
    'SensorArray',Geometry.BSarray,...
    'OperatingFrequency',Pars.fc,...
    'ForwardBackwardAveraging',true,...
    'NumSignalsSource','Property',...
    'DOAOutputPort',true,'NumSignals',2,...
    'AzimuthScanAngles',-90:0.5:90,...
    'ElevationScanAngles',0:0.5:90);
[~,doas] = estimator(chOut);

%Plot the 2-D spatial spectrum
figure
plotSpectrum(estimator);
% Be careful about changing frame of reference!
```

Depending on array geometry you might get mirror images so limiting AOA to 180° can help avoiding that.



DOA Estimation with MUSIC



doas =

55.0000	16.0000
79.0000	83.5000

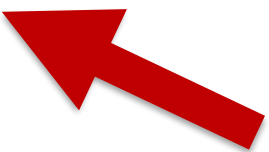
We had to put two different pilot carriers as otherwise we would get an average of the two DOAs.

Simple Beamforming Example

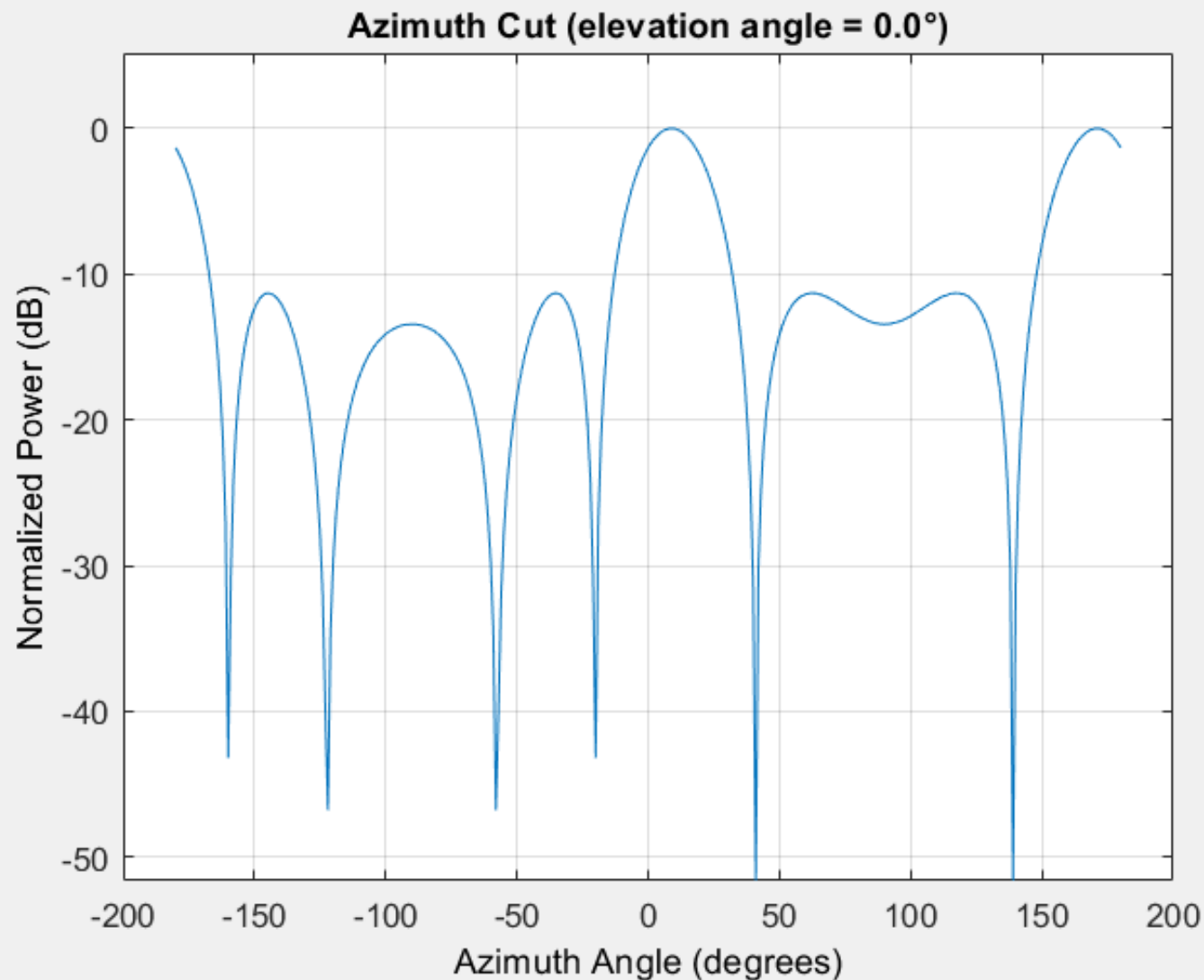
```
% Build a simple beamformer
beamformer = phased.PhaseShiftBeamformer(...
    'SensorArray', Geometry.BSarray, ...
    'OperatingFrequency', Pars.fc, 'Direction', Geometry.DOAV1S', ...
    'WeightsOutputPort', true);

[arrOut, w] = beamformer(chOut);

%plot of the array pattern at azimuth = 0°
figure,
pattern(Geometry.BSarray, Pars.fc, [-180:180], 0, ...
    'CoordinateSystem', 'rectangular', ...
    'Type', 'powerdb', 'PropagationSpeed', physconst('LightSpeed'), ...
    'Weights', w)
```



Simple Beamforming Example



Simple Beamforming Example

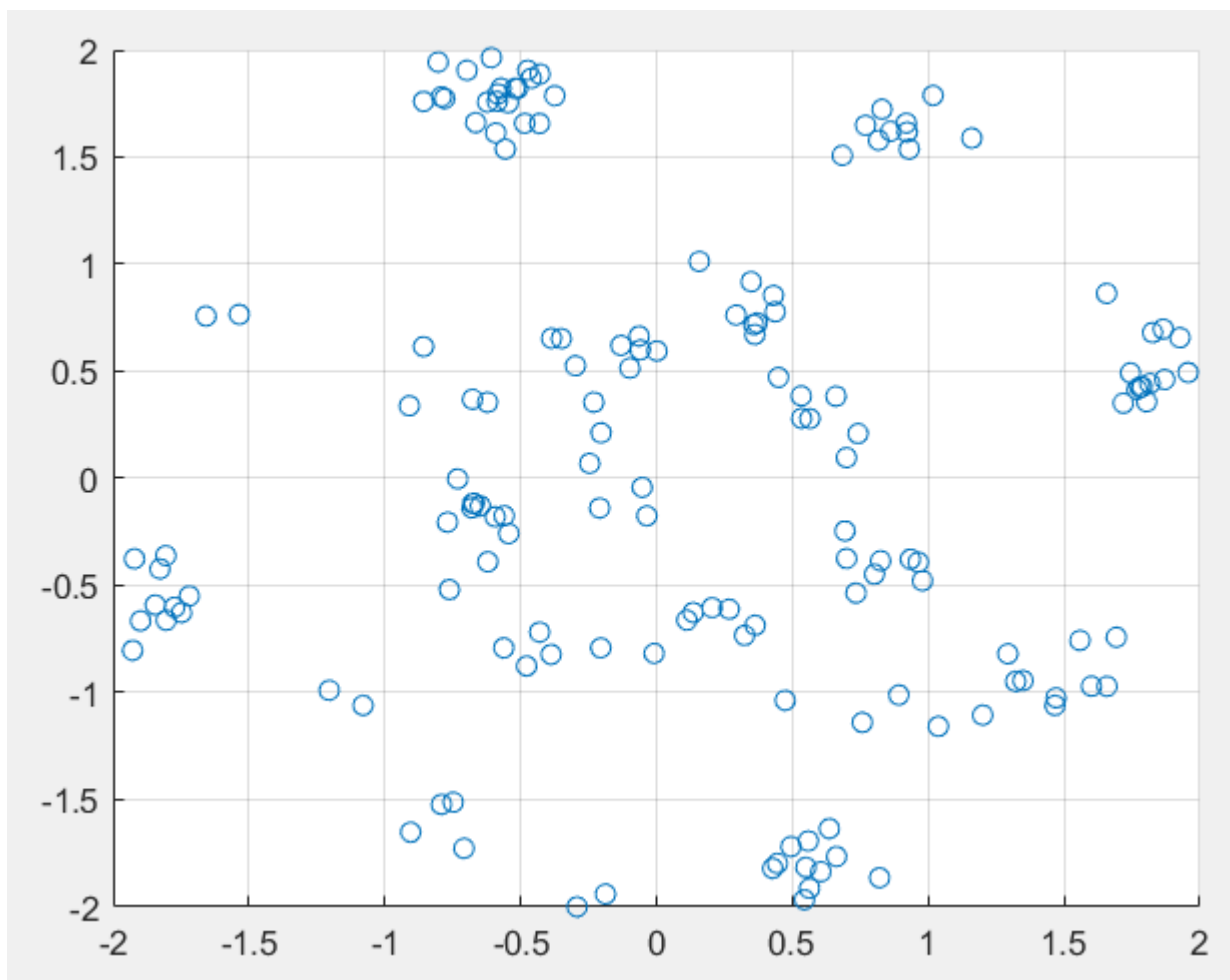
- Plotting a scattergraph to see the impact of BF

```
% Without beamformer
out = ofdmDemod1 (chOut(:,1));
figure,
%Plotting a scattergraph
x=real(out);
x=reshape(x, [11*14,1]);
y=imag(out);
y=reshape(y, [11*14,1]);
scatter(x,y)
```

```
% With beamformer
out = ofdmDemod1 (arrOut);
figure,
%Plotting a scattergraph
x=real(out);
x=reshape(x, [11*14,1]);
y=imag(out);
y=reshape(y, [11*14,1]);
scatter(x,y)
```

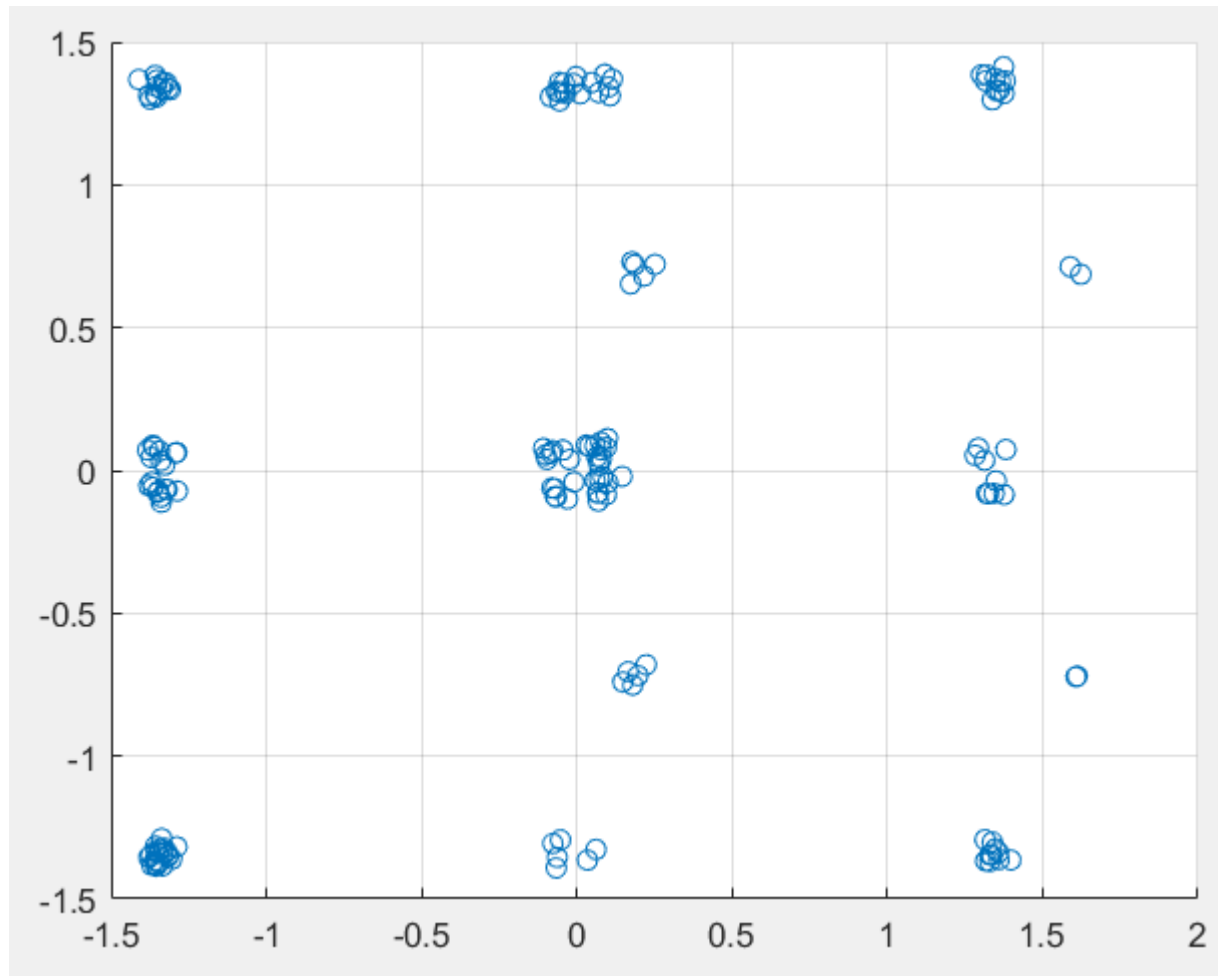
Simple Beamforming Example

- Without Beamforming



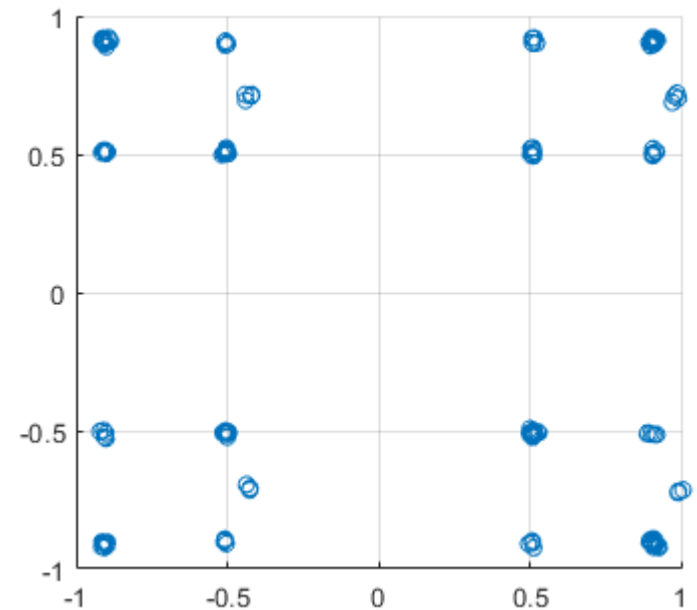
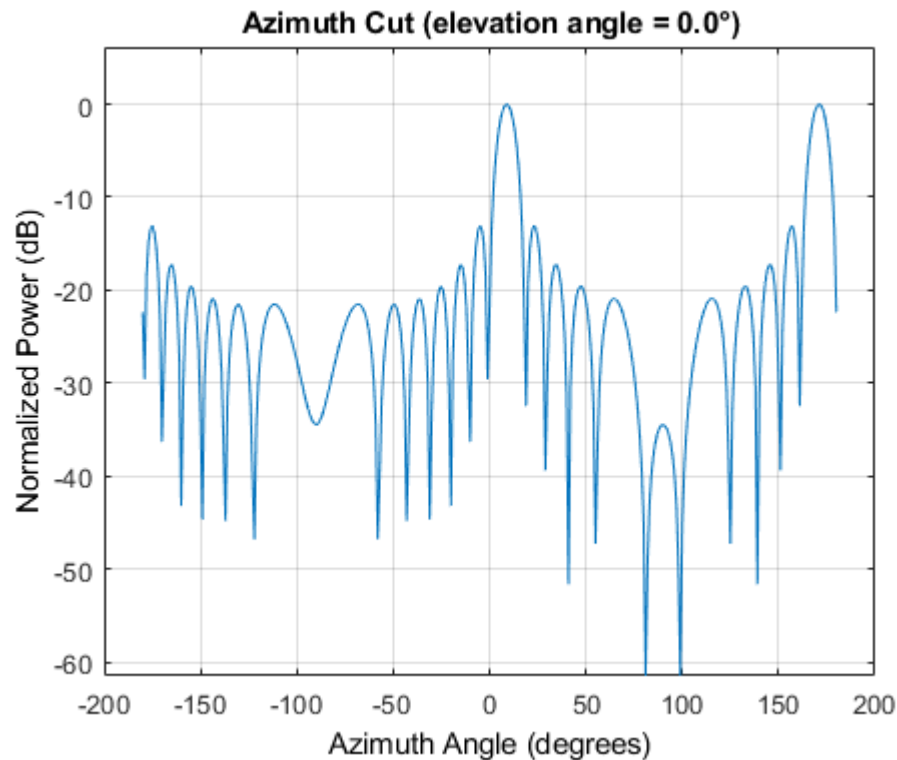
Simple Beamforming Example

- With Beamforming



Simple Beamforming Example

- With more antennas



Thank you!
Questions?

