Technical Note on the Implementation of the Land Mobile Satellite Channel Model - Software Usage

Document-No: DLR-KN-FS-02-07
Issue No: 2
Issue Date: 06/07/07
Prepared by: Bernhard Krach
Dr. Alexander Steingass
Andreas Lehner

Abstract

This document contains information related to the software implementation of the Land Mobile Multipath Channel Model (LMSCM). It gives a detailed overview of the model parameters that are accessible for the user of the LMSCM and describes their effects within the artificial LMSCM scenery. Furthermore the document covers handling of the software implementation and describes model input and output variables.

Contents

1. Introduction

In 2002 the German Aerospace Center DLR performed a measurement campaign for the assessment of the Satellite Navigation Land Mobile Multipath Channel (see [1]). Based upon the obtained measurement data a channel model was developed (see [2]). The resulting model is based on both deterministic and stochastic processes within an artificial scenery that has to be parameterised by the model user.

The model was implemented in MATLAB using object orientated programming. For the use of the model the user has to initialize in a first step an object of the class *LandMobileMultipathChannel* with a valid paramter structure *ChannelParams* e.g. by using the following code:

```
ChannelObject=LandMobileMultipathChannel(ChannelParams);
```

After that step complex time-variant channel impulse responses can be created by using the associated method *generate*:

```
[ChannelObject,OutputVariables]=generate(ChannelObject,InputVariables);
```

The document is organized as follows: Chapter 2 introduces the parameters that have to be assigned within the structure *ChannelParams* and chapter 3 gives a detailed view on the input and output variables. Chapter 4 explains the demo script that is contained in the model implementation.

2. Channel Parameters

The channel parameters are used to parameterize the artificial scenery of the LMSCM. The artificial scenery, wherein the user moves, consists of obstacles, which are house fronts, trees and poles, which are shaped and placed within the scenery by statistical processes that can parameterized by the model user. Once shaped and placed all obstacles have a deterministic behaviour, except for the tree, whose treetop behaviour is modelled by a statistical process.

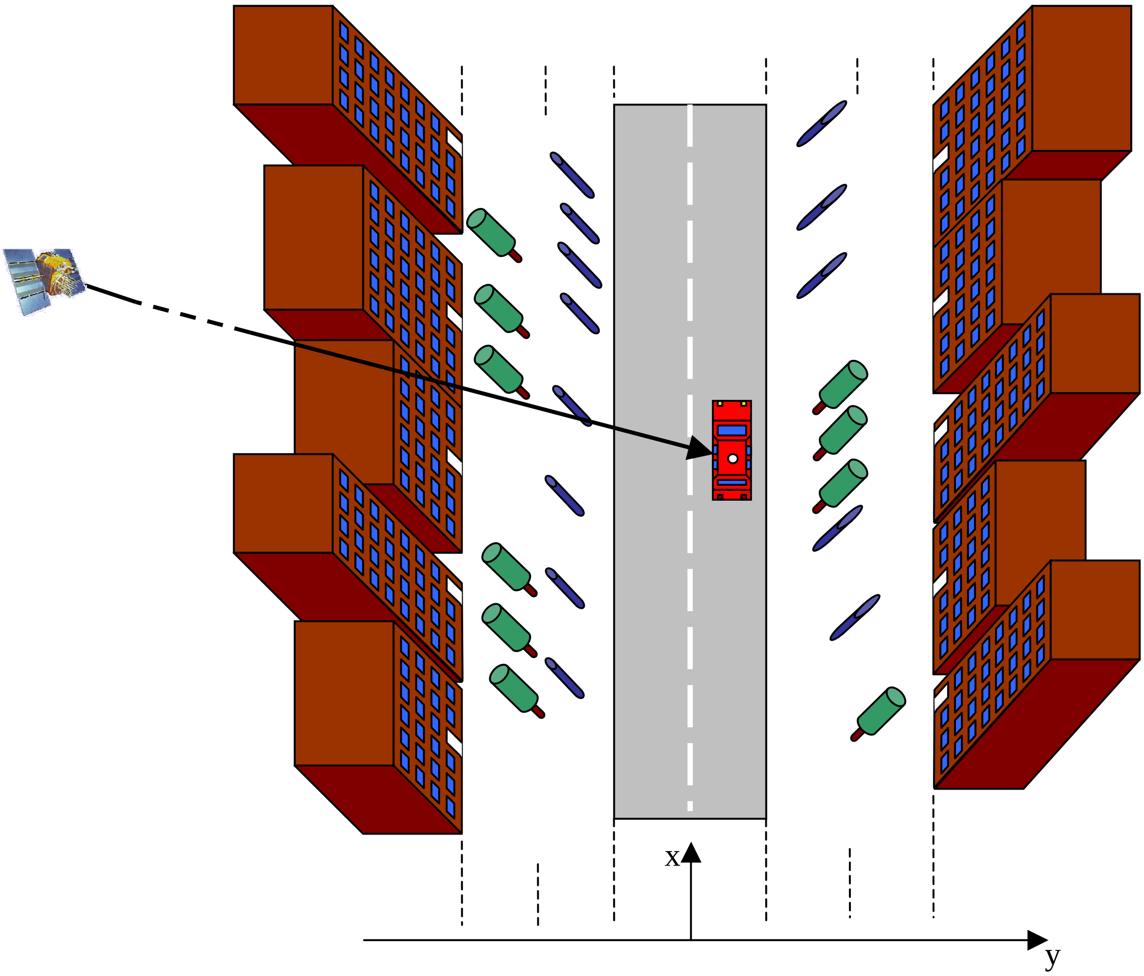

Figure 1: Artificial scenery

The *ChannelParams* struct has to be created as follows:

```
ChannelParams.Paramter1=ValueParamter1;
ChannelParams.Paramter2=ValueParamter2;
        :
        :
        :
```

The main idea behind the user movement within the artificial scenario is that a real user in most cases does move parallel along the road direction. Therefore the user heading in the model is always equal to the direction of the road. Consequently a movement along a turn in the road can be done by simply changing the user heading. The artificial scenarios will then automatically follow this turn.

The necessary model parameter names and their value ranges are listed in the following sections. The model user itself is responsible for setting all numerical parameter values in a physical reasonable range. The model implementation is matched to the parameter setting contained in the demo script in order to approximate the real world behaviour of a certain unique measured scenario as good as possible. There is no guarantee that for another set of parameters the model output describes the real world behaviour of a scenery corresponding to the chosen parameters in a reasonable way. Nevertheless it is expected without guarantee that slight variations of parameter values are possible without loosing too much model performance.

2.1. Shaping and Placement Processes

In general the statistic processes for the shaping and the placement of the obstacles are normal Gaussian distributions characterized by mean and variance

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Some of the shaping and placement processes in the model are bounded by a minimum and/or a maximum value, what means their probability density function is a conditional probability density of the Gaussian process specified by mean and variance, whereas the condition is that x lies within the specified interval:

$$p(x \mid x \in [x_{\min}, x_{\max}]) = \tilde{p}(x) = \begin{cases} 0 & x < x_{\min} \\ \frac{1}{C} \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} & x_{\max} \geq x \geq x_{\min} \\ 0 & x > x_{\max} \end{cases} \quad \text{with} \quad C = \int_{x_{\min}}^{x_{\max}} p(x) \mathrm{d}x$$

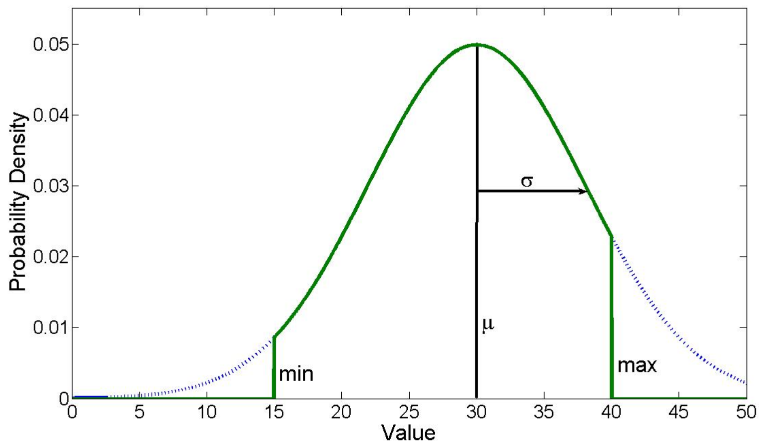

Figure 2: Obstacle shaping and placement processes

2.2. General Parameters

CarrierFreq	[Hz], [0 Inf[	e.g. GPS L1: 1.57542e9 Hz
SampFreq	[Hz], [0 Inf[	Has to be adjusted with maximum user speed in order to fulfill the sampling theorem
EnableDisplay	Logical: 0,1	Not used in the free version
EnableCIRDisplay	Logical: 0,1	Enable CIR display

2.3. Mode Parameters

The mode parameters specify the operation mode of the LMMCM implementation.

UserType	String: 'Car'	Other values reserved for future use
Surrounding	String: 'Urban' 'Suburban'	Other values reserved for future use
AntennaHeight	[m], [0 Inf[	Height of the Antenna
MinimalPowerdB	[dB],]-Inf Inf [	Echos below this Limit are not initialised

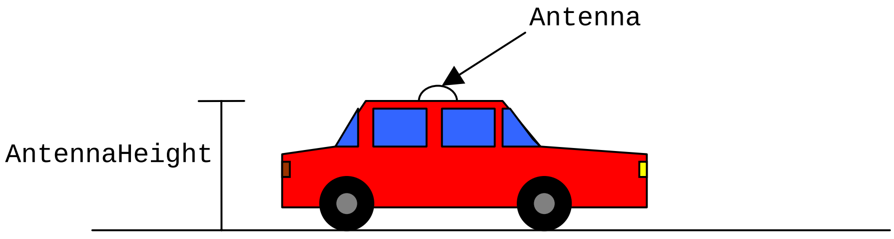

Figure 3: Antenna placement

2.4. User Parameters

DistanceFromRoadMiddle	[m], [0 Inf[	negative: Continental (right), positive: England (left)

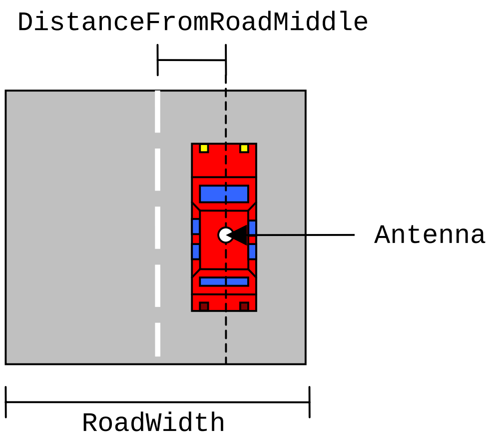

Figure 4: User parameters and antenna placement

2.5. Graphics Parameters

GraphicalPlotArea	[m],[0 Inf]	Not used in the free version
ViewVector	([m],[m]),[0 Inf[	Not used in the free version
RoadWidth	[m] ,[0 Inf]	Not used in the free version

2.6. Building Parameters

The building parameters set up the deterministic and statistical processes for building shaping and placement. There are two rows of buildings that can be switched on. The building rows ca be specified by a shaping process for house width and house height and a placement process, which is governed by a gap placing process. The Y-distance of the building rows is a fixed deterministic value.

BuildingRow1	Logical: 0,1	logical to switch Building Row right(heading 0 deg) on
BuildingRow2	Logical: 0,1	logical to switch Building Row left (heading 0 deg) on
BuildingRow1YPosition	[m]	
BuildingRow2YPosition	[m]	
HouseWidthMean	[m]	
HouseWidthSigma	[m]	
HouseWidthMin	[m]	
HouseHeightMean	[m]	
HouseHeightSigma	[m]	
HouseHeightMin	[m]	
HouseHeightMax	[m]	
GapWidthMean	[m]	
GapWidthSigma	[m]	
GapWidthMin	[m]	
BuildingGapLikelihood	[],[0,1]	Likelihood of a gap after a building

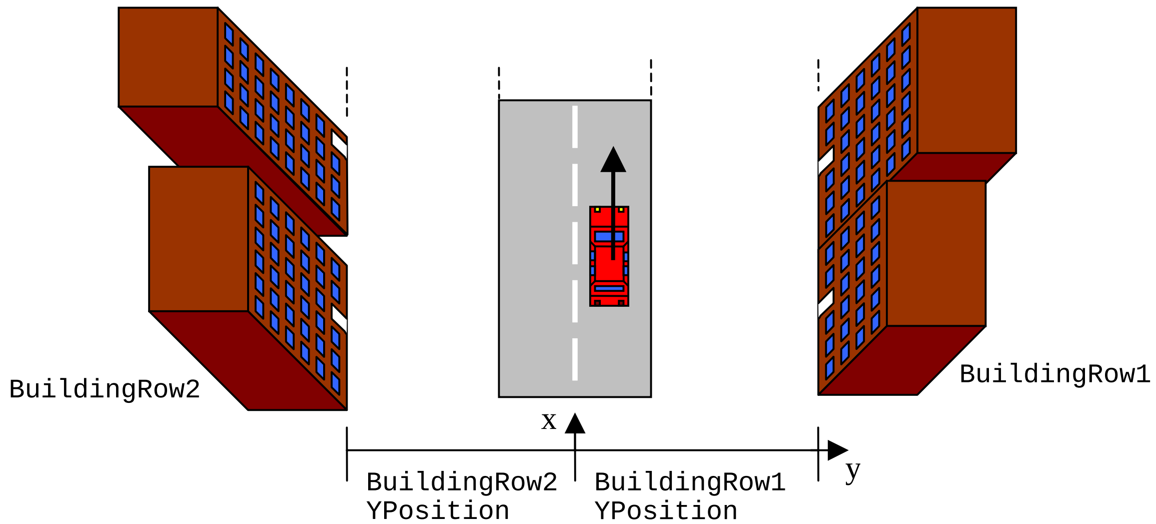

Figure 5: Building placement

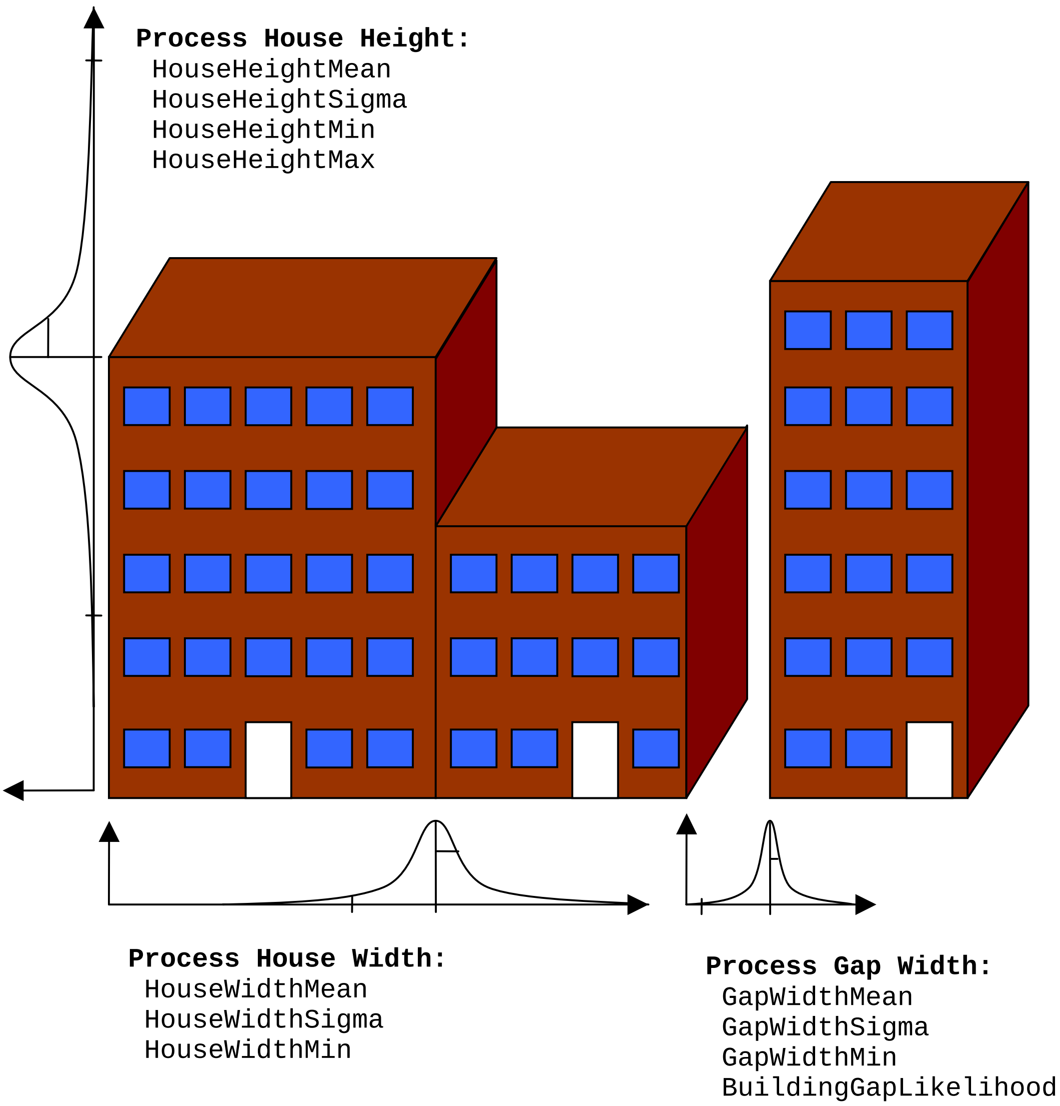

Figure 6: Building shaping and placement processes

2.7. Tree Parameters

Unlike the buildings all trees have the same deterministic shape. There a two rows of trees that can be switched on. The placement of both tree rows is controlled by separate statistical placement processes for each row respectively.

TreeHeight	[m],[0,Inf[	
TreeDiameter	[m],[0,Inf[	
TreeTrunkLength	[m],[0,Inf[	
TreeTrunkDiameter	[m],[0,Inf[	
TreeAttenuation	[dB/m],[0,Inf[	
TreeRow1Use	Logical: 0,1	logical switches tree row 1 on
TreeRow2Use	Logical: 0,1	logical switches tree row 2 on
TreeRow1YPosition	[m],[0,Inf[	
TreeRow2YPosition	[m],[0,Inf[	
TreeRow1YSigma	[m],[0,Inf[	
TreeRow2YSigma	[m],[0,Inf[	
TreeRow1MeanDistance	[m],[0,Inf[	
TreeRow2MeanDistance	[m],[0,Inf[	
TreeRow1DistanceSigma	[m],[0,Inf[	
TreeRow2DistanceSigma	[m],[0,Inf[	

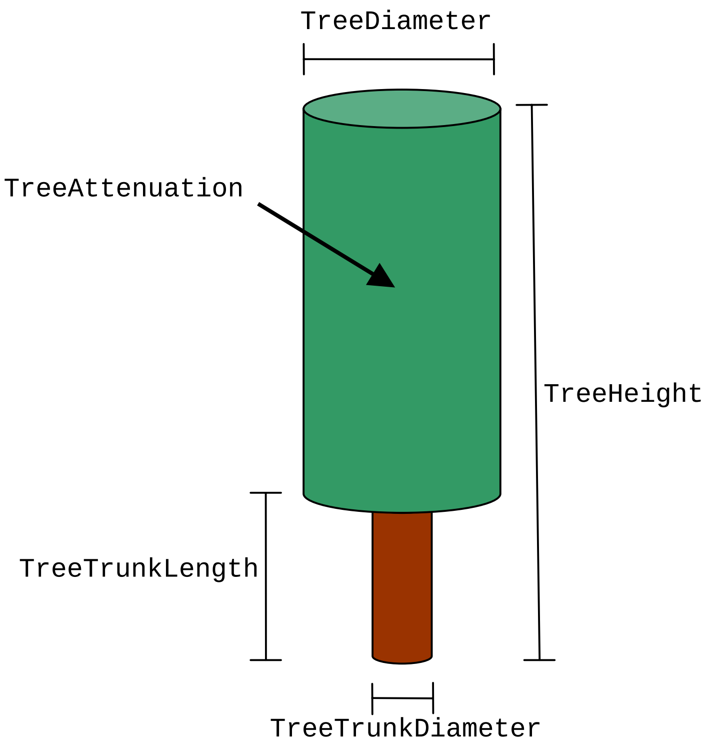

Figure 7: Tree parameters

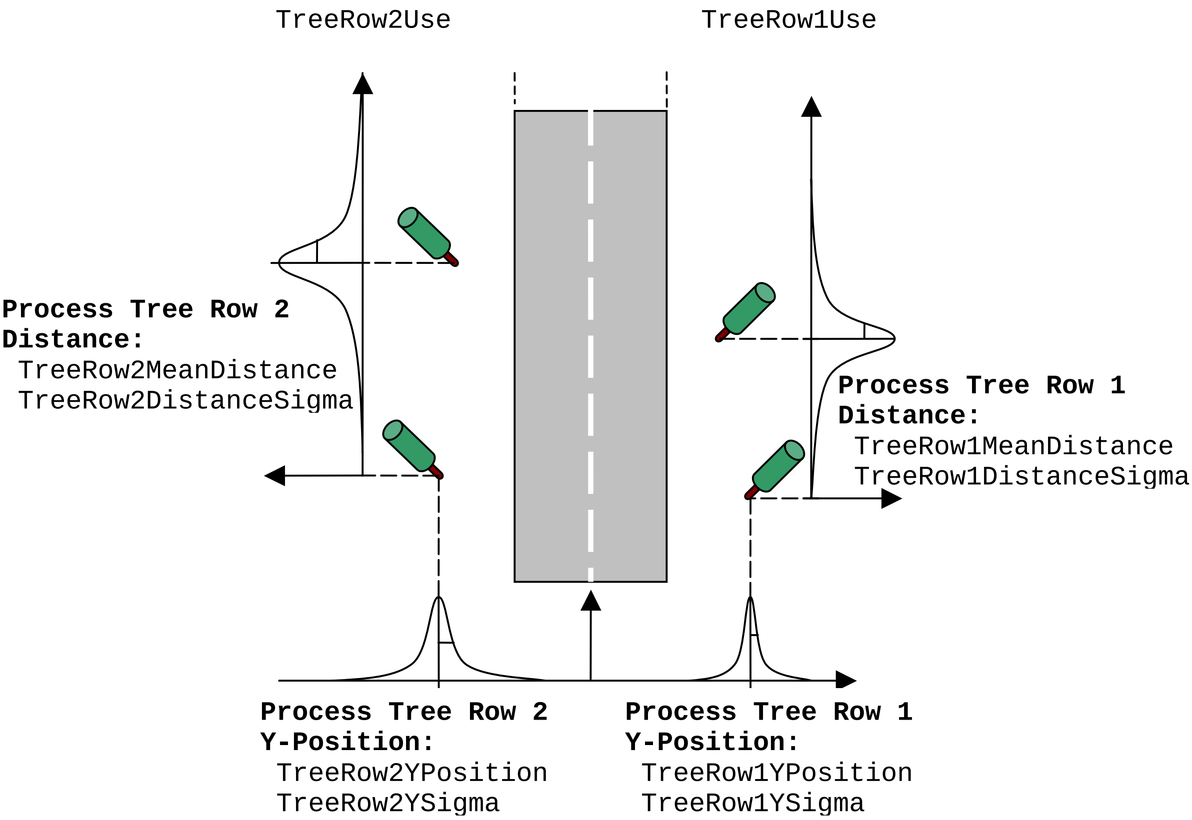

Figure 8: Tree placement processes

2.8. Pole Parameters

The handling of poles is similar to the trees. The shape is the same for all placed poles, there are two rows that can be switched on. The pole placement is controlled by separate statistical placement processes again.

PoleHeight	[m]	
PoleDiameter	[m]	
PoleRow1Use	Logical: 0,1	logical switches Pole row 1 on
PoleRow2Use	Logical: 0,1	logical switches Pole row 2 on
PoleRow1YPosition	[m]	
PoleRow2YPosition	[m]	
PoleRow1YSigma	[m]	
PoleRow2YSigma	[m]	
PoleRow1MeanDistance	[m]	
PoleRow2MeanDistance	[m]	
PoleRow1DistanceSigma	[m]	
PoleRow2DistanceSigma	[m]	

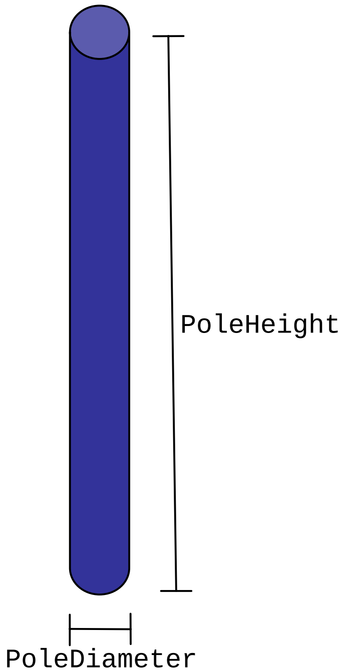

Figure 9: Pole parameters

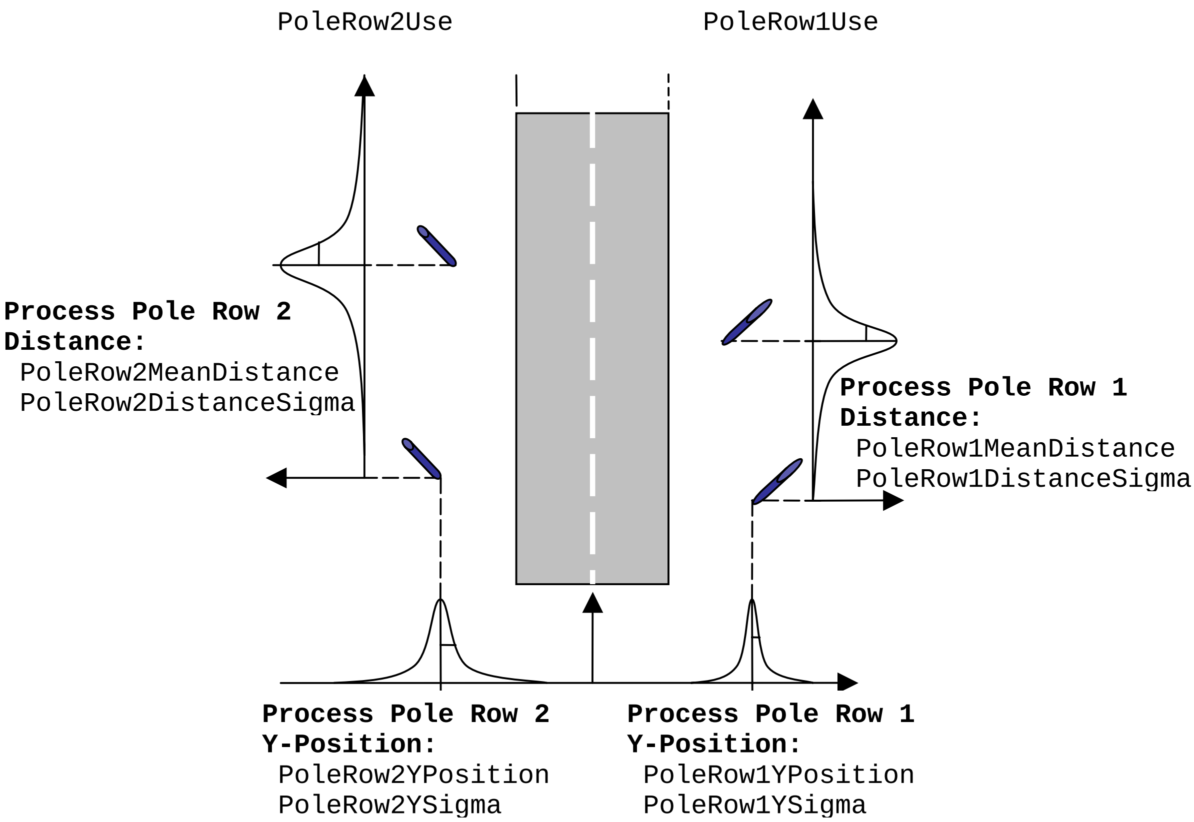

Figure 10: Pole placement processes

3. Model Handling

The instance of the class *LandMobileMultipathChannel*, here denoted by *ChannelObject* controls the processes that have been described within the previous section. The method *generate* requires a set of input variables and produces a set of output variables, which are specified now. The instance of the class itself is always the first input argument that has to be handed over to its method. The method gives the instance back as the first output argument. The syntax of the *generate* method is as follows:

```
[ChannelObject, ...
LOSCoeff,  LOSDelays,...
EchoCoeff, EchoDelays, EchoNumbers, ...
WayVec,    TimeVec]= ...
generate( ChannelObject,...
    ActualSpeed,  ActualHeading,...
    SatElevation, SatAzimuth);
```

3.1. Input Arguments

The angular input variables are defined in a North/East coordinate system, whereas the model output depends only on the angular difference between the heading and the satellite azimuth.

ChannelObject		LMMCM class instance
ActualSpeed	[m/s],[0 Inf[	User speed. Maximum user speed has to be adjusted to the chosen sampling rate in order to fulfill the sampling theorem
ActualHeading	[degrees],[0 360[	User heading relative to north direction
SatElevation	[degrees],[0 90]	Elevation of satellite
SatAzimuth	[degrees],[0 360[	Azimuth of satellite relative to north direction

The maximum user speed has to be handled carefully with respect to the model sampling rate, because the sampling theorem can be violated by to high user speed values. With c_0 being the speed of light, f_s the sampling rate, and f_c the selected carrier frequency the maximum user speed value that may not be exceeded is

$$v_{\max} \leq \frac{c_0}{2} \cdot \frac{f_s}{f_c}$$

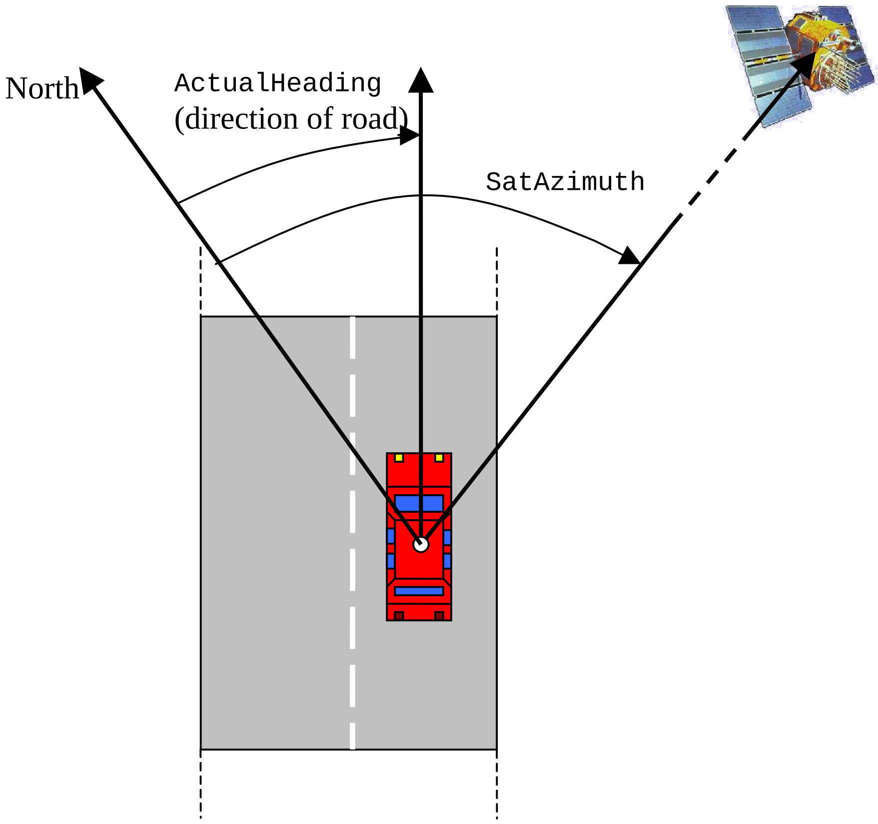

Figure 11: Input Parameters, XY-plane visualization

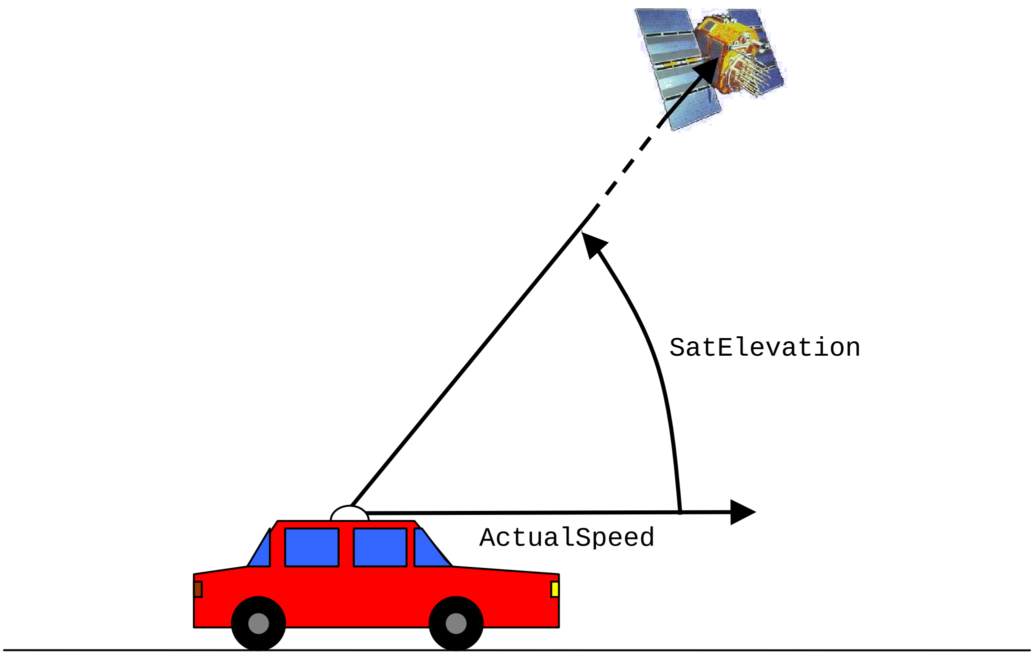

Figure 12: Input Parameters, XZ-plane visualization

3.2. Output Arguments

The output of the LMSCM implementation is a complex time-variant channel impulse response (CIR) of the form

$$h(t,\tau) = \sum_{k=1}^{N(t)} \underline{a_k}(t) \cdot \delta(\tau - \tau_k(t))$$

Every time the method *generate* is called the impulse response is calculated for the new time step. As echoes in general have duration of many time instances, every time the impulse response is calculated a vector containing the echo numbers is output from the model, which allows the identification of certain echo. Every time an echo is created within the model it is tagged with a unique number. The echo number is taken from a counter, which is increased every time when an echo is created. The elements within the output vectors correspond to each other, what means the output values

`EchoCoeff(i), EchoDelays(i), EchoNumbers(i)`

belong to each other and specify one tap. The LOS path is handled in a special way. Due to diffraction effects that can occur at building obstacles the LOS is split into two or three separate paths for certain geometries. Furthermore the LOS delay differs from zero if it is diffracted at a house front. The LOS output vectors correspond to each other again, so that

`LOSCoeff(i), LOSDelays(i)`

specify a LOS tap. The output arguments *WayVec* and *TimeVec* are vectors whose size increases by one every time the method *generate* is called. They contain the history of time and history of the travelled user way.

ChannelObject		LMMCM class instance
LOSCoeff	[1x1],[1x2],[1x3]	1,2 or 3 element vector containing LOS coefficients
LOSDelays	[1x1],[1x2],[1x3]	1,2 or 3 element vector containing LOS delay values
EchoCoeff	[1xM]	M element vector containing echo coefficients
EchoDelays	[1xM]	M element vector containing echo delay values
EchoNumbers	[1xM]	M element vector containing echo numbers
WayVec	[1xT]	Vector containing the history of the travelled way from the model initialization to the current time instance
TimeVec	[1xT]	Vector containing the history of time instance

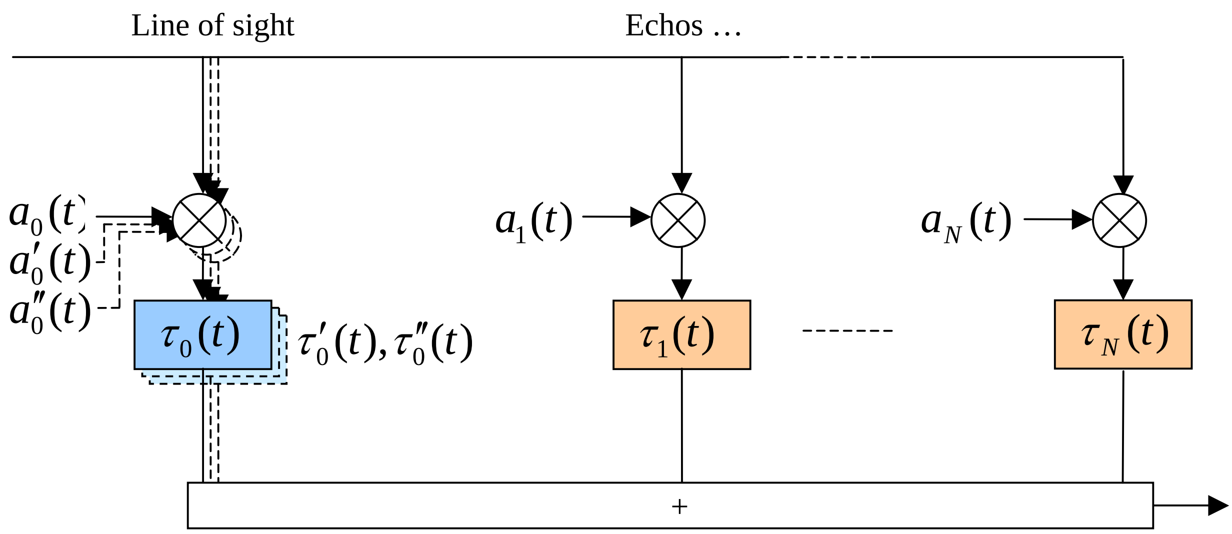

Figure 13: LMSCM CIR output, tap delay line

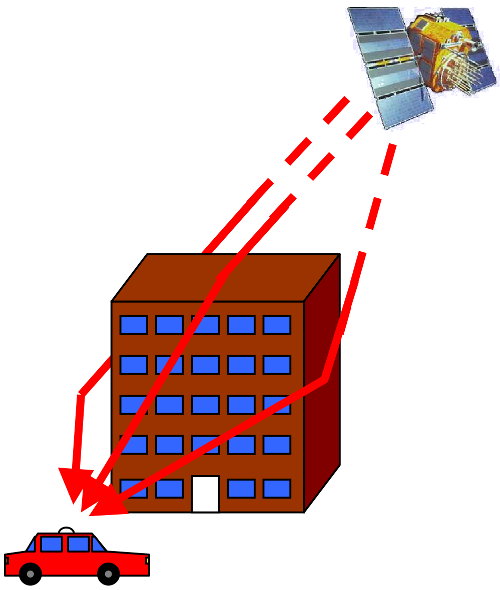

Figure 14: Splitting of the LOS path into three separate paths due to diffraction at a building obstacle

4. Demo Script

This chapter explains the demo script for the urban car environment, that is provided with the LMSCM implementation. Important steps concerning model handling are highlighted:

```
% LandMobileMultipathChannel_Demo_UrbanCar
% Version 1.10
```

Clear workspace and screen.

```
close all
clear all
clc
```

Set sampling frequency, maximum demo speed, satellite elevation and azimuth for the demo and the length of the simulation in time samples.

```
Parameters.SampFreq=300;        % Hz
Parameters.MaximumSpeed=50;     % km/h
Parameters.SatElevation=30;     % Deg
Parameters.SatAzimut=-45;       % Deg (North == 0, East == 90,
                                % South == 180, West == 270)
Parameters.NumberOfSteps=5000;

FontSize=12;
```

Set parameters as described within the previous chapter:

```
% ---- General Parameters ----

ChannelParams.CarrierFreq=1.57542e9;     % Hz
ChannelParams.SampFreq=Parameters.SampFreq;
ChannelParams.EnableDisplay=1;           % 3D visualization is not
                                         % available in the free version
ChannelParams.EnableCIRDisplay=1;        % enables CIR display

% ---- Mode Parameters ----

ChannelParams.UserType = 'Car';
ChannelParams.Surrounding = 'Urban';
ChannelParams.AntennaHeight = 2;         % m Height of the Antenna
ChannelParams.MinimalPowerdB=-40;        % Echos below this Limit are not
                                         % initialised

% ---- UserParameters ---

ChannelParams.DistanceFromRoadMiddle=-5; % negative: continental (right),
                                         % positive England (left)
```

```
% ---- Graphics Parameters ---

ChannelParams.GraphicalPlotArea=50;      %
ChannelParams.ViewVector = [-60,20];     % 3D visualization is not
                                         % available in the free version
ChannelParams.RoadWidth = 15;            %

% --- Building Params ---

ChannelParams.BuildingRow1=1;            % logigal to switch Building Row
                                         % right(heading 0 deg) on
ChannelParams.BuildingRow2=1;            % logigal to switch Building Row
                                          % left (heading 0 deg) on
ChannelParams.BuildingRow1YPosition=-12; % m
ChannelParams.BuildingRow2YPosition=12;  % m

ChannelParams.HouseWidthMean=22;         % m
ChannelParams.HouseWidthSigma=25;        % m
ChannelParams.HouseWidthMin=10;          % m
ChannelParams.HouseHeightMin=4;          % m
ChannelParams.HouseHeightMax=50;         % m
ChannelParams.HouseHeightMean=16;        % m
ChannelParams.HouseHeightSigma=6.4;      % m
ChannelParams.GapWidthMean=27;           % m
ChannelParams.GapWidthSigma=25;          % m
ChannelParams.GapWidthMin=10;            % m
ChannelParams.BuildingGapLikelihood=0.18;% lin Value

% --- Tree Params ---

ChannelParams.TreeHeight = 8;            % m
ChannelParams.TreeDiameter = 5;          % m
ChannelParams.TreeTrunkLength=2;         % m
ChannelParams.TreeTrunkDiameter=.2;      % m

ChannelParams.TreeAttenuation = 1.1;     % dB/m

ChannelParams.TreeRow1Use=1;             % logical switches tree row 1 on
ChannelParams.TreeRow2Use=1;             % logical switches tree row 2 on

ChannelParams.TreeRow1YPosition=-8;      % m
ChannelParams.TreeRow2YPosition=8;       % m

ChannelParams.TreeRow1YSigma=2;          % m
ChannelParams.TreeRow2YSigma=2;          % m

ChannelParams.TreeRow1MeanDistance=60;   % m
ChannelParams.TreeRow2MeanDistance=40;   % m

ChannelParams.TreeRow1DistanceSigma=20;  % m
ChannelParams.TreeRow2DistanceSigma=20;  % m

% --- Pole Params ---

ChannelParams.PoleHeight = 10;           % m
ChannelParams.PoleDiameter = .2;         % m
```

```
ChannelParams.PoleRow1Use=1;            % logical switches Pole row 1 on
ChannelParams.PoleRow2Use=0;            % logical switches Pole row 2 on

ChannelParams.PoleRow1YPosition=0;      % m
ChannelParams.PoleRow2YPosition=10;     % m

ChannelParams.PoleRow1YSigma=1;         % m
ChannelParams.PoleRow2YSigma=1;         % m

ChannelParams.PoleRow1MeanDistance=25;  % m
ChannelParams.PoleRow2MeanDistance=10;  % m

ChannelParams.PoleRow1DistanceSigma=10; % m
ChannelParams.PoleRow2DistanceSigma=10; % m

% ------------ Initial Settings -------------
% - Anything Below here must not be changed -
% -------------------------------------------
```

Set constants and adjust speed:

```
Co=2.99e8; % Speed of Light

MaximumPossibleSpeed=Co*Parameters.SampFreq/ChannelParams.CarrierFreq/2; %
To fulfill the sampling Theorem
SamplingTime=1/Parameters.SampFreq;

% --- Initialising the channel object ---
```

Initialize *LandMobileMultipathChannel* object:

```
pause(1)
disp('Initialising the channel ...')
TheChannelObject=LandMobileMultipathChannel(ChannelParams);

TimeVec=0;
ComplexOutputVec=[];

% --- Specify power and delay bins for output statistics ---
```

Specify parameters for output statistics:

```
pwrvec = [0:-1:-30];           % power bins in dB
dlyvec = [0:10e-9:500e-9];     % delay bins in s

PowerDelayProfile(1:length(pwrvec),1:length(dlyvec)) = 0;   % allocate
                                                             % memory
pwrstp = (pwrvec(end)-pwrvec(1))/(length(pwrvec)-1);        % get step size
dlystp = (dlyvec(end)-dlyvec(1))/(length(dlyvec)-1);        % get step size

% --- start simulation ---
```

Start simulation:

```
h = waitbar(0,'Simulation running ...');
```

Initialize CIR display, if enabled:

```
if ChannelParams.EnableCIRDisplay

    % --- init CIR figure ---

    hh = figure;
    subplot(211)
    xlabel('delay in s')
    ylabel('power in dB')
    axis([-2e-8,40e-8,0,50])
    set(get(hh,'Children'),'YTickLabel',[-40 -30 -20 -10 0 10]);
    hold on
    grid on
    plot (0,0,'r')
    plot (0,0)
    legend ('LOS Paths','Echo Paths')

    subplot(212)
    xlabel('delay in s')
    ylabel('phase in rad')
    axis([-2e-8,40e-8,-pi,pi])
    hold on
    grid on

end
```

Iterate until number of steps is reached:

```
for dhv=1:Parameters.NumberOfSteps
```

Calculate time stamp for current iteration:

```
    TimeVec(end)=dhv/Parameters.SampFreq;
```

Calculate model input values for demo (drunken driver example):

```
    % --- "drunken" driver movement example ---

    ActualSpeed=Parameters.MaximumSpeed/2/3.6*(1+sin(TimeVec(end)/3));
    SpeedVec(dhv)=ActualSpeed;                % m/s
    ActualHeading=20*sin(TimeVec(end)/3);     % Deg (North == 0, East == 90,
                                              % South == 180, West == 270)




    % --- generate CIR ---
```

Call *generate* method of *LandMobileMultipathChannel* instance:

```
[TheChannelObject,LOS,LOSDelays,ComplexOutputVec,DelayVec,EchoNumberVec,...
WayVec(dhv),TimeVec(dhv)]=generate(TheChannelObject,ActualSpeed,...
ActualHeading,Parameters.SatElevation,Parameters.SatAzimut);

    waitbar(dhv/Parameters.NumberOfSteps,h)
```

Bin output of current time step:

```
% --- binning LOS ---

for sfg = 1:length(LOSDelays)

    dlybin = round(LOSDelays(sfg)/dlystp) + 1;
    pwrbin = round(20*log10(abs(LOS(sfg)))/pwrstp) + 1;

    if pwrbin<=length(pwrvec) & pwrbin>0 & dlybin<=length(dlyvec)
        PowerDelayProfile(pwrbin,dlybin) = ...
        PowerDelayProfile(pwrbin,dlybin) + 1;
    end

end

% --- binning echoes ---

for sfg = 1:length(DelayVec)

    dlybin = round(DelayVec(sfg)/dlystp) + 1;
    pwrbin = round(20*log10(abs(ComplexOutputVec(sfg)))/pwrstp) + 1;

    if pwrbin<=length(pwrvec) & pwrbin>0 & dlybin<=length(dlyvec)
        PowerDelayProfile(pwrbin,dlybin) = ...
        PowerDelayProfile(pwrbin,dlybin) + 1;
    end

end
```

Display CIR, if enabled:

```
    if ChannelParams.EnableCIRDisplay

        % --- display CIR ---

        figure(hh);
        subplot(211)
        cla
        Time = dhv/Parameters.SampFreq;
        title(['Channel Impulse Response, T = ',num2str(Time,'%5.2f'),...
               ' s, v = ',num2str(ActualSpeed*3.6,'%4.1f'),' km/h'])
        stem(LOSDelays,40 + 20*log10(abs(LOS)),'r');
        stem(DelayVec,40 + 20*log10(abs(ComplexOutputVec)));

        subplot(212)
        cla

        stem(LOSDelays,angle(LOS),'r');
        stem(DelayVec,angle(ComplexOutputVec));

    end

end%
close(h);
```

Visualize output as Power Delay Profile:

```
% --- calculate probability density function ---

PowerDelayProfile = PowerDelayProfile/sum(sum(PowerDelayProfile));

% --- display PowerDelayProfile ---

figure
surf(dlyvec,pwrvec,10*log10(PowerDelayProfile+eps),'LineStyle','none',...
     'FaceColor','interp','EdgeLighting','phong');
caxis([-70,-10]);
view(2)
xlabel('delay in s')
ylabel('power in dB')
title('power delay profile - probability density function')
hc = colorbar;
set(hc,'YLim',[-70,-10])
clear newYTic YTic
YTic = get(hc,'YTickLabel');
axes(hc);
ax = axis;
dta = (ax(3)-ax(4))/(size(YTic,1)-1);
for kk = 1:size(YTic,1)
    oldYTic(kk,:) = [' '];
    newYTic(kk,:) = ['10^',num2str(str2num(YTic(kk,:))/10),''];
    text(1.2,ax(3)-dta*(kk-1),['10^{',num2str(str2num(YTic(kk,:))/10), ...
         '}'], 'interpreter','tex','horizontalalignment','left',...
         'verticalalignment','middle');
end
set(hc,'YTickLabel',oldYTic);

% ---------------------------------

disp(' ');
disp('Simulation finished');
```

5. References

[1] Alexander Steingaß, Andreas Lehner "Measuring the Navigation Multipath Channel – A Statistical Analysis" ION GPS 2004 Conference, September 21-24, 2004 Long Beach, California USA

[2] Alexander Steingaß, Andreas Lehner "A Channel Model for Land Mobile Satellite Navigation" GNSS 2005 Conference, July 19-22, 2005 Munich, Germany

6. List of Figures

7. Acronyms

CIR	Channel Impulse Response
GPS	Global Positioning System
LMSCM	Land Mobile Satellite Channel Model
LOS	Line-of-sight