Wireless Communications

Channel models for the project

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

POLITECNICO
MILANO 1863

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 cal
Geometry.BSAntennaPos=getElementPosition(Geometry.BSarray);
```

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

Simple LOS Channel Model

- Calculate path loss from the distance:

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

- 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!

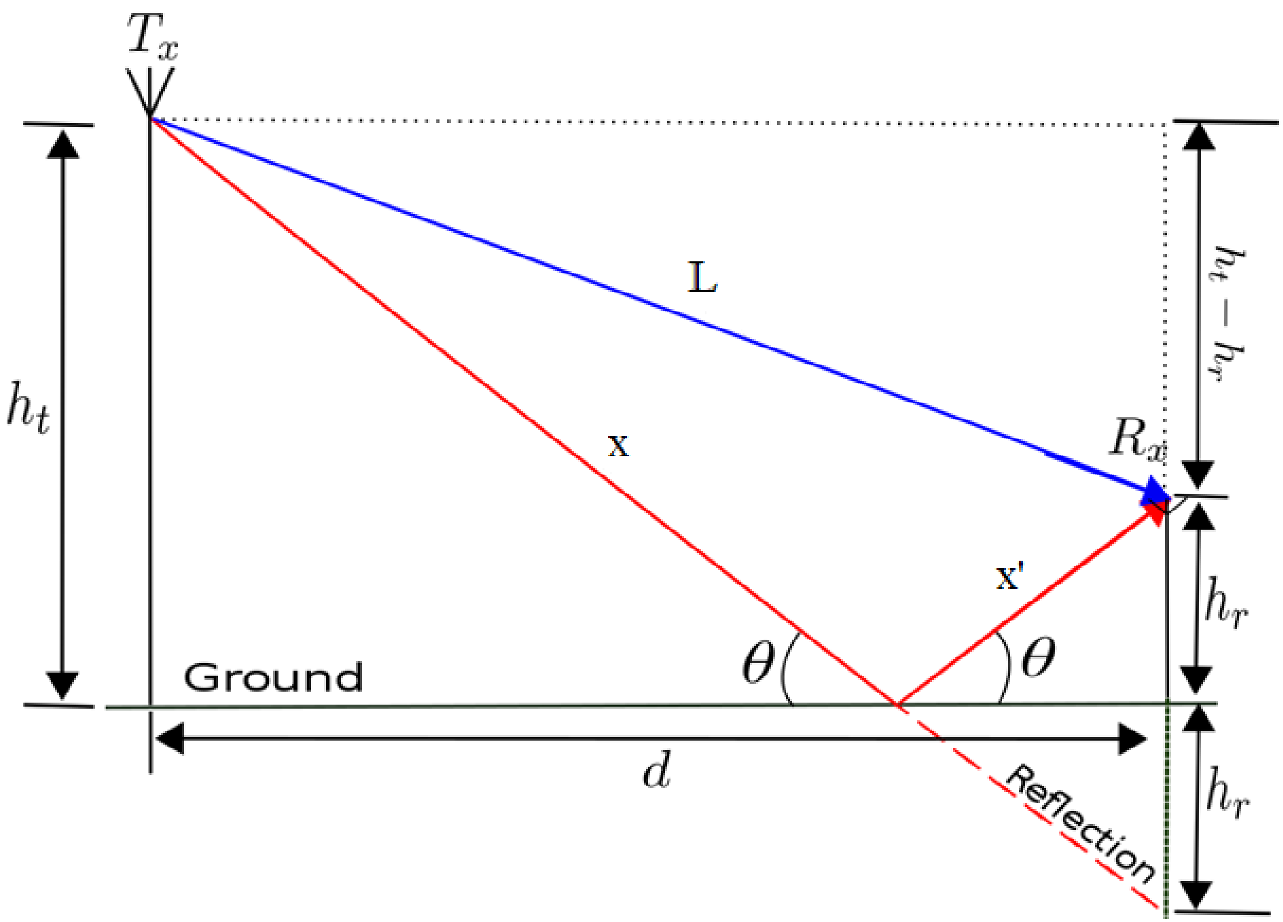
Two ray reflection

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

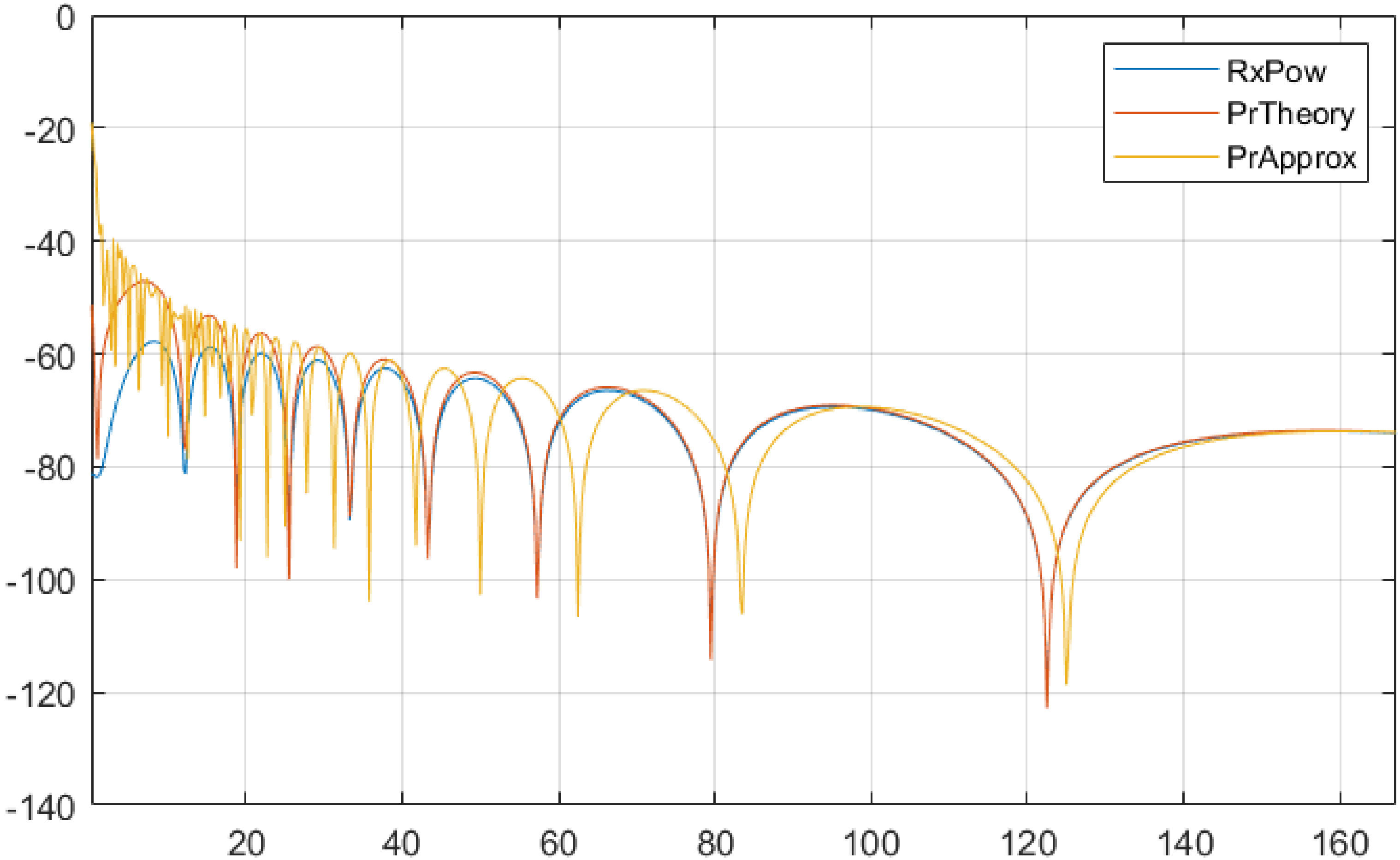
- We get an output sinusoid, calculating its power we get:

```
>> 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

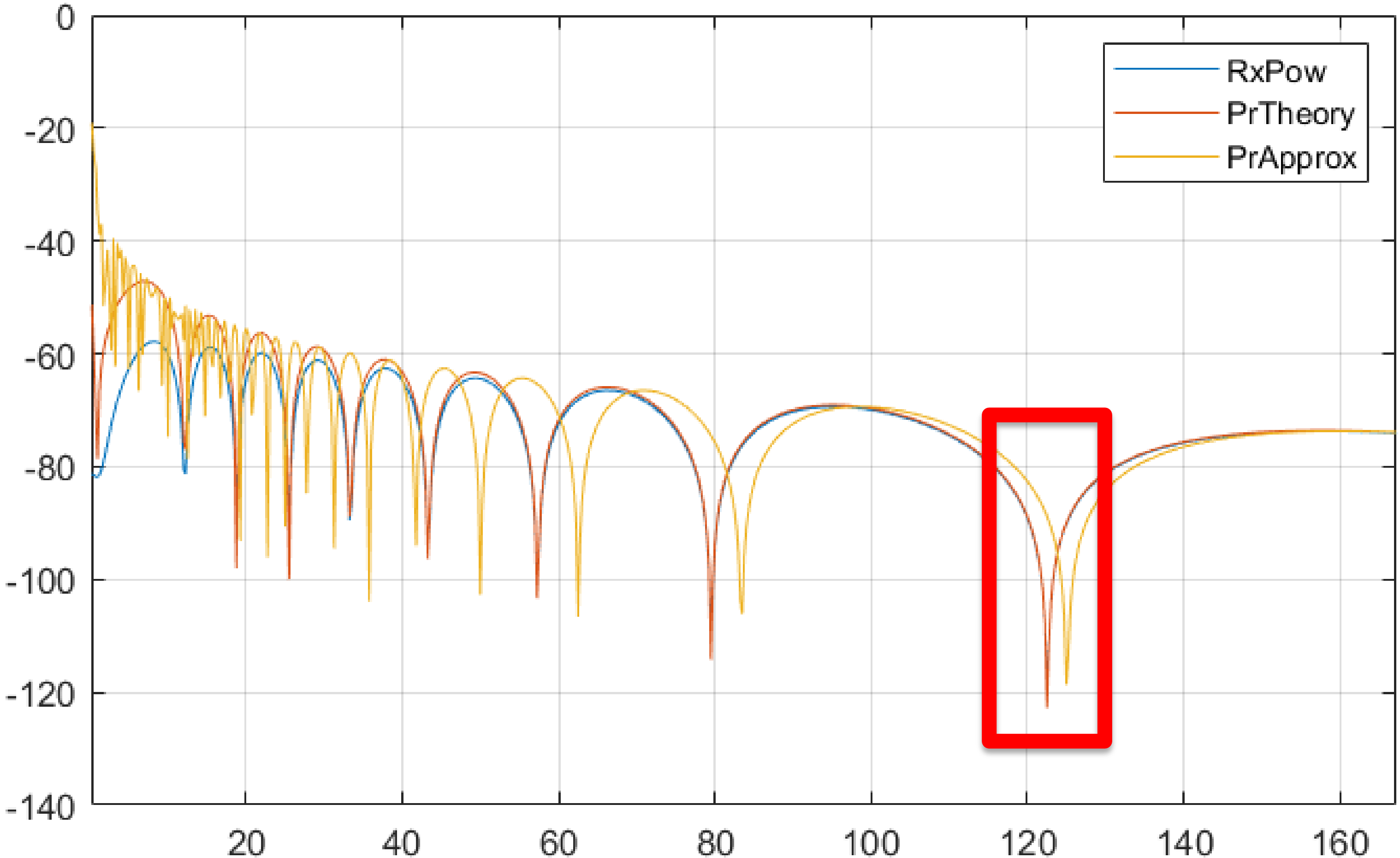
Two ray reflection

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

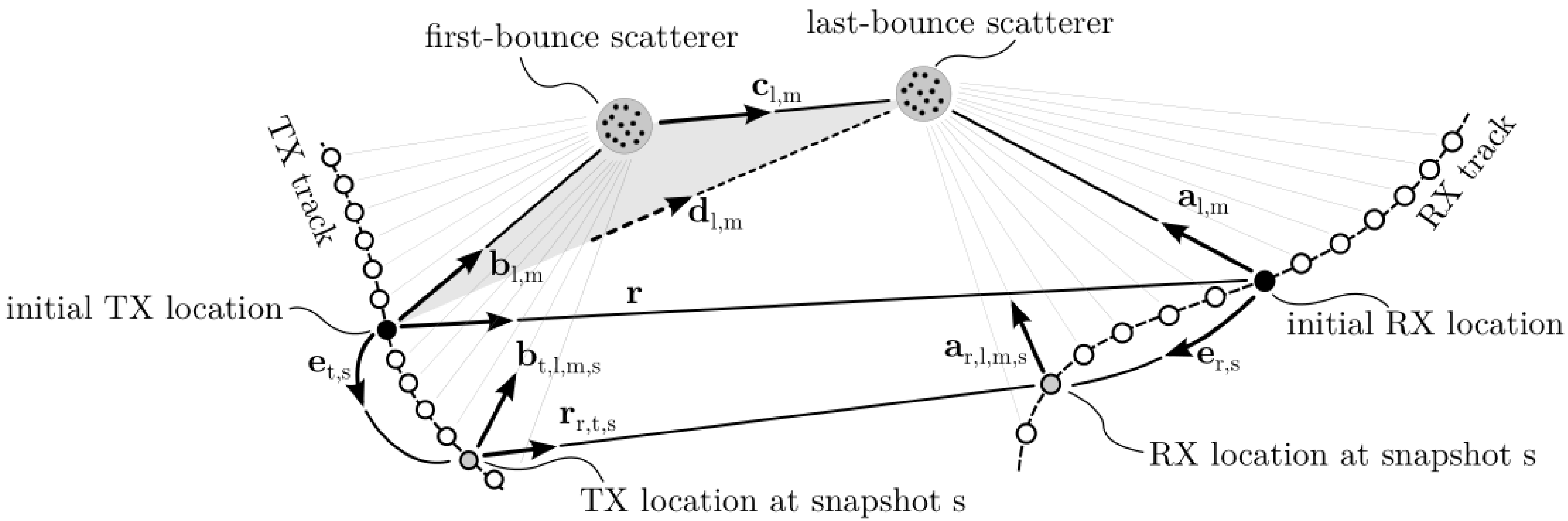


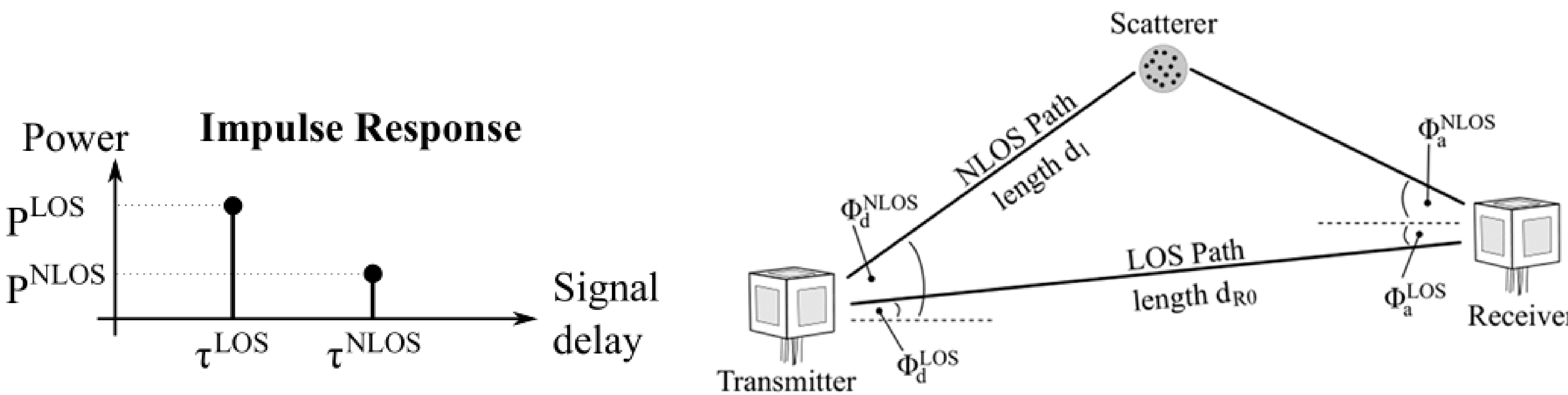
Two ray reflection

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



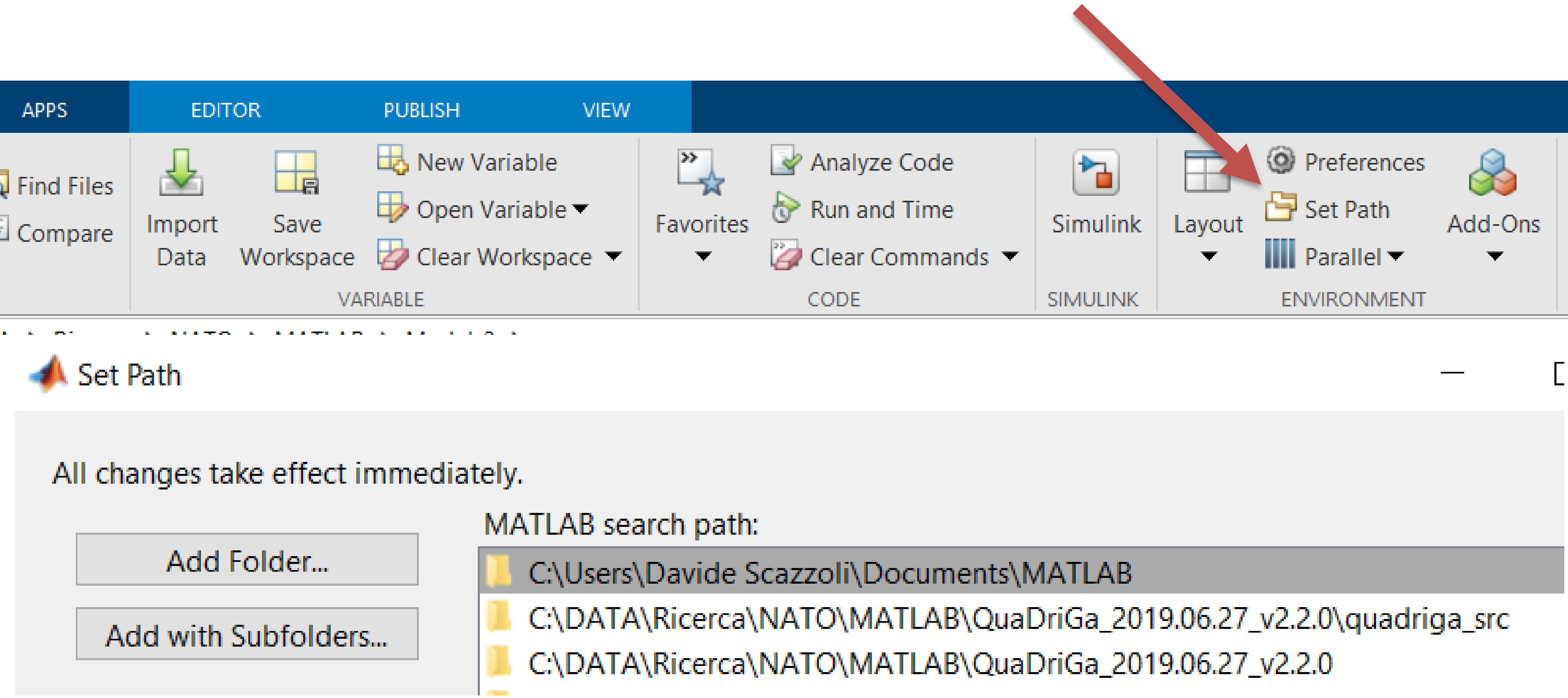
Quadriga Channel Model





Quadriga Channel Model

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



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

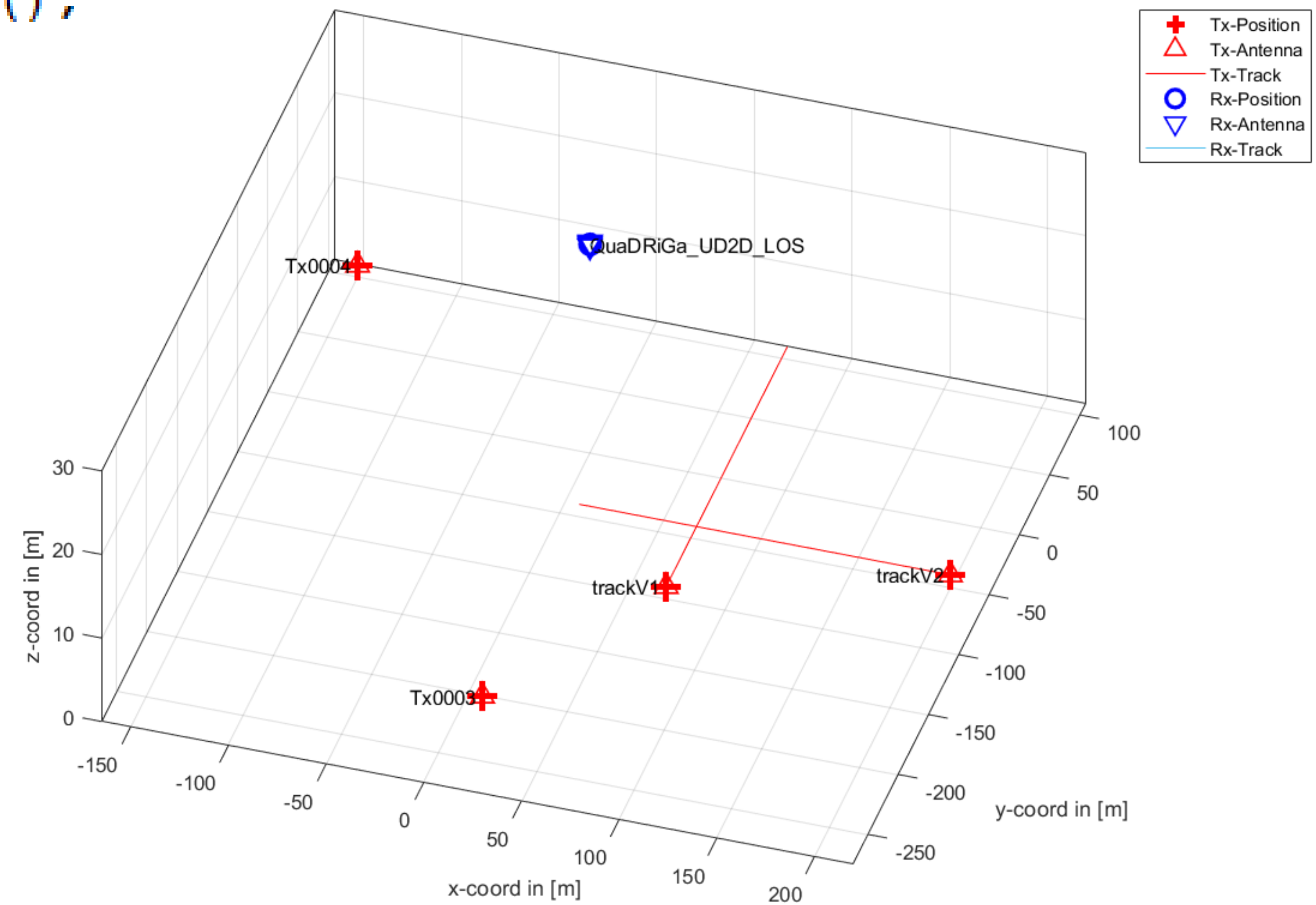
- Give the geometry information to the Quadriga Layout:

```
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

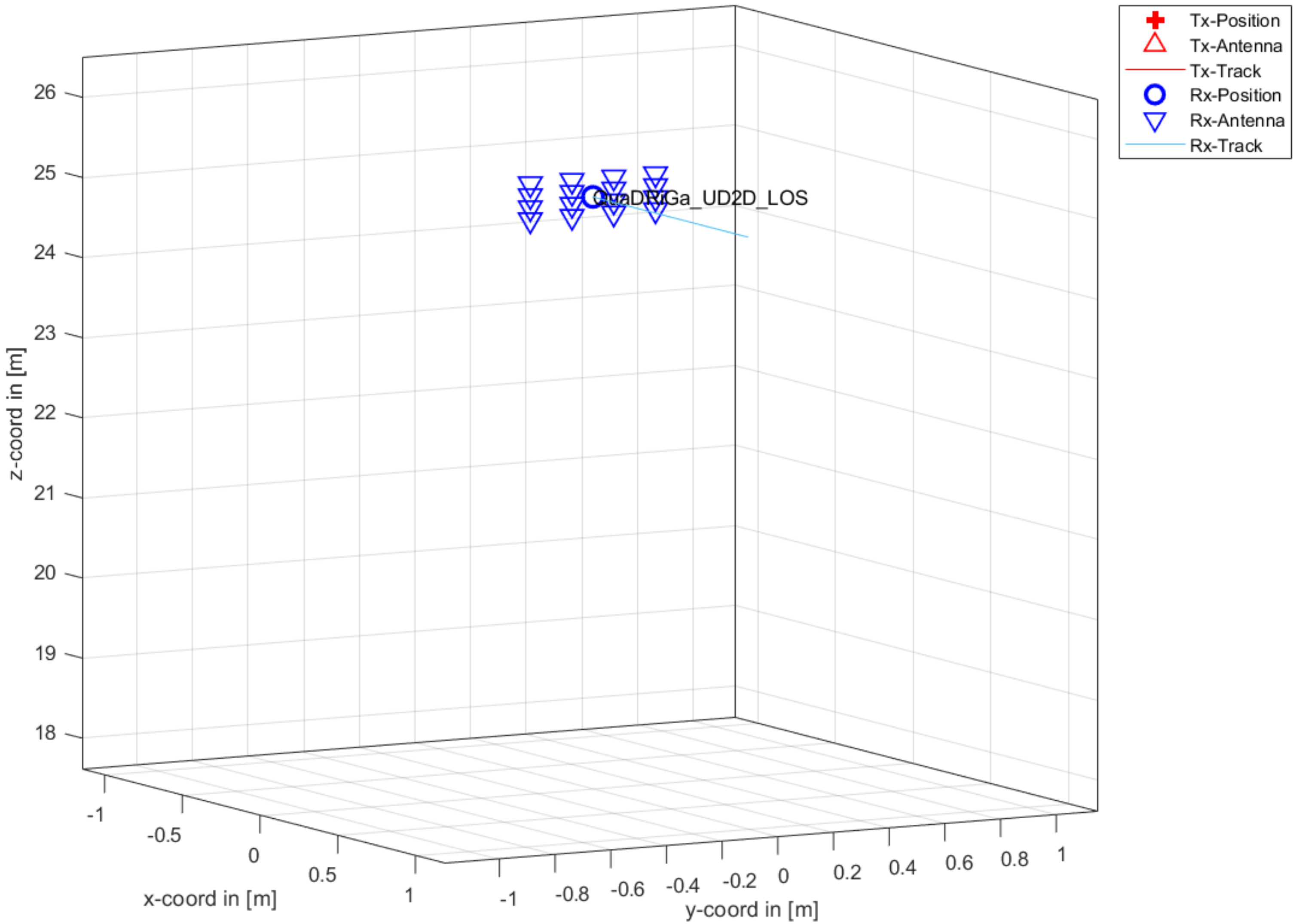
- Visualize the scenario:

```
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 qd_layout/get_channels (line 268)
  Generating channel builder objects
   - 1 builder, 4 channel segments
  Initializing random generators
  Generating parameters
  SSF Corr.     [oooooooooooooooooooooooooooooooooooooooooooooooooo]
  Channels      [oooooooooooooooooooooooooooooooooooooooooooooooooo]
  Merging       [oooooooooooooooooooooooooooooooooooooooooooooooooo]
  Formatting output channels - 4 channel objects
  Total runtime: 1 seconds
```

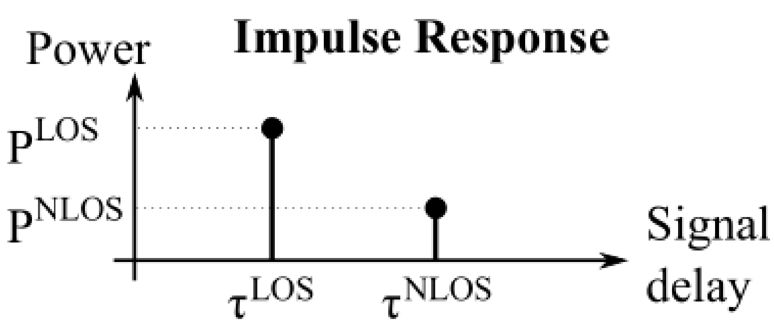
Quadriga Channel Model

- We get the following struct as an output:

Channel Coefficients

Channel Delays

```
chan =

  1×4 qd_channel array with properties:

    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 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.

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
```

Quadriga Channel Model

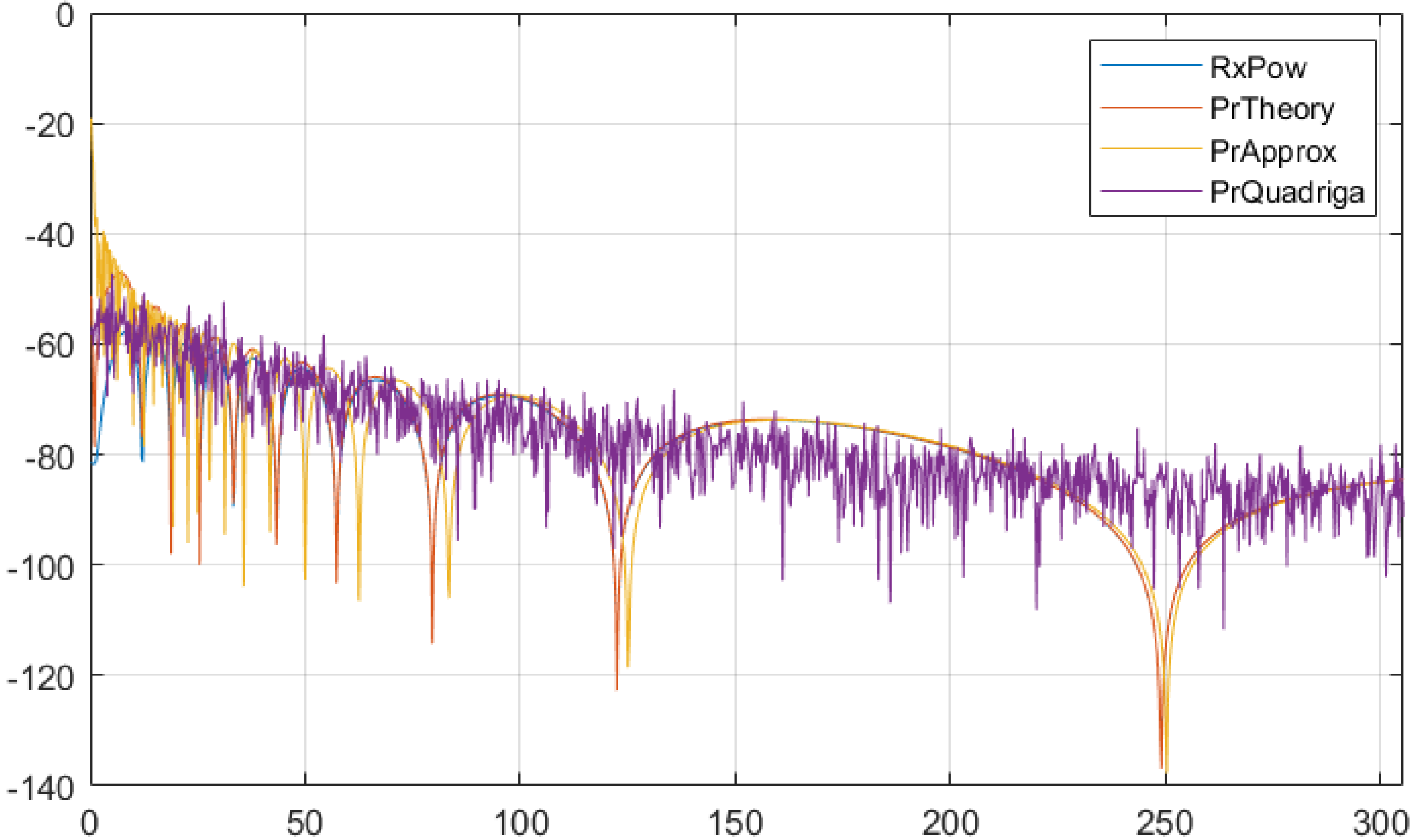
- To get the output we must perform the convolution:

```
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!

F.A.Q

- Can I use ____________________ ?

  Yes! Provided that:

  1. **<u>You know what it is doing.</u>**

  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`

F.A.Q

- How do I make two separate beams?

  1. Calculate two set of weights

  2. Multiply the [$N_{antennas}$ X $W_{aveform}$] channel output matrix by the two set of weights to get the two different outputs

Thank you!
Questions?