Development of a Matlab Toolbox for Mobile Fading Channel Simulators

by

Farouk Dhahbi

Thesis in partial fulfilment of the degree of
Master in Technology in
Information and Communication Technology

Agder University College
Faculty of Engineering and Science

Grimstad
Norway

May 2007

Abstract

Knowledge of channel behaviours in mobile radio communication is extensively recommended for the study of transmitter/receiver performances. Our intention in this master's thesis is to develop various kinds of mobile fading channel simulators using Matlab.

The developed channel simulators were combined into a user-friendly toolbox from which users can easily select well-known channel models to test and to study the performance of mobile communication systems. The toolbox can be integrated into the new release of Matlab software.

The toolbox contains channel simulators for spatial shadowing processes, MIMO channels, frequency-selective channels, multiple uncorrelated Rayleigh fading channels, etc. Recent new algorithms proposed to compute the model parameters of the channel simulators were also implemented in the toolbox to enable the parameterization of the channel simulators under specific propagation conditions.

Finally, we developed a set of test procedures, such as the autocorrelation function ACF, average duration of fades ADF, the probability density function PDF, and the level-crossing rate LCR etc., in order to test and to confirm the correctness of the implemented channel simulators.

Keywords: Matlab toolbox, Mobile fading channel simulators, Parameter computation methods, MIMO channels.

Preface

This thesis project results from the research work pursued on behalf of Pätzold's group at HiA University. It is based on the project tasks done by Ph.d and masters students. The previous works serve as support for my thesis and stand as a starting point for the required tasks.

The thesis is basically addressed to researchers or students who have an interest in subjects dealing with mobile fading channels. In addition, it is addressed to Matlab users, since there lacks a toolbox which confines the mobile radio issues.

I would like to send my gratitude and thanks to my direct supervisor, Mr. Pr. Matthias Pätzold, for his guidance and advice. I would especially like to thank my co-supervisor, Mr. Pr. Frank Reichert, who has assisted me with technical advice and recommendations. I would sincerely like to thank Mr. Haugstad (Ph.d student) and Mr. Kun Yang (Master's student); without their help this work couldn't be achieved within the project timeframe. Finally, I am grateful to the administration, Mr. Stein Bergsmark, head of master's studies, Ms. Sissel Andreassen, Mr. Tor Erik Kristiansen and Mr. Pål of the international office, as well as my teachers Mr. Simenson, Mr. Ole Christopher, and Mr. Jan Perterson..

Table of Contents

List of figures

CHAPTER 1 - INTRODUCTION

Recently, an important emergence of mobile systems has pushed researchers to supply new ways and methods for achieving good performance and high communication quality. Since the number of users of such systems is increasingly growing, a compromise between quality and capacity should be taken in consideration. The revolution of many mobile communication sectors such as mobile radio communication, which is of interest in this thesis, has been significant in our daily lives. Many factors cause such revolution. The improvement of modulation schemes reached an acceptable BER, leaving the systems more reliable and efficient. Among these schemes, the appearance of modern digital modulations, for instance x-ASK, x-PSK, x-FSK, QAM-x, PAM-x, etc., has had an impact on the sophistication of communication quality; it keeps the same performance and reduces the user data rate. Consequently, the number of users goes up.
The appearance of mobile radio communication as a functional domain began approximately 40 years ago. The first generation of mobile systems was entirely based on analogue modulations as well as techniques of transmission. The RTM system is considered a typical example of this generation. Yet the capacity of the system regarding the number of users is limited and it could not satisfy all demands. Therefore, a new generation of mobile radio systems appeared. It was called the second generation of mobile radio systems, and was characterized by the introduction of digitalization of the entire network. Nowadays, these systems are still functional and respond to user's needs. However, the demands of users on the quality and efficiency of the networks gives reason to invent new systems. The so-called third generation mobile radio systems, though, are not practically ready for users in many European countries and the rest of the world, with the exception of Japan. Potentially, it brings many new features, such as high internet quality, real-time video calls, etc. In the future, a universal new system, called the fourth generation mobile radio systems, will be of interest to scientists. The goal of this generation is the consolidation of broadband mobile services which require a frequency range of 100 GHz [1].
Mobile radio communication was investigated theoretically and experimentally in order to build up these systems. The channel modelling is an important part of mobile radio communication. Nevertheless, a mediocre model leads to inefficient systems. Therefore, investing in research and cases studies should be more prevalent.

1.1 BASIC CONCEPTS

The modelling of channels has an important place in mobile radio communication areas. Obviously, an acceptable model avoids unexpected degradations whenever it goes to fabrication. In addition, especially in mobile radio communication, knowledge of the channels is more than compulsory. The diagram below depicts a simple transmission chain.

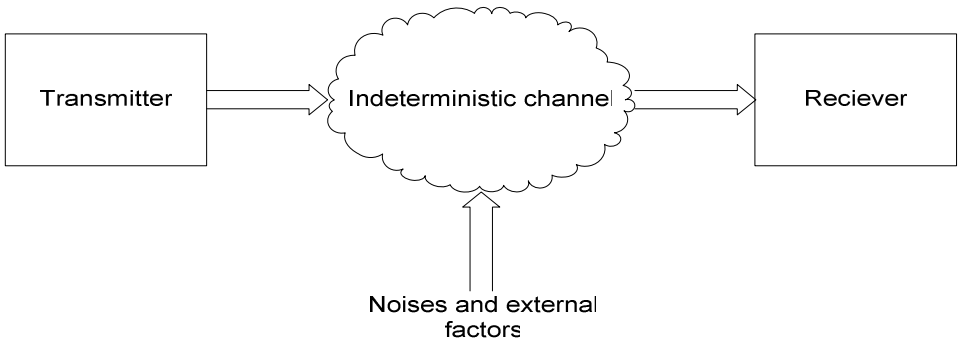

Figure 1-1: A simple transmission chain

In order to get a fitting model, following some scientifically approved steps is required. First of all, a concrete phenomena or case must be present. Then, measurements should be taken in order to specify the physical aspects. Finally, we compare the simulation model with the reference model. Thus, we try to minimize the error modelling as much as possible to set the best match. The Figure 1-2 stresses the relationships in modelling areas.

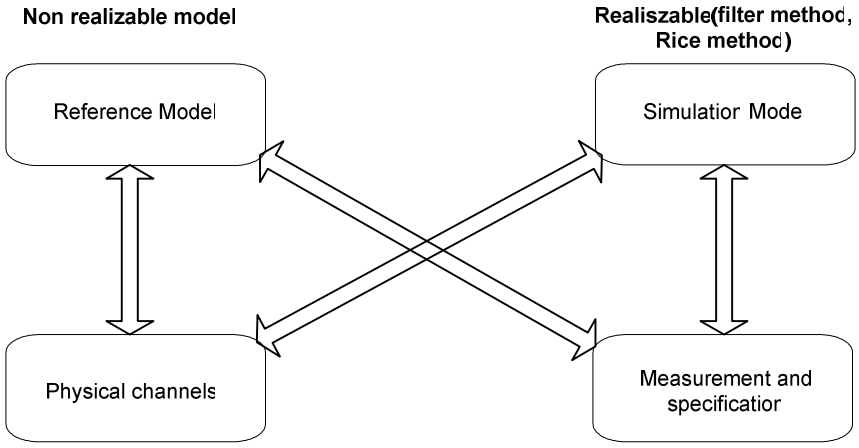

Figure 1-2: The different steps for channel modelling

After observing the natural phenomena, we analyse and look for ideas which can be used to solve the problems; for instance, mathematical and physical backgrounds could be important. The modelling step requires the reference model, exact model and simulation model in order for us to check the correctness of the model against the reference model.
The modelling fields have a big impact on the building-up of the whole system since the implementation of the system is based on the mathematical model. Therefore, the performance of the model controls the performance of the entire system.
Mainly, most channels model simulators are based on the sum-of-sinusoids principle which is first introduced by Rice than developed later. Starting from the idea that each signal can be expressed under sum-of-sinusoids, we assumed that any model type is able to be implemented following the sum-of-sinusoids principle. In mobile channel modelling, we begin from fundamental channel models which are Rayleigh and Rice models. Thus, these two latter are

the result of the sum of two Gaussian processes. However, the Gaussian process is structured as follow:

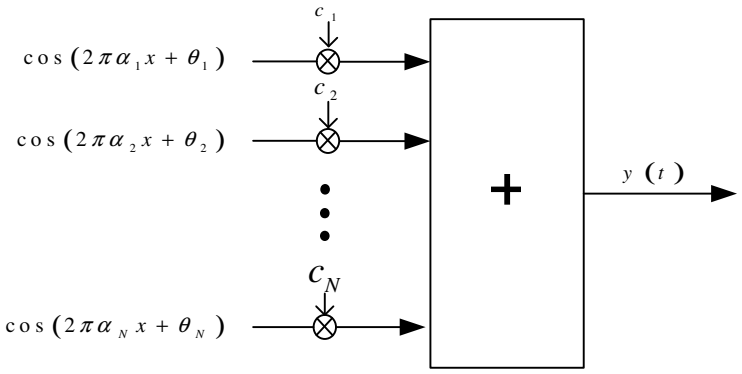

Figure 1-3: An example of sum-of-sinusoids principle

As we know, when using the Rice method, we assume that number of sinusoids is infinite. But, in real-world we consider N as small as possible aiming to let the implementation of the simulation realizable. This principle can be applied on any kind of simulation model. In addition, the filter method is another method for the implementation of channel models that exists in literature but we didn't exploit it since it exhibits some drawbacks like for instance the exigency of ideal filter. Further explanation of these concepts is noticed in [1] chapter 4. Throughout this thesis, we just consider the sum-of-sinusoids principle for developing all channel fading simulators.

1.2 REQUIREMENTS

In this thesis, we require some papers and publications which contain the results of research work done over the last few years. Among these articles we used the listed papers and publications in [2]-[19].
In order to implement the toolbox fitted for mobile channels modelling, an exact solution should be presented. On the other hand, we must ensure the correctness of our implementation. Therefore, our toolbox can be integrated into the new release of Matlab.
The toolbox devoted to mobile channels modelling is missing in the Matlab environment. Within this thesis, we provide some useful programs presented in m-file format designed for all tools needed to model any kind of channel.

1.2.1 Presentation of Matlab Environment

"Matlab® is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include math and computation, algorithm development, data acquisition modelling, simulation, and prototyping, data analysis, exploration, and visualization, scientific and engineering graphics, and application development, including graphical user interface building [20]."

1.2.2 MATLAB's Features

"Matlab features a family of add-on application-specific solutions called toolboxes. The Matlab system consists of five main parts:

- Development Environment. This is the set of tools and facilities that help you use Matlab functions and files. Many of these tools are graphical user interfaces. It includes the Matlab desktop and Command Window, a command history, an editor and debugger, and browsers for viewing help, the workspace, files, and the search path.
- The Matlab Mathematical Function Library. This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.
- The Matlab Language. This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create large and complex application programs.
- Matlab applications program Interface (API). This is a library that allows you to write C and FORTRAN programs that interact with Matlab. It includes facilities for calling routines from Matlab (dynamic linking), calling Matlab as a computational engine, and for reading and writing MAT-files". [20]

1.3 THESIS OUTLINES

After the introduction, which contains a brief summary of some useful terms that will be used in this thesis, chapter 2 puts emphasis onto all methods and procedures for calculating the parameters of the simulation models. The first part concerns the frequencies' non-selective channels. However, in the second part the frequencies' selective channels are discussed.

Chapter 3 deals with mobile fading channel simulators. The first part concerns the spatial channel simulator. There, we briefly explain the fundamental concept as well as some characteristics. After that, we describe how we implement the generation of the process and functions such as ACF, LCR, etc. for testing the fitting of the simulation model against the reference model. In the second part, we introduce MIMO channels. All MIMO models are devoted. At first, we implement MIMO one-ring model simulator. Then we extend to MIMO tow-ring model simulator. The last part consists of the development of MIMO elliptical simulator. It is made clear that for each simulator, we implement some functions for testing, such as time, ACF, CCF, etc…

The last chapter is entirely dedicated to the discussion and interpretation of the present results. It is important to show how far we achieve our goals during this project. Then, a conclusion and some perspectives open a window for further work which will be done and for any future work.

CHAPTER 2 - PARAMETERS COMPUTATION METHODS

In this chapter, we present all possible parameter computation methods founded in literature and issued from many articles and publications. Each channel is characterized by a set of parameters that should be determined.
We face two channels types: non-selective-frequency and selective-frequency. For the first kind, we are intended to determine the discrete Doppler frequencies as well as Doppler coefficients. In the second part, our focus is to compute the path gains and the discrete propagation delays. Hence, this chapter is divided into two sections; one called frequency-nonselective channels and one called frequency-selective channels. Therefore, our intention is to compute the parameters of each channel type using many methods such as MEA, MED, MSEM, MCM, LPNM, MEDS, R-MEDS, MEDS-sp, etc...

2.1 FREQUENCY-NONSELECTIVE CHANNELS

Generally, in the frequency-non-selective channels, we just put emphasis onto the factors that are related to Doppler effects, such as Doppler frequencies and Doppler coefficients. As a result, in the following, we consider different methods to compute these parameters. First, we explain briefly the principles of these methods. Afterwards, we depict our implementation approach and finally we show some explicit results.

2.1.1 Methods for the Computation of the Discrete Doppler Frequencies and Doppler Coefficients

In this section, we note briefly the principle of each method and the way it was implemented. Further details can be founded in [1] chapter 5.

2.1.1.1 Method of Equal Distances (MED)

The method of equal distances has as principle that the Doppler frequencies set is divided into equal distances in such a way the neighboured pairs have the some distances. Hence, the Doppler frequencies are given by:

$$f_{i,n} = \frac{\Delta f_i}{2}(2n-1), \qquad n = 1, 2, \ldots, N_i, \tag{2.1}$$

where

$$\Delta f_i = f_{i,n} - f_{i,n-1}, \qquad n = 2, 3, \ldots, N_i \tag{2.2}$$

And the Doppler coefficients are determined by the integral of the power spectral density over the interval $I_{i,n} = \left[f_{i,n} - \frac{\Delta f_i}{2}, f_{i,n} + \frac{\Delta f_i}{2}, \right]$ as shown here:

$$c_{i,n} = \sqrt{\int_{f \in I_{i,n}} S_{\mu_i \mu_i}(f)\, df} \tag{2.3}$$

We take as an example of application of the above principle.

Jakes power spectral density:

Using Jakes's approach, the Doppler frequencies and the Doppler coefficients are given by:

$$f_{i,n} = \frac{f_{\max}}{2N_i}(2n-1) \tag{2.4}$$

$$c_{i,n} = \frac{2\sigma_0}{\sqrt{\pi}}\left[\arcsin\left(\frac{n}{N_i}\right) - \arcsin\left(\frac{n-1}{N_i}\right)\right]^{1/2} \tag{2.5}$$

where

$$\sigma_0^2 = \tilde{r}_{\mu_i \mu_i}(0) = \sum_{n=1}^{N_i} \frac{c_{i,n}^2}{2} \tag{2.6}$$

Here σ_0^2 denotes the variance of the process $\mu_i(t)$.

Gauss power spectral density:

The Doppler frequencies and the Doppler coefficients are determined through equations below:

$$f_{i,n} = \frac{\kappa_c f_c}{2N_i}(2n-1) \tag{2.7}$$

$$c_{i,n} = \sigma_0\sqrt{2}\left[erf\left(\frac{n\kappa_c}{N_i}\right) - erf\left(\frac{(n-1)\kappa_c\sqrt{\ln 2}}{N_i}\right)\right]^{1/2} \tag{2.8}$$

where

$$\kappa_c = 2\sqrt{2/\ln 2} \tag{2.9}$$

More explanation about this method is presented in [1], chapter 5, section 5.1.1.

2.1.1.2 Mean-Square-Error Method (MSEM)

In this method, the computation of the channel parameters is based on the minimization of the mean-square error:

$$E_{r_{\mu_i \mu_i}} = \frac{1}{\tau_{\max}} \int_0^{\tau_{\max}} \left(r_{\mu_i \mu_i}(\tau) - \tilde{r}_{\mu_i \mu_i}(\tau)\right)^2 d\tau \tag{2.10}$$

Thus, when minimizing the error function we should take into consideration that we can't optimize both sets of parameters $f_{i,n}$ and $c_{i,n}$ simultaneously. Consequently, we fix $f_{i,n}$ which are given by (2.1) and try to optimize the Doppler coefficients. Then, we obtain:

$$c_{i,n} = \sqrt{\frac{1}{\tau_{\max}} \int_0^{\tau_{\max}} r_{\mu_i\mu_i}(\tau) \cos\left(2\pi f_{i,n}\tau\right) d\tau}, \quad n = 1, 2, \ldots, N_i \left(i = 1, 2\right) \tag{2.11}$$

where

$$\tau_{\max} = {T_i}/{4} = {1}/{2\Delta f_i} \tag{2.12}$$

This method is applied to the Jakes and Gaussian power spectral densities.

Jakes power spectral density:

$f_{i,n}$ can be obtained from (2.4) and the Doppler coefficients are determined by the following formula:

$$c_{i,n} = 2\sigma_0 \sqrt{\frac{1}{\tau_{\max}} \int_0^{\tau_{\max}} J_0\left(2\pi f_{\max}\tau\right) \cos\left(2\pi f_{i,n}\tau\right) d\tau} \tag{2.13}$$

where

$$\tau_{\max} = {N_i}/{\left(2 f_{\max}\right)} \tag{2.14}$$

Gauss power spectral density:

$f_{i,n}$ are given by (2.7). According to (2.10), we get the expression for the Doppler coefficients:

$$c_{i,n} = 2\sigma_0 \sqrt{\frac{1}{\tau_{\max}} \int_0^{\tau_{\max}} e^{{-(\pi f_c \tau)^2}/{\ln 2}} \cos\left(2\pi f_{i,n}\tau\right) d\tau}, \quad n = 1, 2, \ldots, N_i \left(i = 1, 2\right) \tag{2.15}$$

where κ_c is defined by (2.9). More explanation about this method is presented in [1], chapter 5, section 5.1.2.

2.1.1.3 Method of Equal Areas (MEA)

The method of equal areas supposes that the obtained Doppler frequencies should fulfil the condition where the area under the Doppler power spectral density $S_{\mu_i\mu_i}(f)$ must be equal to $\sigma_0^2/2N_i$ within $f \in \left[f_{i,n-1}, f_{i,n}\right]$. Then, we found the Doppler frequencies depending on the number of scatterers as well as the variance of the process.

$$f_{i,n} = G_{\mu_i}^{-1}\left[\frac{\sigma_0^2}{2}\left(1 + \frac{n}{N_i}\right)\right], \qquad n = 1, 2, \ldots, N_i \left(i = 1, 2\right) \tag{2.16}$$

where

$$G_{\mu_i}\left(f_{i,n}\right)=\int_{-\infty}^{f_{i,n}} S_{\mu_i\mu_i}\left(f\right)df=\frac{\sigma_0^2}{2}\left(1+\frac{n}{N_i}\right) \tag{2.17}$$

However, the method of equal areas presumes the Doppler coefficients are constant independently of the index of the scatterer. Here, the Doppler coefficients are shown.

$$c_{i,n}=\sigma_0\sqrt{\frac{2}{N_i}}\ ,\ n=1,2,\ldots,N_i\left(i=1,2\right) \tag{2.18}$$

As an example of applications, we consider as the previous methods the Jakes and Gaussian power spectral densities.

Jakes power spectral density:

Applying Jakes power spectral density; we get the Doppler frequencies in the equation below.

$$f_{i,n}=f_{\max}\sin\left(\frac{\pi n}{2N_i}\right),\ \ n=1,2,\ldots,N_i\left(i=1,2\right) \tag{2.19}$$

And the Doppler coefficients $c_{i,n}$ are given by (2.18).

Gauss power spectral density:

When using method of equal areas, the Doppler frequencies can be founded by solving the following expression by means of a proper numerical root-finding technique.

$$\frac{n}{N_i}-erf\left(\frac{f_{i,n}}{f_c}\sqrt{\ln 2}\right)=0,\qquad \forall n=1,2,\ldots,N_i\ \left(i=1,2\right) \tag{2.20}$$

As with the Jakes examples, the Doppler coefficients $c_{i,n}$ are given by (2.18). More explanation about this method is presented in [1], chapter 5, section 5.1.3.

2.1.1.4 Monte Carlo Method (MCM)

The main idea behind the Monte Carlo method is to realize the discrete Doppler frequencies according to the probability density function $p_{\mu_i}\left(f\right)=\frac{1}{\sigma_0^2}S_{\mu_i\mu_i}\left(f\right)$. As result, the Doppler frequencies are obtained from:

$$f_{i,n}=g_{\mu_i}\left(u_n\right)=F_{\mu_i}^{-1}\left(u_n\right) \tag{2.21}$$

where

$$F_{\mu_i}\left(f_{i,n}\right)=\int_{-\infty}^{f_{i,n}} p_{\mu_i}\left(f\right)df=\int_{-\infty}^{f_{i,n}}\frac{1}{\sigma_0^2}S_{\mu_i\mu_i}\left(f\right)df \tag{2.22}$$

u_n is a random variable uniformly distributed over $\left(0,1\right]$. $c_{i,n}$ are given by the equation. Again, we apply this method to Jakes and Gaussian power spectral densities.

Jakes power spectral density:

The employment of this method for Jakes power spectral density leads to the expression depicted here:

$$f_{i,n} = f_{\max} \sin\left(\frac{\pi}{2} u_n\right) \tag{2.23}$$

$c_{i,n}$ are given by (2.18).

Gauss power spectral density:

With the method of Monte Carlo in connection to Jakes power spectral density, a closed-form expression of Doppler frequencies could not be achieved. However, they can be determined by the zeros of the following equation

$$u_n - erf\left(\frac{f_{i,n}}{f_c}\sqrt{\ln 2}\right) = 0, \qquad \forall n = 1,2,\ldots,N_i \ (i = 1,2) \tag{2.24}$$

And $c_{i,n}$ are given by (2.18). More explanation about this method is presented in [1], chapter 5, section 5.1.4.

2.1.1.5 L_p-norm Method (LPNM)

Similarly to the mean-square error method, the L_p-norm method (LPNM) considers minimizing the error function presented below in order to get optimized Doppler frequencies and Doppler coefficients. Experimentally speaking, when we set $c_{i,n}$ to $\sigma_0\sqrt{2/N_i}$ and look for an optimized solution of

$$E_{p_{\mu_i}}^{(p)} = \left\{\frac{1}{\tau_{\max}}\int_0^{\tau_{\max}} \left|r_{\mu_i\mu_i}(\tau) - \tilde{r}_{\mu_i\mu_i}(\tau)\right|^p dx\right\}^{1/p}, \quad p = 1,2,\ldots \tag{2.25}$$

we can find optima $f_{i,n}$.

As starting point, we take $f_{i,n}$ obtained from the MEA method in (2.19) for Jakes power spectral density and in (2.20) for Gaussian power spectral density. More explanation about this method is presented in [1], chapter 5, section 5.1.5.

2.1.1.6 Method of Exact Doppler Spread (MEDS)

As implied in the name, the method of exact Doppler spread obeys power spectral density. In fact, this method, despite its simplicity, depicts a high performance and leads to nearly perfect fitting of the autocorrelation of both reference and simulation models. Here, we present two different approaches for the definition of this method.

Jakes power spectral density:

As previously, $c_{i,n}$ are given again by (2.18), and the Doppler frequencies are identified after some mathematical demonstration steps by

$$f_{i,n} = f_{\max} \sin\left[\frac{\pi}{2N_i}\left(n-\frac{1}{2}\right)\right] \tag{2.26}$$

Gauss power spectral density:

The Doppler coefficients $c_{i,n}$ are given by (2.18). In addition, the Doppler frequencies are founded when solving the following equation

$$\frac{2n-1}{2N_i} - erf\left(\frac{f_{i,n}}{f_c}\sqrt{\ln 2}\right) = 0, \quad \forall n = 1,2,\ldots,N_i - 1 \tag{2.27}$$

and

$$f_{i,N_i} = \sqrt{\frac{\beta N_i}{\left(2\pi\sigma_0\right)^2} - \sum_{n=1}^{N_i-1} f_{i,n}^2} \tag{2.28}$$

2.1.1.7 Jakes Method (JM)

The present method is dedicated, as the name refers, to Jakes power spectral density. However, as the above method, the Jakes method will not be demonstrated in detail unless the highlight expressions for each Doppler frequencies as well as Doppler coefficients which are given in the following equation. More explanation about this method is presented in [1], chapter 5, section 5.1.6.

$$c_{i,n} = \begin{cases} \dfrac{2\sigma_0}{\sqrt{N_i - \frac{1}{2}}}\sin\left(\dfrac{\pi n}{N_i - 1}\right), & n = 1,2,\ldots,N_i - 1, \quad i = 1 \\ \dfrac{2\sigma_0}{\sqrt{N_i - \frac{1}{2}}}\cos\left(\dfrac{\pi n}{N_i - 1}\right), & n = 1,2,\ldots,N_i - 1, \quad i = 2 \\ \dfrac{\sigma_0}{\sqrt{N_i - \frac{1}{2}}}, & n = N_i, i = 1,2 \end{cases} \tag{2.29}$$

The Doppler frequencies are shown here

$$f_{i,n} = \begin{cases} f_{\max}\cos\left(\dfrac{n\pi}{2N_i - 1}\right), & n = 1,2,\ldots,N_i - 1, \quad i = 1,2 \\ f_{\max}, & n = N_i, i = 1,2 \end{cases} \tag{2.30}$$

Finally, we consider the Doppler phases set to zeros for all scatterers.

$$\theta_n = 0,\ n = 1,2,\ldots,N_i,\ i = 1,2 \tag{2.31}$$

2.1.1.8 Randomized MEDS (R-MEDS)

The Randomized method of exact Doppler spread R-MEDS is a kind of improvement for the original MEDS in order to get higher performance for a small number of sinusoids. In [4], it is pointed out that when adding randomized phase shift uniformly distributed on $[-\pi/4N_i, \pi/4N_i]$, a significant improvement shows up. The Doppler frequencies are given by

$$\alpha_{i,n}^{(k)} = \frac{\pi}{2N_i}\left(n - \frac{1}{2}\right) + u_{i,n}^{(k)}, \tag{2.32}$$

where

$$u_{i,n}^{(k)} \sim U\left(-\frac{\pi}{4N_i}, \frac{\pi}{4N_i}\right] \tag{2.33}$$

2.1.1.9 MEDS with Set Partitioning (MEDS-sp)

The principle of MEDS with set partitioning is derived from the design of trellis-coded schemes. Since the placement of the scatterers all around the ring is not uniformly distributed, our intention then is to divide the aimed constellation of locations of scatterers into sub-constellation as shown in the figure below.

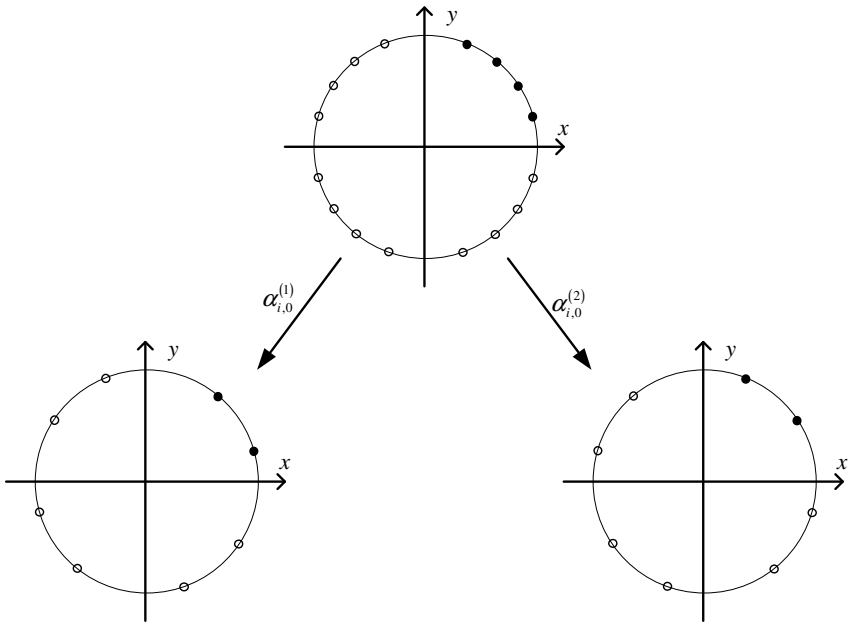

Figure 2-1: Set-partitioning principle

The figure 2-1 depicts an example of set-partitioning of a 4-scatterers constellation into two of 2-scatterers sub-constellation (empty circle stand for redundant scatterers and full circle for relevant scatterers). When we apply the principle which is detailed further off in [5], we find the Doppler angles given by

$$\alpha_{i,n}^{(k)} = \frac{\pi}{2N_i}\left(n - \frac{1}{2}\right) + \alpha_{i,0}^{(k)} \tag{2.34}$$

where

$$\alpha_{i,0}^{(k)} = \frac{\pi}{2KN_i}\left(k - \frac{K+1}{2}\right) \tag{2.35}$$

Remark: For the drawing of the ACF, we should take into consideration the number of constellation K so that we figure out the best match between the reference model and simulation model. Thus, the expression of the sample ACF is given by

$$\overline{r}_{\mu_i\mu_i}(\tau) = \frac{1}{K}\sum_{k=1}^{K}\tilde{r}_{\mu_i\mu_i}^{(k)}(\tau) \tag{2.36}$$

2.1.2 Implementation and Results

In this section, we present the implementation steps and how the user can use this toolbox. In addition, a detailed description of the toolbox is shown here.

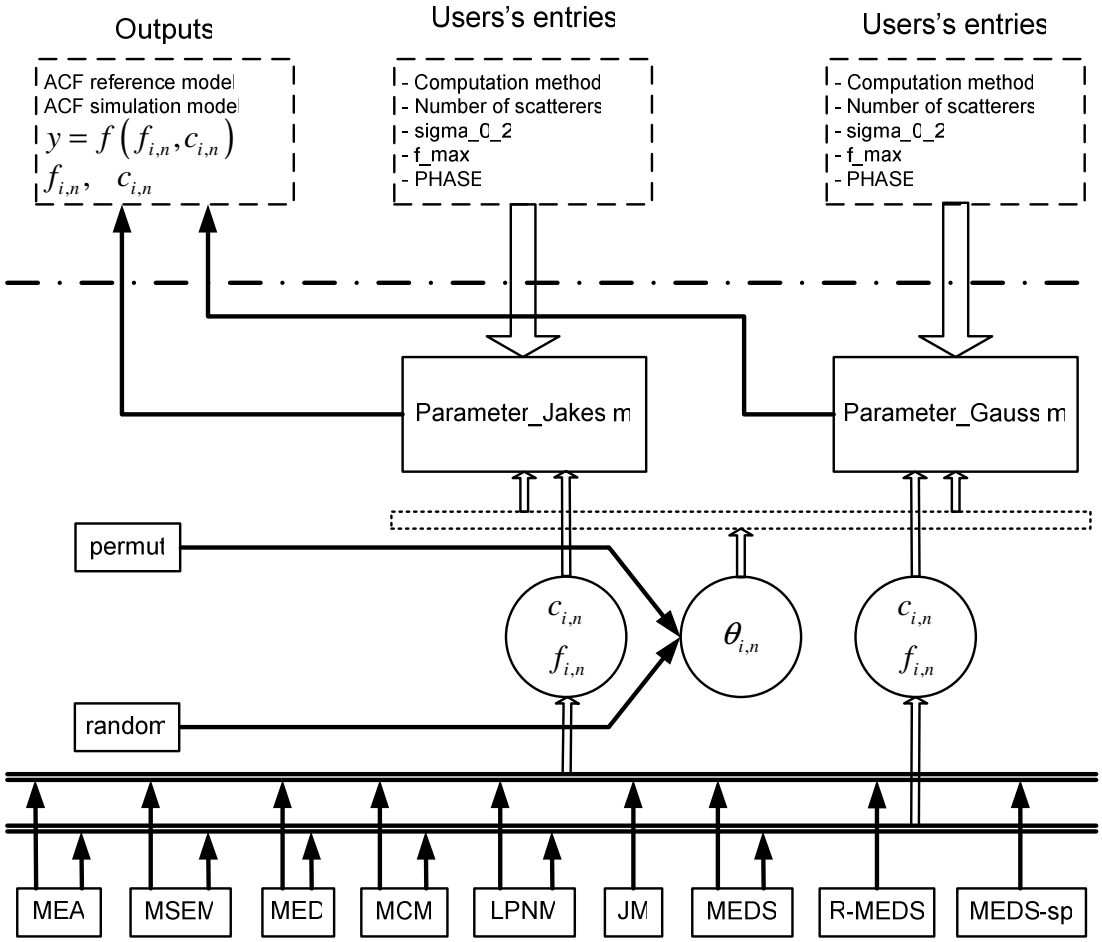

Figure 2-2: The implementation diagram of the parameters computation methods for the frequency-non-selective channel

The figure 2-2 presents the way that we proceeded to implement our toolbox for the computation of channel parameters. The user has to first pick the appropriate computation method for the Doppler frequencies, and Doppler coefficients that can be, for instance, the

method of equal areas MEA. Secondly, he sets the number of scatterers, the variance σ_0^2, and the frequency Doppler shift. In the last phase, he has to choose the method for the computation of the phase which can be randomly or permitted.
The algorithm depicts the following characteristics shown in bullets.

- Conditions on the computation methods: MEA, MSEM, MED, MCM, LPNM, JM, MEDS, R-MEDS, and MEDS-sp
- The LPNM method needs four functions (LPNM_opt_Jakes.m, LPNM_opt_Gauss.m, fun_Gauss.m, and fun_Jakes.m) so that it computes the error function relative to each model (either Jakes or Gauss).
- To evaluate the performance of each method, we altogether implement some testing programs for the visualization of the ACF of reference and simulation model.

2.1.2.1 Results

In order to evaluate the performance of each parameter computation method, we show some results coming from each method separately. Afterwards, we evaluate each method by comparison with others. The plots were taken for $N = 25$, $\sigma_0^2 = 1$, $f_{\max} = 91Hz$ and for different parameters computation methods.

The first set of plots considers Jakes power spectral density. Here, each figure contains a shape for the power spectral density in function of repartition of channel frequencies and a shape for the autocorrelation function in function of the normalized time separation.

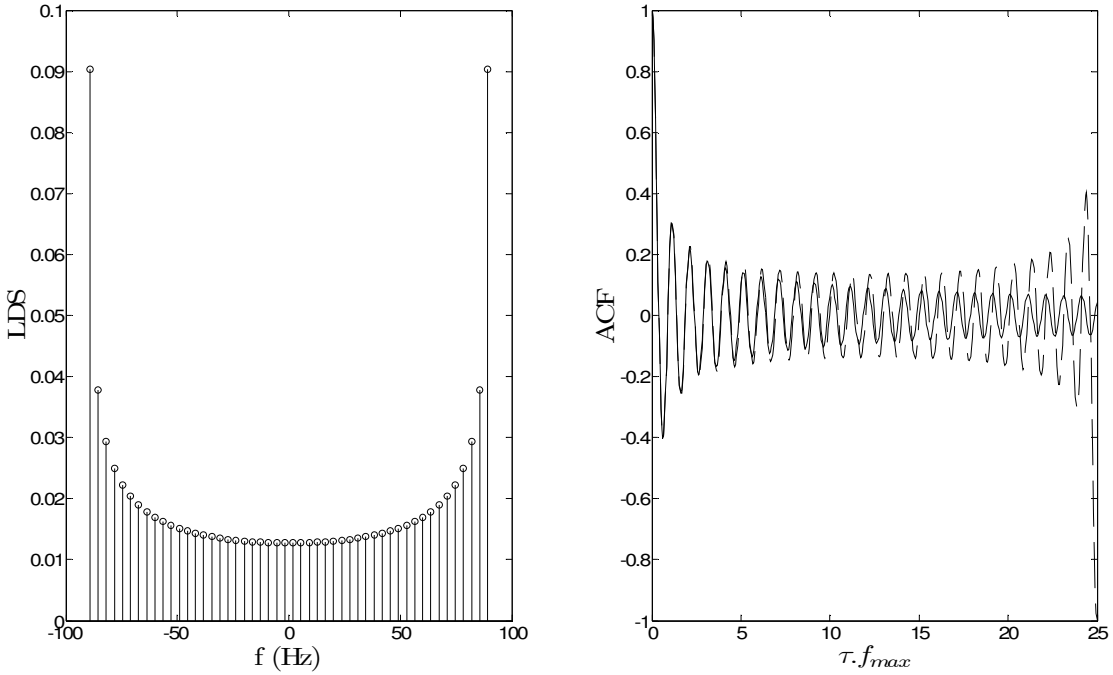

Figure 2-3: PSD and ACF of Jakes model using MED method

The method of equal distances doesn't fulfil the constraint of a good match between the simulation model and reference model until $N/2$.

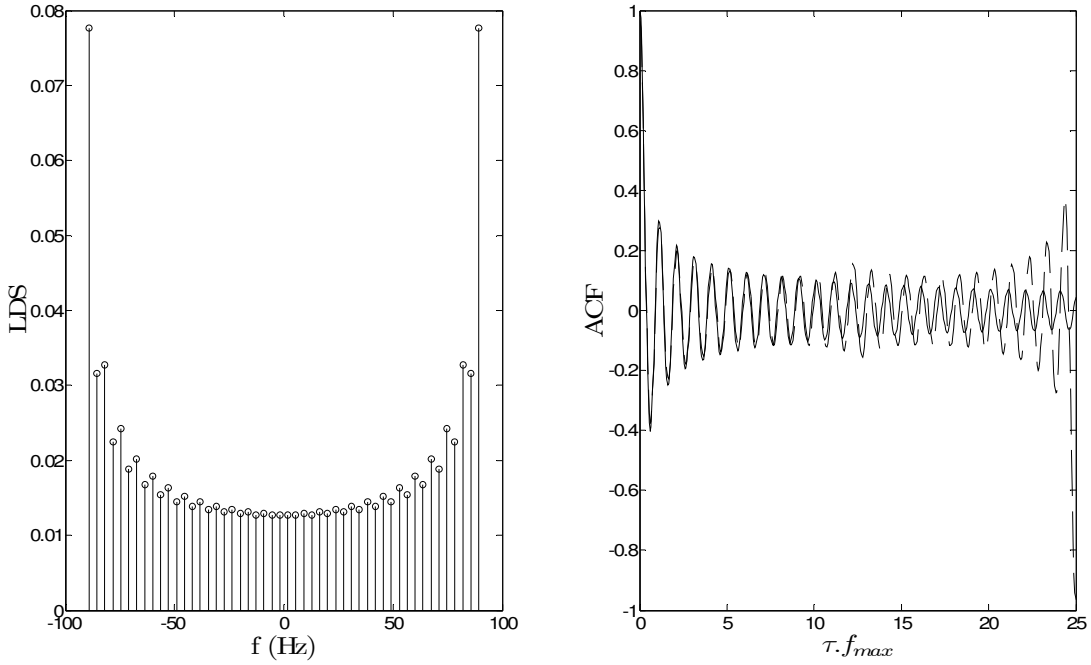

Figure 2-4: PSD and ACF of Jakes model using MSEM method

Unless a fine improvement shapes the autocorrelation function, the MSEM method doesn't reach the full requirement.

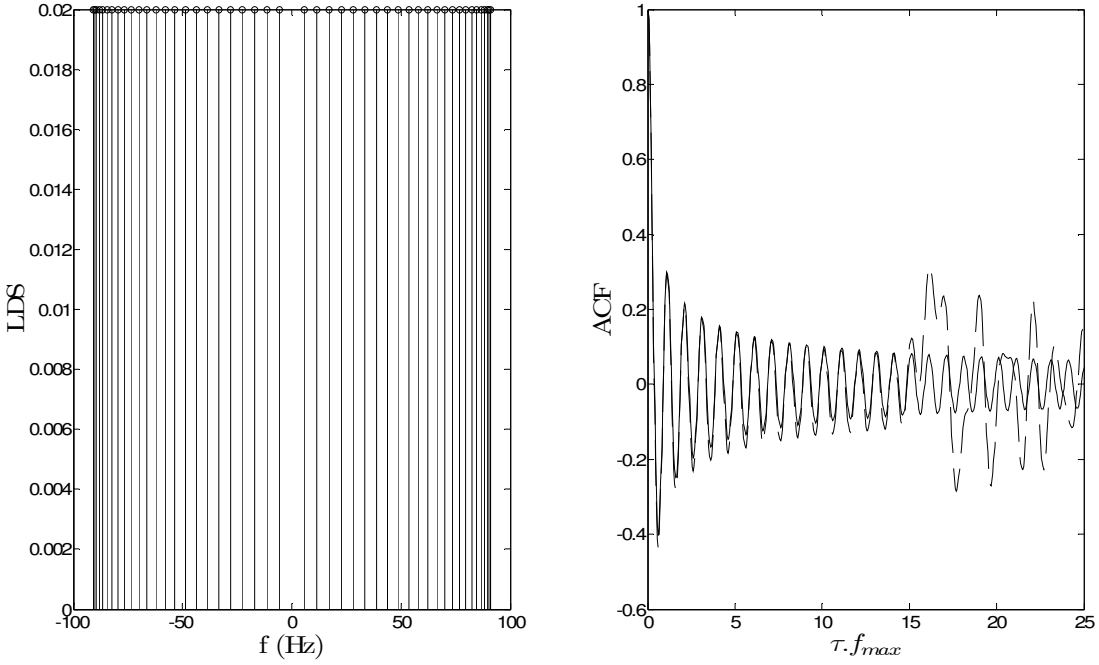

Figure 2-5: PSD and ACF of Jakes model using MEA method

Despite its simplicity, the method of equal areas presents some impurity between the reference and simulation models.

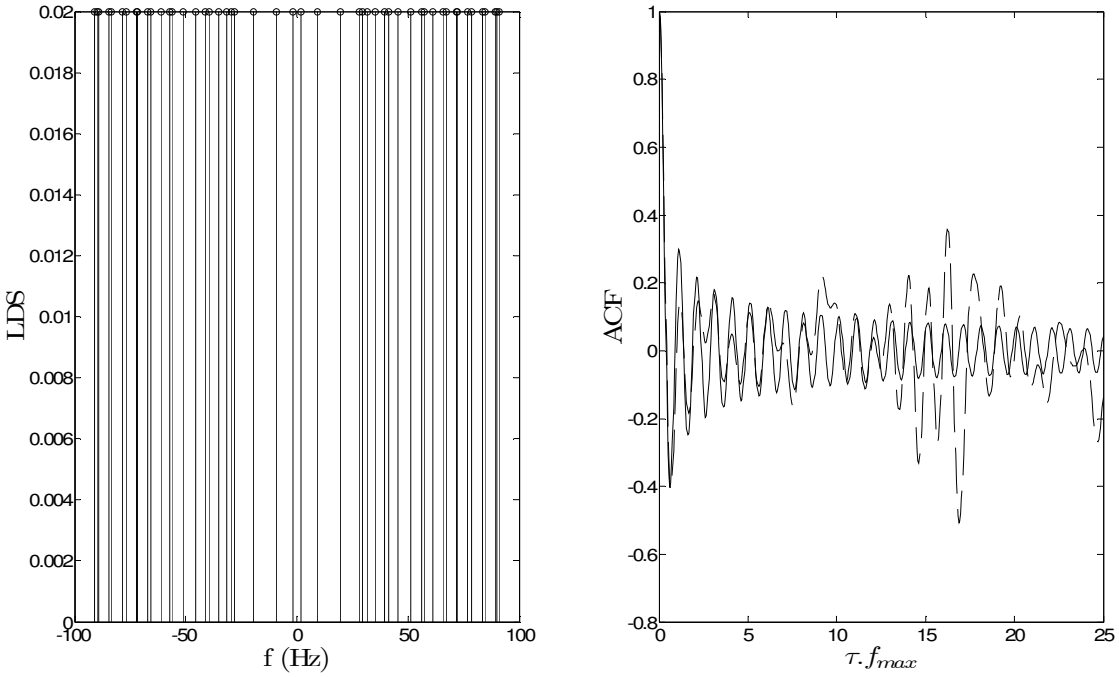

Figure 2-6: Power density function and Autocorrelation function using MCM method for Jakes model

The figure 2-6 is given by just one realization of the MCM method. However, this method requires several realizations so that we approach the desired fitting.

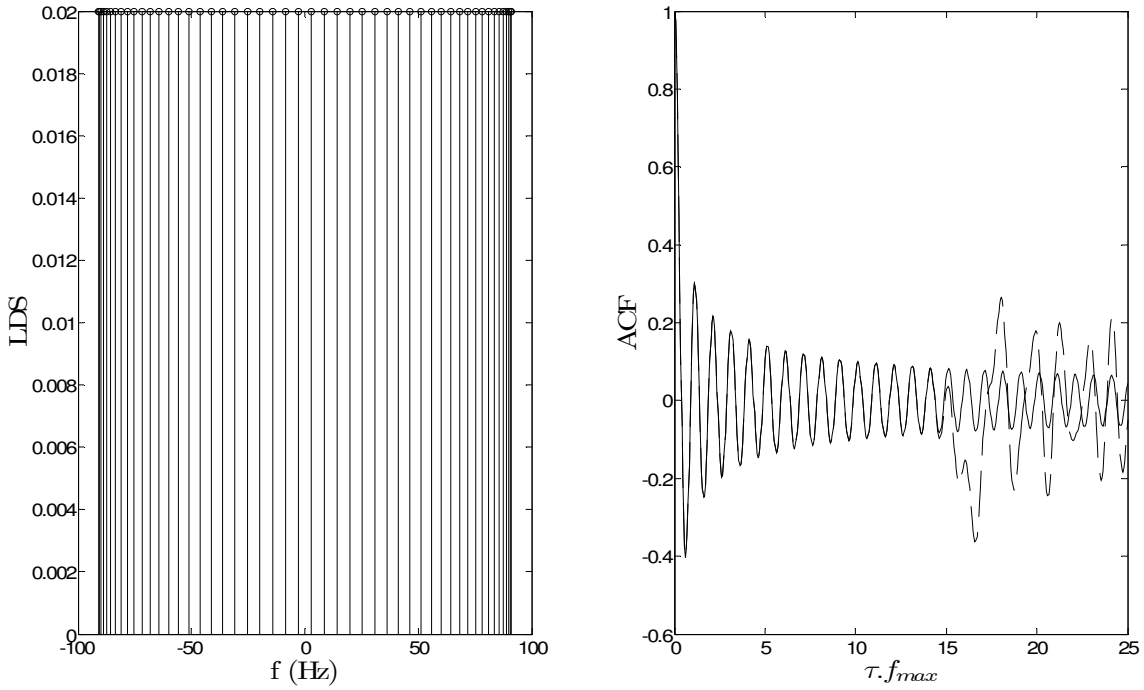

Figure 2-7: PSD and ACF of Jakes model using LPNM method

Apparently, a perfect fitting is reached by using the LPNM method since we get an overlapping of both shapes until more than $N/2$.

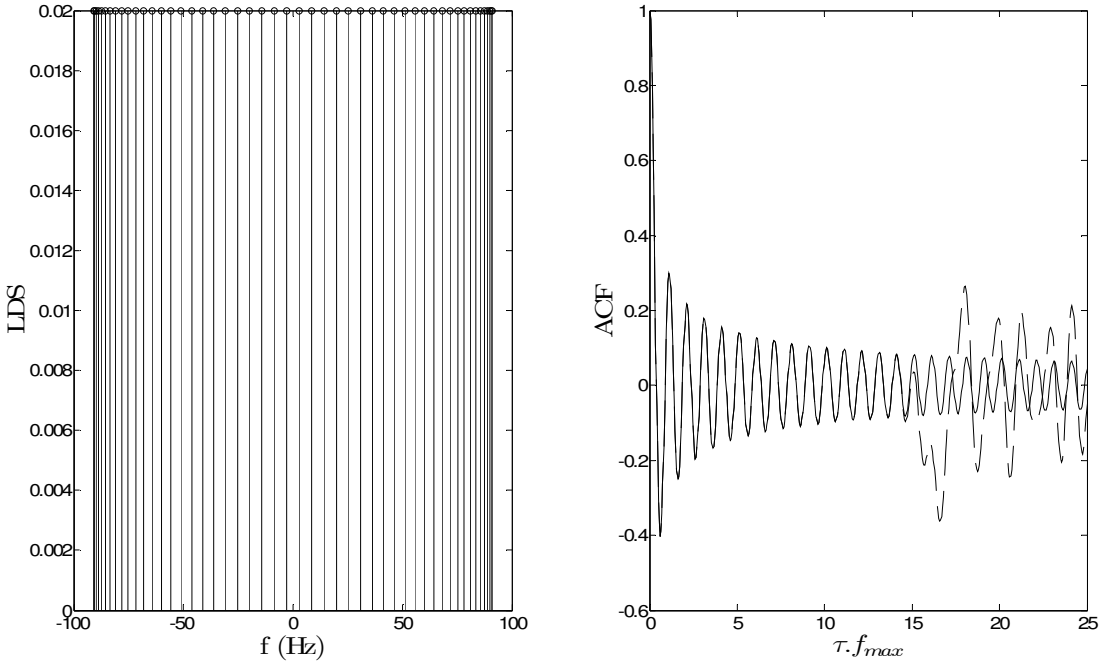

Figure 2-8: PSD and ACF of Jakes model using MEDS method

As with the LPNM method, MEDS provides a perfect performance on the simulation model.

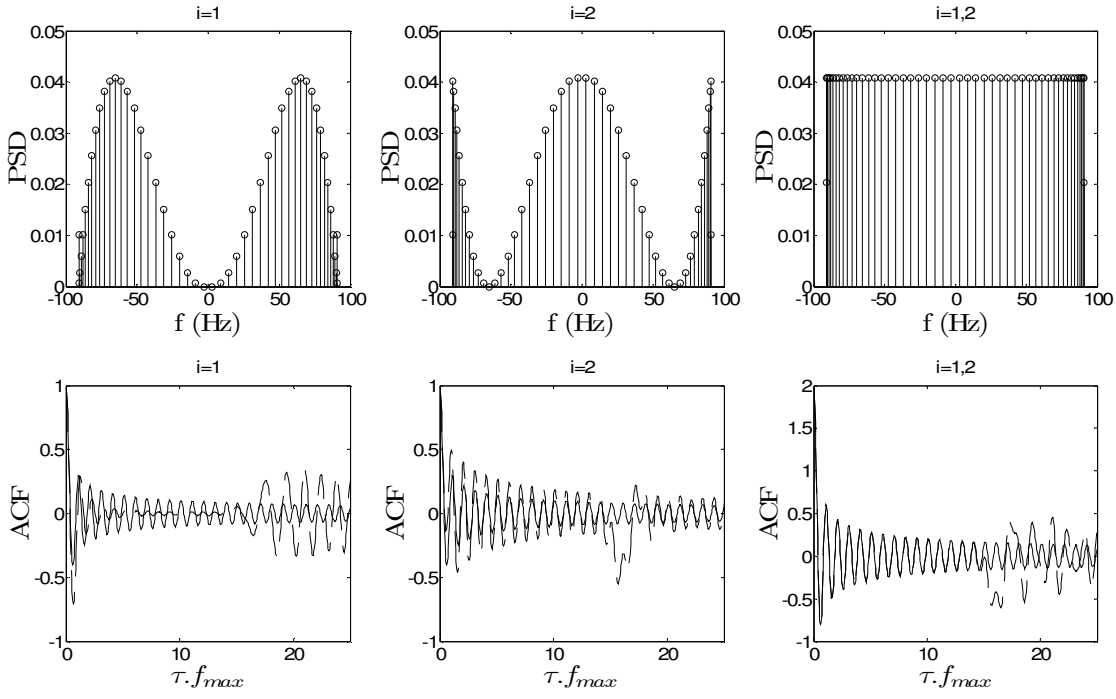

Figure 2-9: PSD and ACF of Jakes model using JM method

The JM method is supposed to have much iteration in order to compensate the error caused by one realization. When we sum-up two realizations as depicted in figure 2-9, we can reach a good performance.

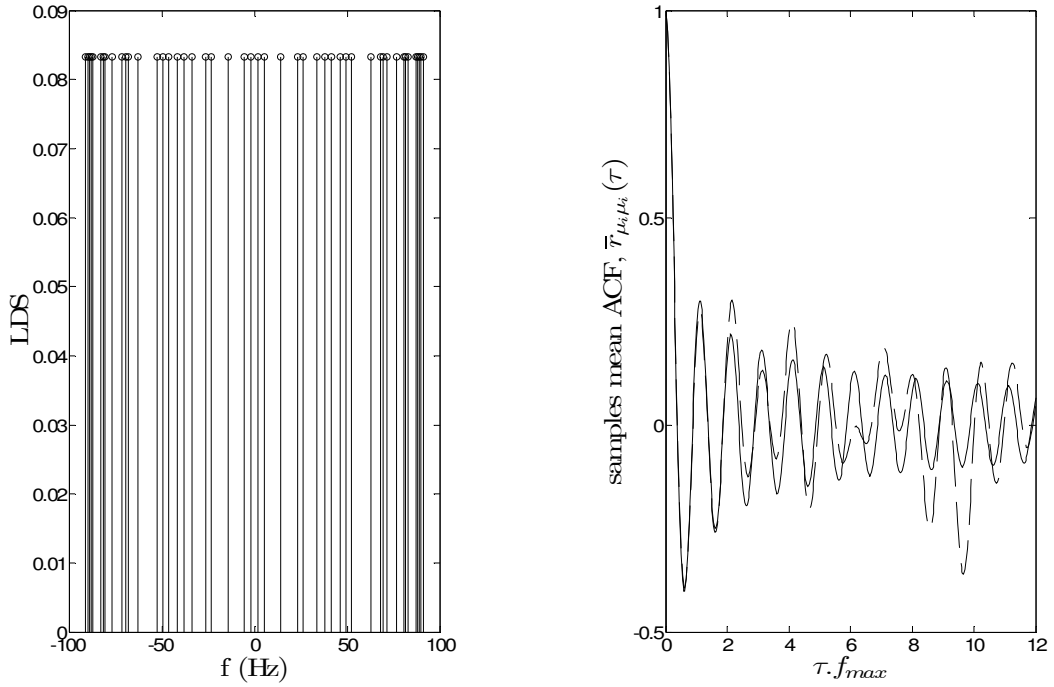

Figure 2-10: PSD and ACF of Jakes model using R-MEDS method

In our case, the R-MEDS doesn't response to our aim of getting a good fitting between the simulation model and reference model. As shown in figure 2-10, the samples mean ACF of the reference model has the some behaviour as the simulation model until merely $N/8$.

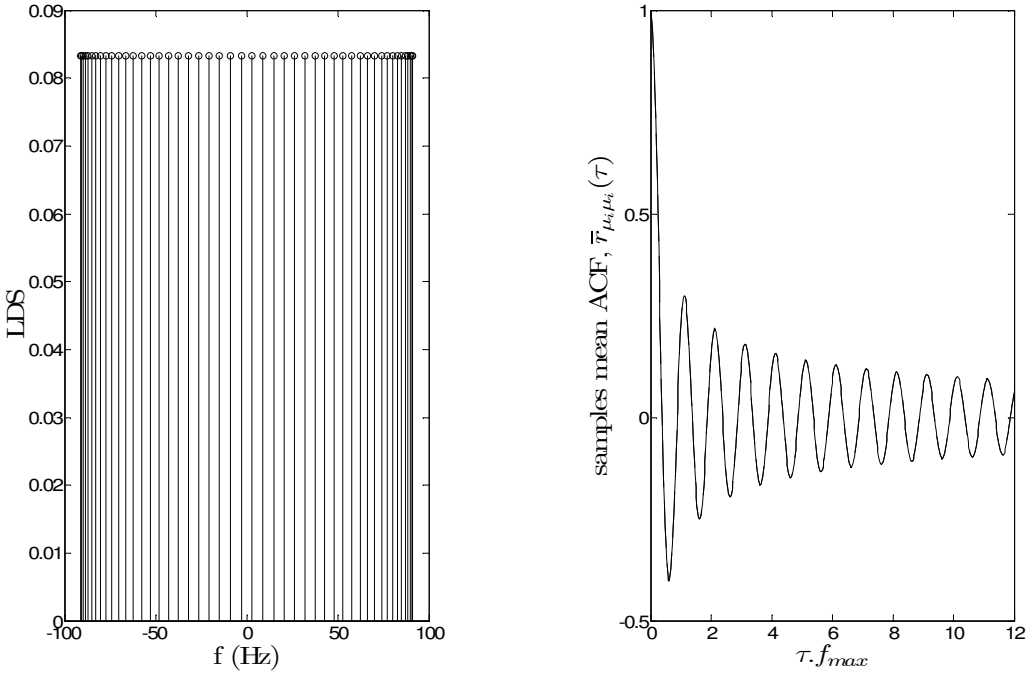

Figure 2-11: PSD and ACF of Jakes model using MEDS-sp

The MEDS-sp is considered to be the best method altogether until $N/2$. The main advantage of this method is the time consuming and the efficiency since we reduce the number of iteration for the computation parameters. For instance, in figure 2-11 the number of sub-constellation chosen is $K = 4$ and $N_i = 6$. Comparing with the results found for the MEDS method with $N_i = 25$, we merely reach the same performance. Therefore, the MEDS-sp considerably reduces the complexity of the system when it comes to realization.
In the second set of figures, we consider the Gaussian power spectral density. All plots were taken for $N = 25$, $\sigma_0^2 = 1$, $f_{\max} = 91Hz$, $f_c = 70Hz$ and for different methods.

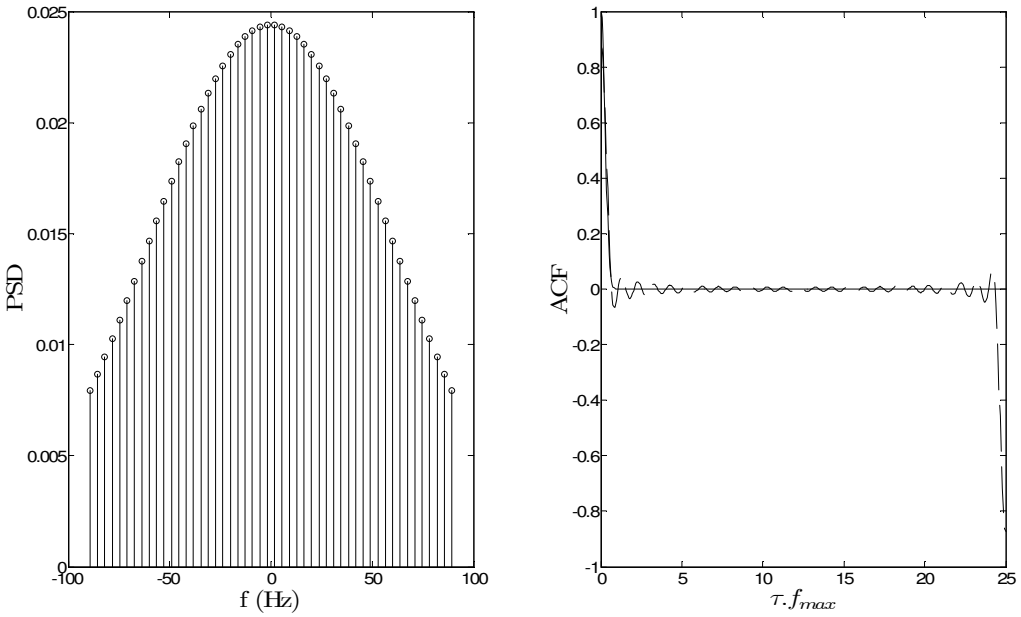

Figure 2-12: PSD and ACF of Gauss model using MED method

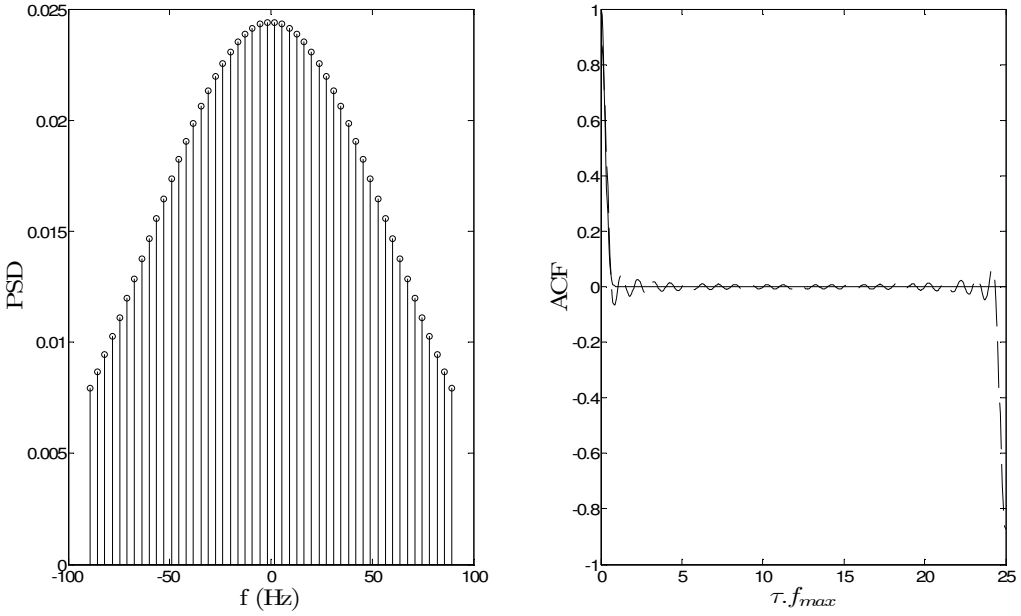

Figure 2-13: PSD and ACF of Gauss model using MSEM method

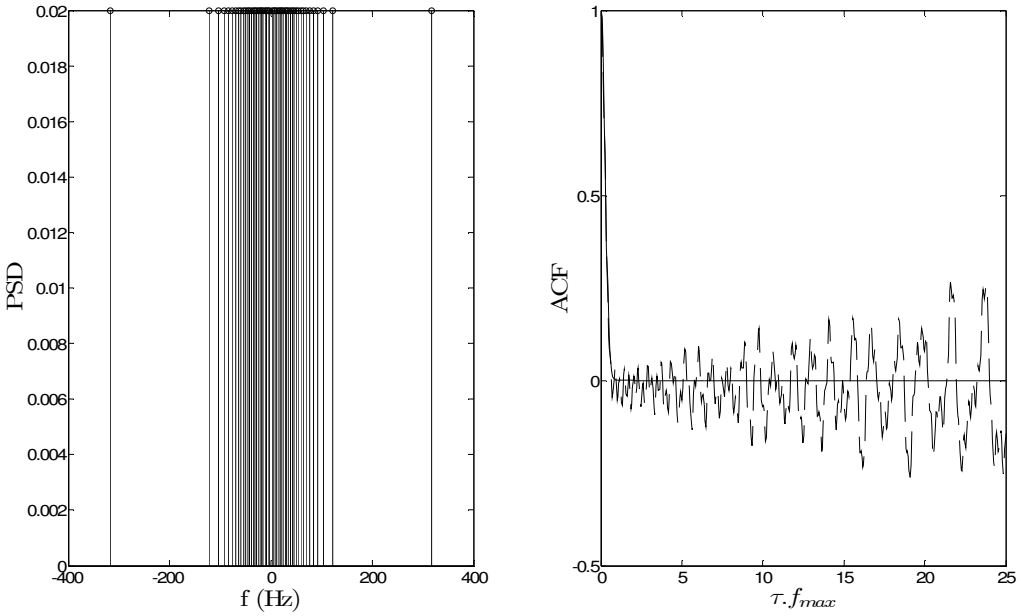

Figure 2-14: PSD and ACF of Gauss model using MEA method

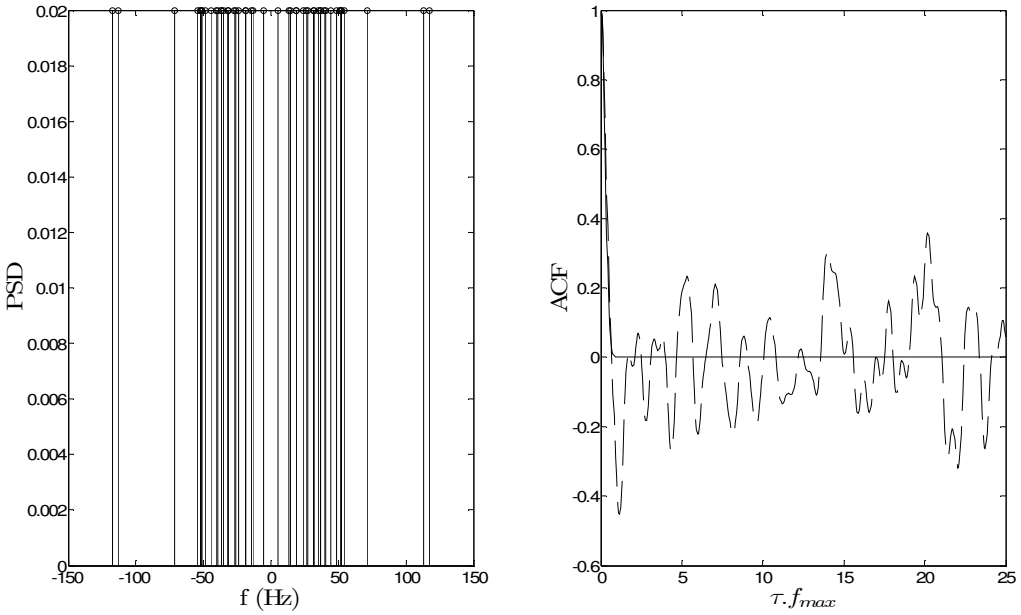

Figure 2-15: PSD and ACF of Gauss model using MCM method

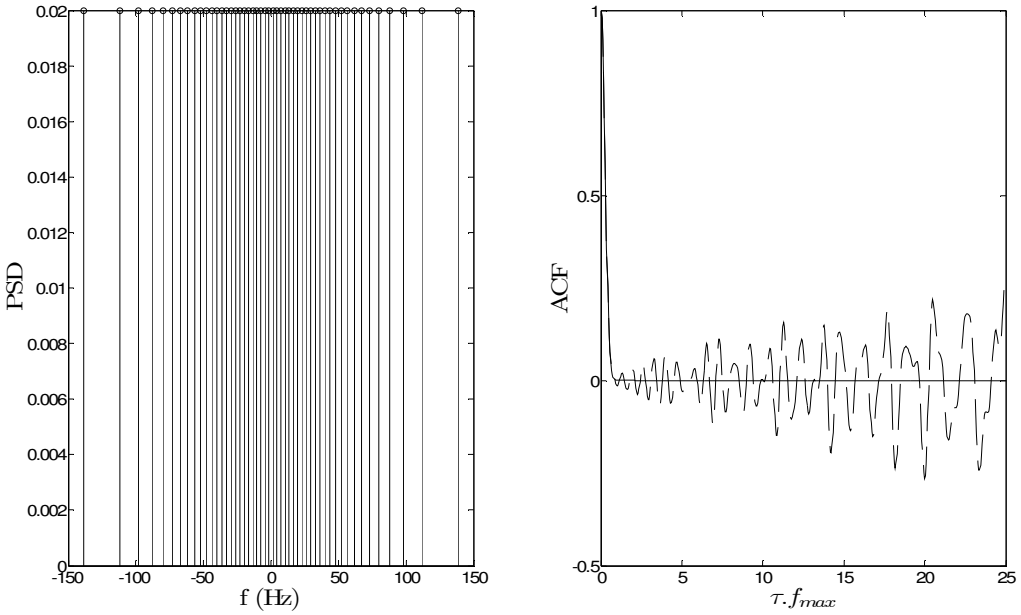

Figure 2-16: PSD and ACF of Gauss model using MEDS method

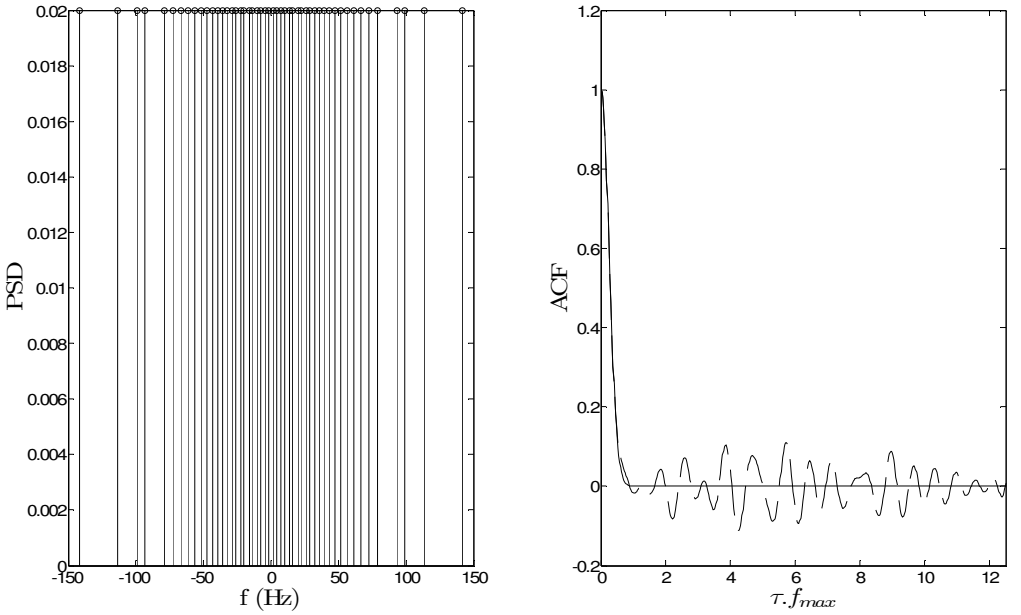

Figure 2-17: PSD and ACF of Gauss model using LPNM method

The correspondent programs for the implementation of the listed methods can be found in appendix 1.

2.1.2.2 Comparison

In this section, we compare the parameters computation methods (MED, MSEM, MEA, MCM, LPNM, MEDS, JM, R-MEDS, MEDS-sp), presented in the previous section, by judging the fitting between the simulation model and the reference mode. According to figures 2-3 – 2-11, it appears for Jakes power spectral density that the LPNM and MEDS-sp method carry out the best performance. Nevertheless, the MCM method for one realization and R-MEDS method consists of the worst ones. The rest of the methods didn't respond to

our goal, fitting the reference model in regards to the simulation model, but they can be used in some cases. For Gaussian power spectral density, as shown in figures 2-12 – 2-17 the MED method realize the best results. To conclude, the new set of parameters computation methods comes up to get rid of the drawbacks which occurred in the previous methods.

2.2 FREQUENCY-SELECTIVE CHANNELS

In this section, our focus is to determine the channel gains and delays of frequency-selective channels. The first part is devoted to describe some basic concepts concerning the frequency-selective channel. In the second part, we exhibit some useful parameters computation methods used to compute the channel gains and delays. The last part deals with the implementation and comparison between different methods.

2.2.1 Basic Concepts

Our approach is based on the so-called Tapped-delay line model for a frequency-selective mobile fading channel which is drawn in the following figure.

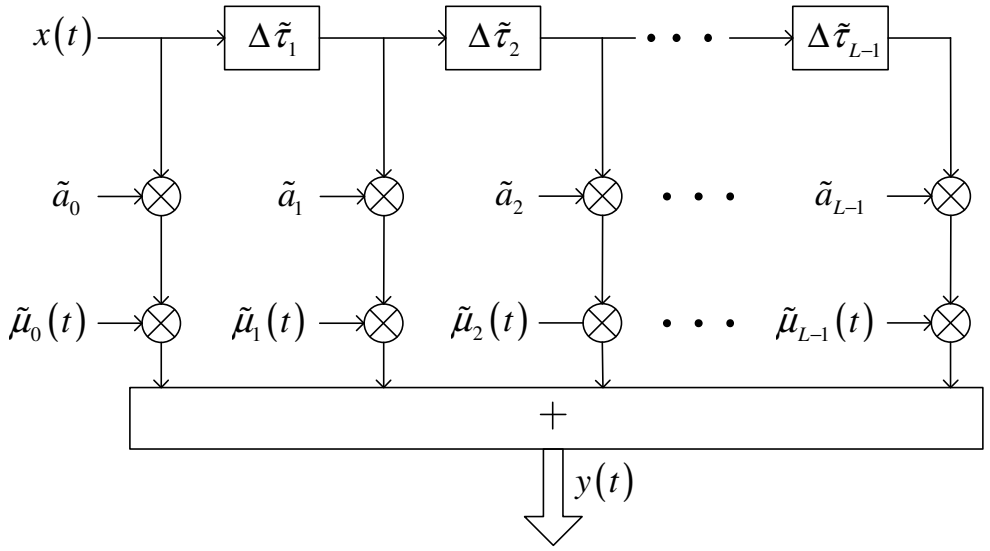

Figure 2-18: Tapped-delay line model for a frequency-selective mobile fading channel

According to figure 2-18, the tapped-delay line channel simulator is featured by the time-variant impulse response which resumes the above structure

$$\tilde{h}(\tau,t)=\sum_{\ell=0}^{L-1}\tilde{a}_\ell\tilde{\mu}_\ell(t)\,\delta(\tau-\tilde{\tau}_\ell) \tag{2.37}$$

We assume that the propagation delays and gains are statistically uncorrelated. These two important parameters can be obtained from the multipath power delay profile $\tilde{S}_{\tau'\tau'}(\tau')$. Next, we will show the relationship between the channel delays and gains and frequency correlation function as well as the multipath power delay profile according to reference model and simulation model.

2.2.1.1 Reference Model

Generally, for any kind of channel, the frequency correlation function FCF can be expressed by the following equation

$$\tilde{r}_{\tau'\tau'}(\nu') = \sum_{\ell=0}^{L-1} \tilde{a}_\ell^2 e^{-j2\pi\nu'\tilde{\tau}'_\ell} \tag{2.38}$$

The multipath power delay profile is given by

$$\tilde{S}_{\tau'\tau'}(\tau') = \sum_{\ell=0}^{L-1} \tilde{a}_\ell^2 \delta(\tau' - \tilde{\tau}'_\ell) \tag{2.39}$$

2.2.1.2 Simulation Model

The simulation model, in contrast with the reference model, takes into account the environment type. For that reason, we present in this section the expression of the FCF and the multipath power delay profile relative to each environment type.
For the RA and TU channels, the multipath power delay profile holds on the following equation

$$S_{\tau'\tau'}(\tau') = \begin{cases} c_1 \cdot e^{-b_1 \cdot \tau'}, & 0 \le \tau' < \tau'_{\max} \\ 0, & otherwise \end{cases} \tag{2.40}$$

The frequency correlation function is obtained from

$$r_{\tau'\tau'}(\nu') = c_1 \cdot \frac{1 - e^{-\tau'_{\max}(b_1 + j2\pi\nu')}}{b_1 + j2\pi\nu'} \tag{2.41}$$

For the BU and HT, we get the following equations respectively for the multipath power delay profile

$$S_{\tau'\tau'}(\tau') = \begin{cases} c_1 \cdot e^{-b_1 \cdot \tau'}, & 0 \le \tau' < \tau'_1 \\ c_2 \cdot e^{b_2 - b_3 \cdot \tau'}, & \tau'_2 \le \tau' < \tau'_{\max} \\ 0, & otherwise \end{cases} \tag{2.42}$$

$$r_{\tau'\tau'}(\nu') = c_1 \cdot \frac{1 - e^{-\tau'_{\max}(b_1 + j2\pi\nu')}}{b_1 + j2\pi\nu'} + c_2 \cdot e^{b_2} \cdot \frac{e^{-\tau'_2(b_3 + j2\pi\nu')} - e^{\tau'_{\max}(b_3 + j2\pi\nu')}}{b_3 + j2\pi\nu'} \tag{2.43}$$

Where the constants used in the above equations are given in the following table. It considers the four environments specified by COST 207

Environment	c_1	c_2	b_1	b_2	b_3	$\tau_1'(\mu s)$	$\tau_2'(\mu s)$	$\tau_{\max}'(\mu s)$
Rural area (RA)	c_{RA}	-	9.2	-	-	-	-	0.7
Typical urban (TU)	c_{TU}	-	1	-	-	-	-	7
Bad urban (BU)	c_{BU}	$0.5\,c_{BU}$	1	5	1	5	5	10
Hilly terrain (HT)	c_{HT}	$0.1\,c_{HT}$	3,5	15	1	2	15	20

Where

$$c_{RA} = \frac{9.2}{1-e^{-6.44}}$$

$$c_{TU} = \frac{1}{1-e^{-7}}$$

$$c_{BU} = \frac{2}{3\left(1-e^{-5}\right)}$$

$$c_{HT} = \frac{1}{\left(1-e^{-7}\right)/3.5 + \left(1-e^{-5}\right)/10}$$

2.2.2 Parameters Computation Methods

Similar to section 2.1.1, we will go through the different computation methods that are mostly used to compute the discrete delays and discrete power delay profile. Our inspiration is taken mainly from [2] and [21].

2.2.2.1 Method of Equal Distances (MED)

The idea behind the MED method is previously presented in section 2.1.1.1. However, in this section we just consider the discrete delays and gains. Therefore, we compute the discrete delays and gains according to MED concept.

The discrete delays:

For RA, TU and BU environment, the discrete delays are given by

$$\tilde{\tau}_\ell' = \ell \cdot \Delta\tilde{\tau}_\ell', \qquad \ell = 0,1,\ldots,L-1 \tag{2.44}$$

where

$$\Delta\tilde{\tau}_\ell' = \frac{\tau_{\max}'}{L-1} \tag{2.45}$$

For HT environment, they are expressed by

$$\tilde{\tau}_\ell' = \begin{cases} \ell \cdot \Delta\tilde{\tau}_{\ell HT}', & \ell = 0,1,\ldots,L_1 \\ \left(\ell + L_2 - L_1 - 1\right)\cdot \Delta\tilde{\tau}_{\ell HT}', & \ell = L_1+1, L_1+2,\ldots,L-1 \end{cases} \tag{2.46}$$

where

$$\Delta\tilde{\tau}'_{\ell HT} = \frac{\tau'_{\max} - \tau'_2 + \tau'_1}{L-1} \tag{2.47}$$

$$L_1 = \left\lfloor \frac{\tau'_1}{\Delta\tilde{\tau}'_{\ell HT}} \right\rfloor \tag{2.48}$$

$$L_2 = round\left\{ \frac{\tau'_2}{\Delta\tilde{\tau}'_{\ell HT}} \right\} \tag{2.49}$$

NB: round is a an operator that rounds a real-valued number towards nearest integer, and $\lfloor \cdot \rfloor$ rounds the elements of x to the nearest integers towards minus infinity

The discrete gains:

For RA and TU environment, the discrete gains are split under some conditions which are shown here

$$\tilde{a}_\ell = \begin{cases} \sqrt{\frac{c_1}{b_1}\left[1 - e^{-b_1 \Delta\tilde{\tau}'_\ell/2}\right]}, & \ell = 0 \\ \sqrt{\frac{c_1}{b_1}\left[e^{-b_1(\tilde{\tau}'_\ell - \Delta\tilde{\tau}'_\ell/2)} - e^{-b_1(\tilde{\tau}'_\ell + \Delta\tilde{\tau}'_\ell/2)}\right]}, & \ell = 1,2,\ldots,L-2 \\ \sqrt{\frac{c_1}{b_1}\left[e^{-b_1(\tilde{\tau}'_{\max} - \Delta\tilde{\tau}'_\ell/2)} - e^{-b_1\tilde{\tau}'_{\max}}\right]}, & \ell = L-1 \end{cases} \tag{2.50}$$

Also for the BU environment, the expression of the discrete gains is given through intervals, as depicted here in the following equation.

$$\tilde{a}_\ell = \begin{cases} \sqrt{\frac{c_1}{b_1}\left[1 - e^{-b_1 \Delta\tilde{\tau}'_\ell/2}\right]}, & \ell = 0 \\ \sqrt{\frac{c_1}{b_1}\left[e^{-b_1(\tilde{\tau}'_\ell - \Delta\tilde{\tau}'_\ell/2)} - e^{-b_1(\tilde{\tau}'_\ell + \Delta\tilde{\tau}'_\ell/2)}\right]}, & \ell = 1,2,\ldots,L_3-1 \\ \left\{ \frac{c_1}{b_1}\left[e^{-b_1(\tilde{\tau}'_\ell - \Delta\tilde{\tau}'_\ell/2)} - e^{-b_1\tau'_1}\right] + \right. & \\ \left. \frac{c_2}{b_3} e^{b_2}\left[e^{-b_3\tau'_1} - e^{-b_3(\tilde{\tau}'_\ell + \Delta\tilde{\tau}'_\ell/2)}\right] \right\}^{1/2}, & \ell = L_3 \\ \sqrt{\frac{c_2}{b_3} e^{b_2}\left[e^{-b_3(\tilde{\tau}'_\ell - \Delta\tilde{\tau}'_\ell/2)} - e^{-b_3(\tilde{\tau}'_\ell + \Delta\tilde{\tau}'_\ell/2)}\right]}, & \ell = L_3+1,\ldots,L-2 \\ \sqrt{\frac{c_2}{b_3} e^{b_2}\left[e^{-b_3(\tau'_{\max} - \Delta\tilde{\tau}'_\ell/2)} - e^{-b_3\tau'_{\max}}\right]}, & \ell = L-1 \end{cases} \tag{2.51}$$

where

$$L_3 = round\left\{\frac{\tau_1'}{\Delta\tilde{\tau}_\ell'}\right\} \tag{2.52}$$

For HT environment, we get the following expression

$$\tilde{a}_\ell = \begin{cases} \sqrt{\frac{c_1}{b_1}\left[1-e^{-b_1\Delta\tilde{\tau}_{\ell HT}'/2}\right]}, & \ell = 0 \\ \sqrt{\frac{c_1}{b_1}\left[e^{-b_1(\tilde{\tau}_\ell'-\Delta\tilde{\tau}_{\ell HT}'/2)}-e^{-b_1(\tilde{\tau}_\ell'+\Delta\tilde{\tau}_{\ell HT}'/2)}\right]}, & \ell = 1,2,\ldots,L_1-1 \\ \sqrt{\frac{c_1}{b_1}\left[e^{-b_1(\tilde{\tau}_\ell'-\Delta\tilde{\tau}_{\ell HT}'/2)}-e^{-b_1\tau_1'}\right]}, & \ell = L_1 \\ \sqrt{\frac{c_2}{b_3}e^{b_2}\left[e^{-b_3\tau_2'}-e^{-b_3(\tilde{\tau}_\ell'+\Delta\tilde{\tau}_{\ell HT}'/2)}\right]}, & \ell = L_1+1 \\ \sqrt{\frac{c_2}{b_3}e^{b_2}\left[e^{-b_3(\tilde{\tau}_\ell'-\Delta\tilde{\tau}_{\ell HT}'/2)}-e^{-b_3(\tilde{\tau}_\ell'+\Delta\tilde{\tau}_{\ell HT}'/2)}\right]}, & \ell = L_1+2,\ldots,L-2 \\ \sqrt{\frac{c_2}{b_3}e^{b_2}\left[e^{-b_3(\tilde{\tau}_\ell'-\Delta\tilde{\tau}_{\ell HT}'/2)}-e^{-b_3\tau_{\max}'}\right]}, & \ell = L-1 \end{cases} \tag{2.53}$$

2.2.2.2 Method of Equal Areas (MEA)

The principle of MEA method is demonstrated in the section 2.1.1.3 unless the Doppler frequencies and coefficients are replaced by the discrete gains and delays.

The discrete delays:

First for the RA and TU environments the discrete delays are given by

$$\tilde{\tau}_\ell' = -\frac{1}{b_1}\cdot\ln\left(-\frac{b_1}{c_1}\sigma_d^2\frac{\ell}{L}+1\right), \quad \ell = 0,1,2,\ldots,L-1 \tag{2.54}$$

and for the BU and HT environment, the discrete delays are given by

$$\tilde{\tau}_\ell' = \begin{cases} -\frac{1}{b_1}\cdot\ln\left(-\frac{b_1}{c_1}\sigma_d^2\frac{\ell}{L}+1\right), & \ell \le L_4 \\ -\frac{1}{b_3}\cdot\ln\left(\frac{b_3}{c_2\cdot e^{b_2}}\left[A-\sigma_d^2\frac{\ell}{L}\right]+e^{-b_3\cdot\tau_2'}\right), & \ell > L_4 \end{cases} \tag{2.55}$$

where

$$A = \frac{c_1}{b_1}\left(1-e^{-b_1\cdot\tau_1'}\right) \tag{2.56}$$

$$L_4 = \left\lfloor A \cdot \frac{L}{\sigma_d^2} \right\rfloor \tag{2.57}$$

The discrete gains:

The discrete gains are equal for all environment types referring to the MEA method. The general expression of these gains is shown here

$$\tilde{a}_\ell = \frac{\sigma_d}{\sqrt{L}}, \qquad \ell = 0,1,2,\ldots,L-1 \tag{2.58}$$

2.2.2.3 Mean-Square-Error Method (MSEM)

To find the channel parameters, the MSEM method proposes to minimize the mean-square error of the FCF

$$E_{r_{\tau'\tau'}} = \frac{1}{v'_{\max}} \int_0^{v_{\max}} \left| r_{\tau'\tau'}(v) - \tilde{r}_{\tau'\tau'}(v) \right|^2 dv \tag{2.59}$$

The discrete delays:

The discrete delays for the RA, TU and BU environments are given by the equation(2.44), however, for the HT environment they are given by (2.46)

The discrete gains:

After substituting the equation (2.38) in (2.59), we get the following expression for the discrete gains whatever the environment type

$$\tilde{a}_\ell = \sqrt{\mathrm{Re}\left\{ \frac{1}{v'_{\max}} \int_0^{v'_{\max}} r_{\tau'\tau'}(v') e^{j2\pi v' \tilde{\tau}'_\ell} dv' \right\}}, \qquad \ell = 0,1,\ldots,L-1 \tag{2.60}$$

where

$$v'_{\max} = \frac{L-1}{2\tau'_{\max}} \tag{2.61}$$

2.2.2.4 Monte-Carlo Method (MCM)

According to the Monte-Carlo method, the discrete propagation delays are generated following a given probability density function which is related to the multipath power delay profile. A detailed demonstration, found in reference [2], leads to the expressions of channels gains and delays shown below.

The discrete delays:

For the RA and TU environment, the discrete delays are given by

$$\tau'_\ell = -\frac{1}{b_1} \cdot \ln\left(-\frac{b_1}{c_1} \sigma_d^2 u_\ell + 1 \right), \qquad \ell = 0,1,\ldots,L-1 \tag{2.62}$$

and for the BU and HT environment, they are expressed by

$$\tau'_\ell = \begin{cases} -\frac{1}{b_1}\cdot\ln\left(-\frac{b_1}{c_1}\sigma_d^2 u_\ell + 1\right), & \ell \le L_4 \\ -\frac{1}{b_3}\cdot\ln\left(\frac{b_3}{c_2\cdot e^{b_2}}\left[A-\sigma_d^2\right]u_\ell + e^{-b_3\cdot\tau'_2}\right), & \ell > L_4 \end{cases} \tag{2.63}$$

The discrete gains:

The delays coefficients are given by (2.58) for all environment types.

2.2.2.5 L_p-norm Method (LPNM)

As the MSEM method, the LPNM method requires optimizing an error function which is called L_p-norm

$$E_{r_{\tau'\tau'}}^{(p)} = \left\{\frac{1}{v'_{\max}}\int_0^{v'_{\max}}\left|r_{\tau'\tau'}(v') - \tilde{r}_{\tau'\tau'}(v')\right|^p dv'\right\}^{1/p} \tag{2.64}$$

where

$$v'_{\max} = \frac{L-1}{2\tau_{\max}} \tag{2.65}$$

Thus, to find the channels gains as well as delays we just need to optimize the above function by setting both sets of parameters $\{\tilde{\tau}'\}_{\ell=0}^{L-1}$ and $\{\tilde{a}'_\ell\}_{\ell=0}^{L-1}$ to initial values using the MSEM method.

2.2.3 Implementation and Results

In this section, we present the way of implementing our toolbox for the computation of frequency-selective channels parameters. The figure 2-19 shows the structure that we proceed to carry out the best and optimized solution. Mainly, our approach is simple to exploit so that the user can modify the parameters entries and figure out the results by looking at the autocorrelation function of both the simulation model. In addition, the results give us a better idea of the efficiency of the method. The present figure exhibits the structure of the toolbox.

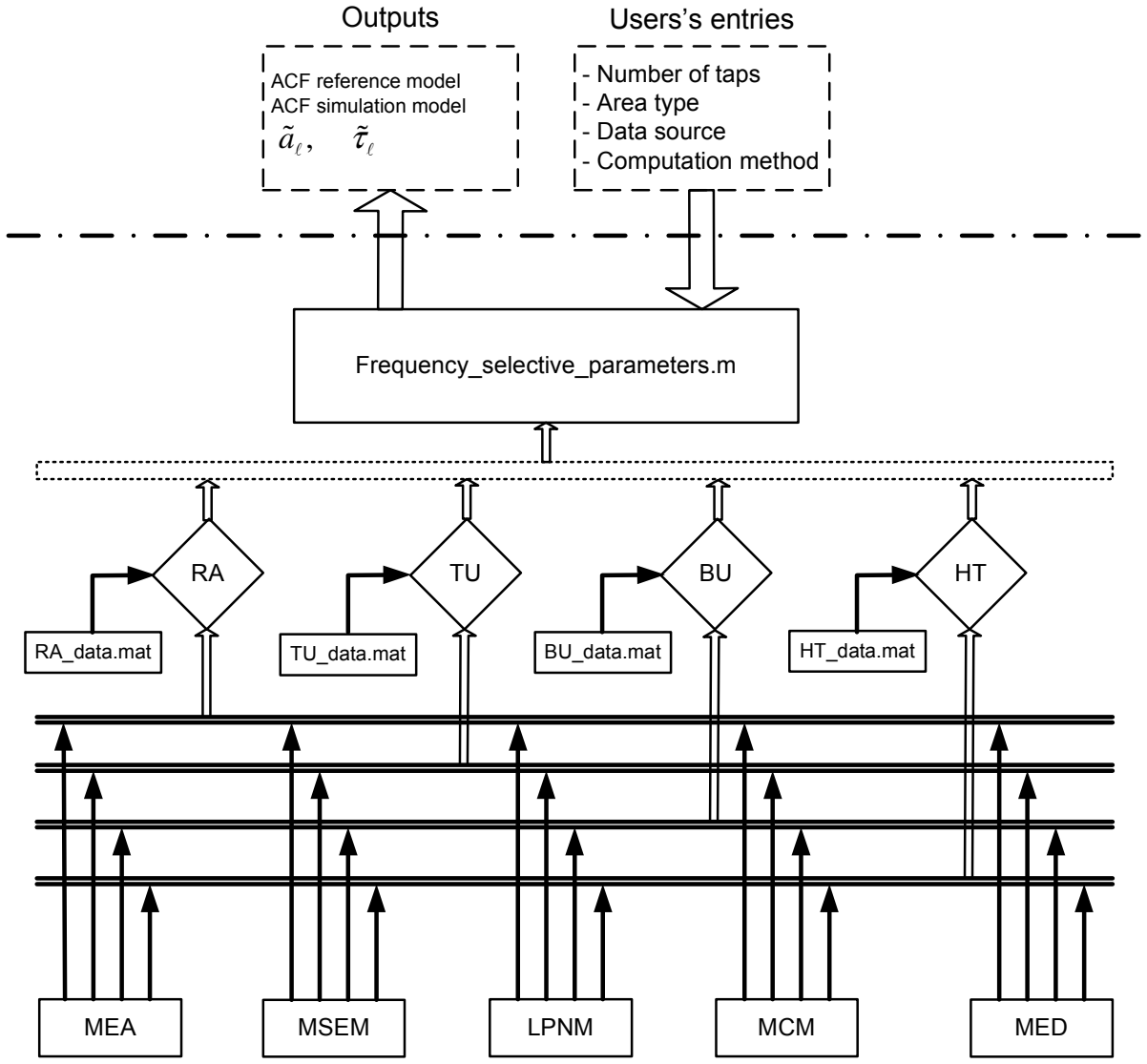

Figure 2-19: The implementation diagram of the parameters computation methods for the frequency-selective channel

We divided the program into five parts according to the number of methods. Afterwards, each method is split into four subprograms relative to each environment type.
The LPNM method is related to five subprograms which are errorfuncRA.m, errorfuncTU.m, errorfuncBU.m, and errorfuncHT.m to compute the error function separately.
The data relative to each environment type is stored into a matrix file in order to keep track of the parameters values. As the number of the environment type, the matrix files are of a number of four, which are RA_data.mat, TU_data.mat, BU_data.mat, and HT_data.mat.
Here, we present some results gotten from our implementation. The following figures depict the frequency correlation function of the reference model towards the frequency correlation function of the simulation. The fitness of both simulation and reference model shows how far the method fulfils our requirements. As parameters values, we take for number of ellipses 18 and as standard, the COST 207.

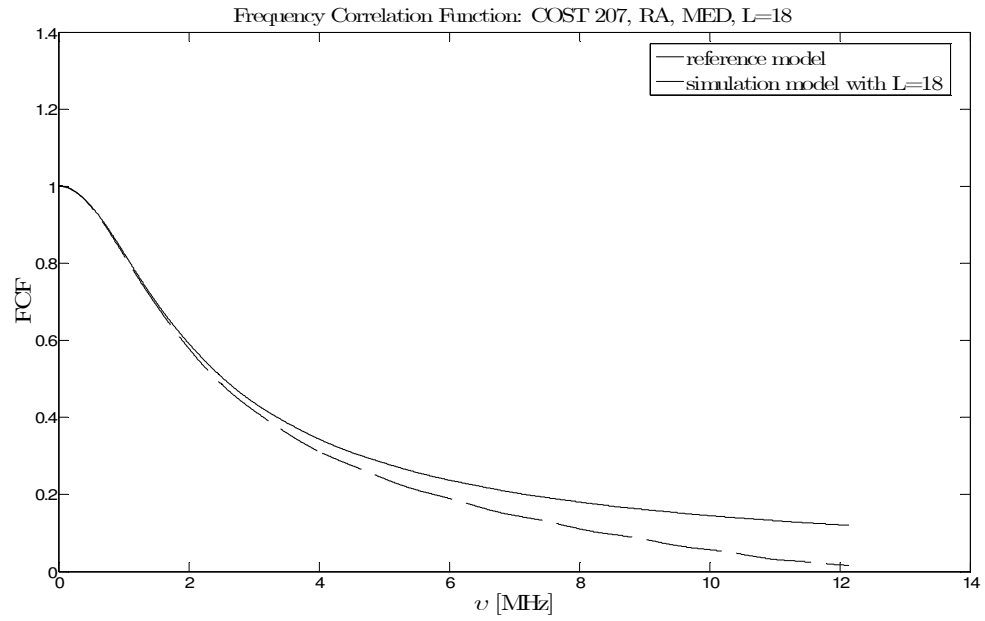

Figure 2-20: The magnitude of the FCF with (RA environment, MED method)

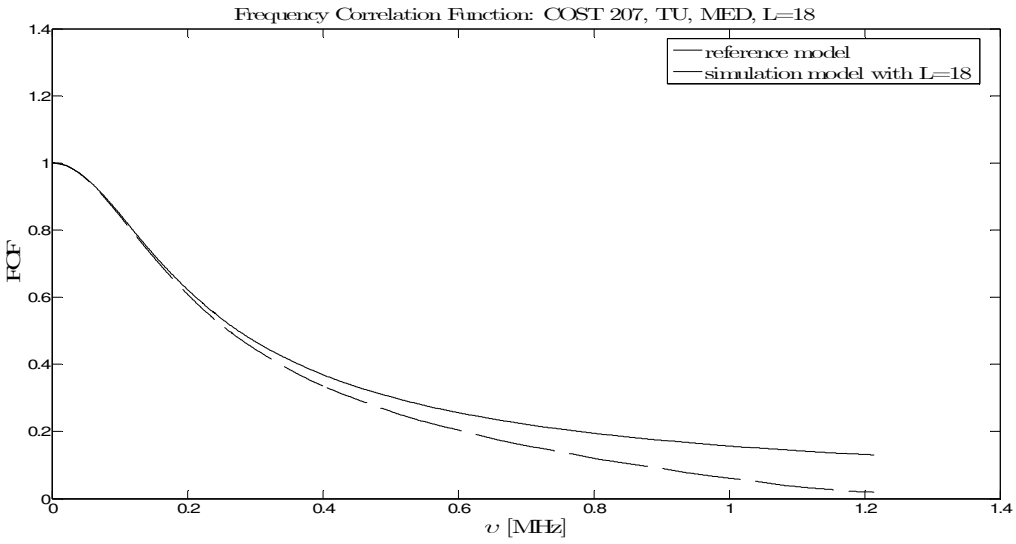

Figure 2-21: The magnitude of the FCF with (TU environment, MED method)

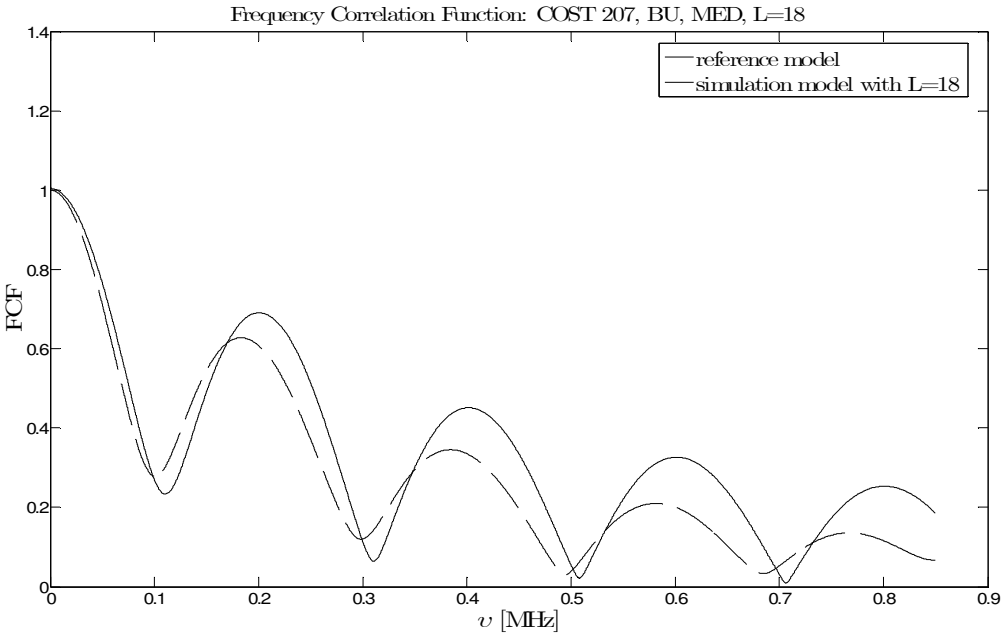

Figure 2-22: The magnitude of the FCF with (BU environment, MED method)

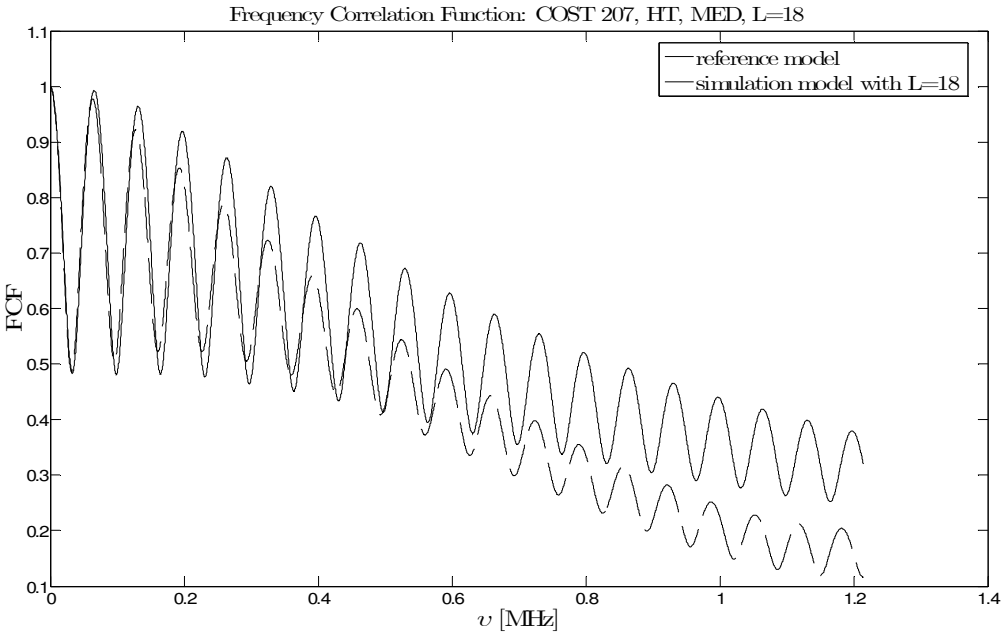

Figure 2-23: The magnitude of the FCF with (HT environment, MED method)

We remark that the MED method has some limitations:

- The shapes didn't overlap until $\tau_{\max}/2$
- The same performance with all environment types
- The error increases when the frequency gets higher

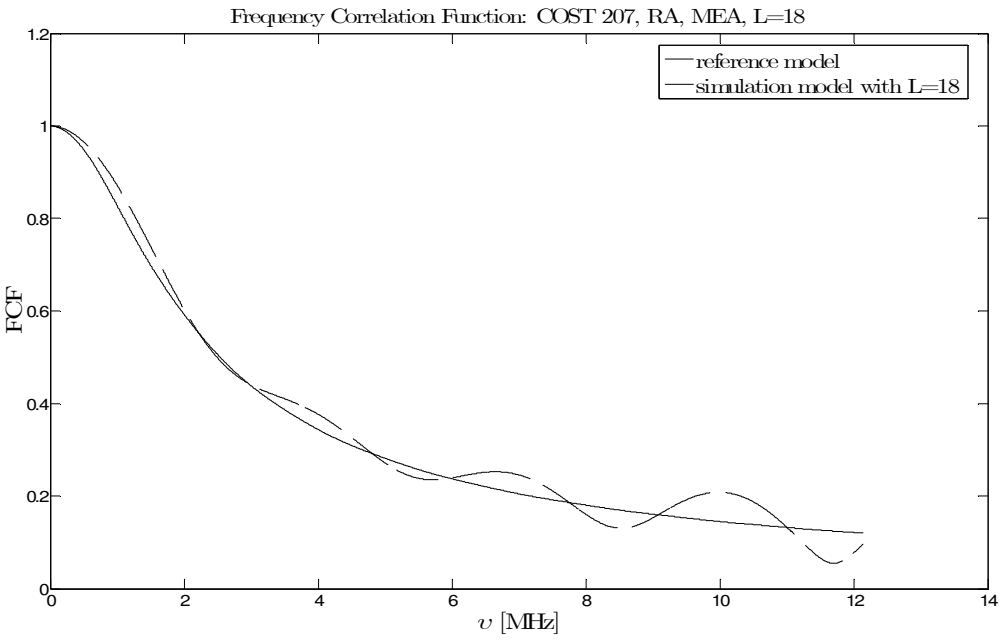

Figure 2-24: The magnitude of the FCF with (RA environment, MEA method)

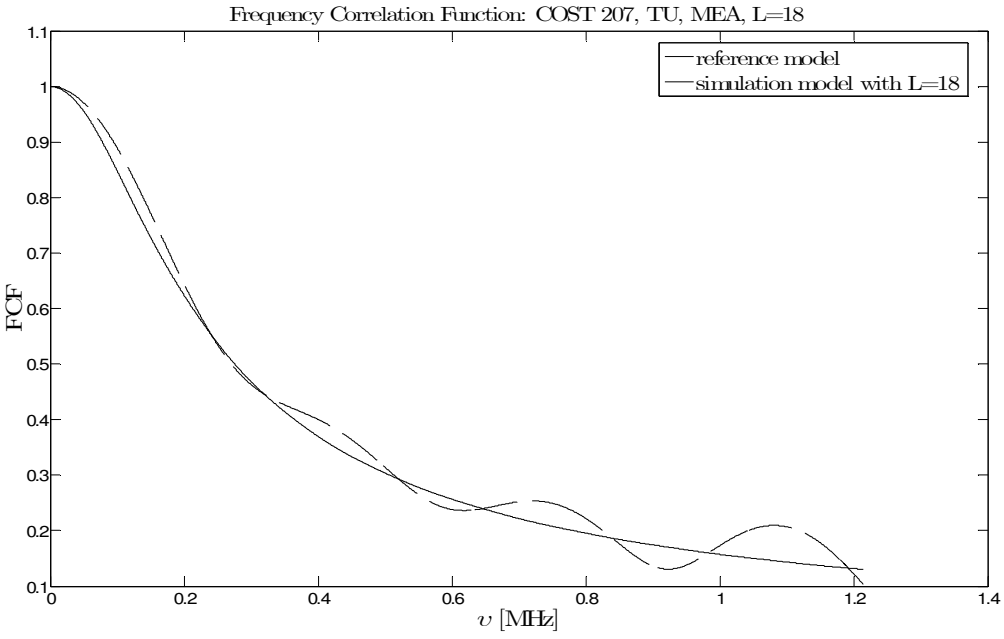

Figure 2-25: The magnitude of the FCF with (TU environment, MEA method)

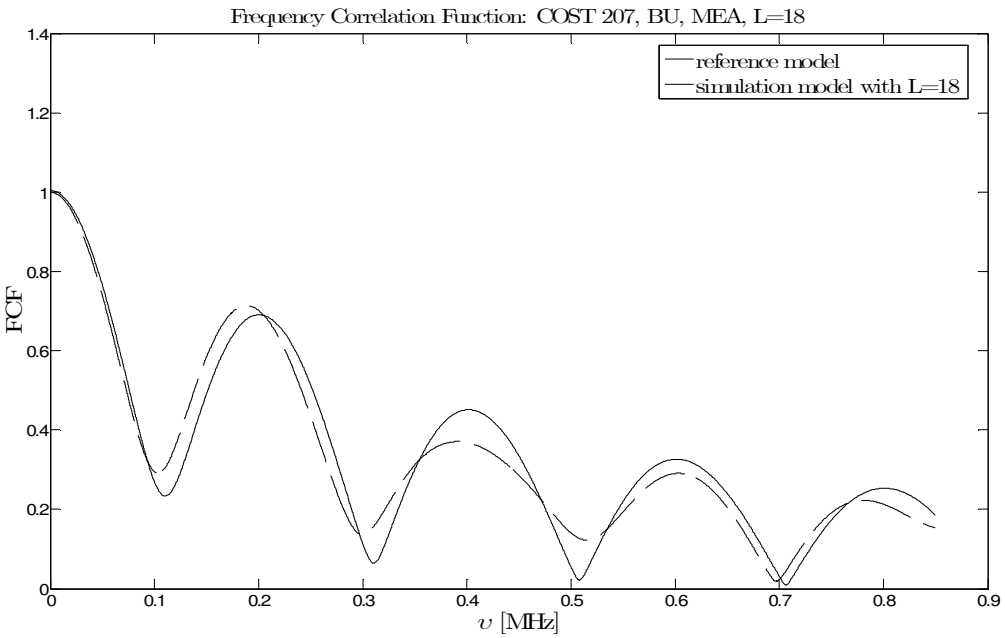

Figure 2-26: The magnitude of the FCF with (BU environment, MEA method)

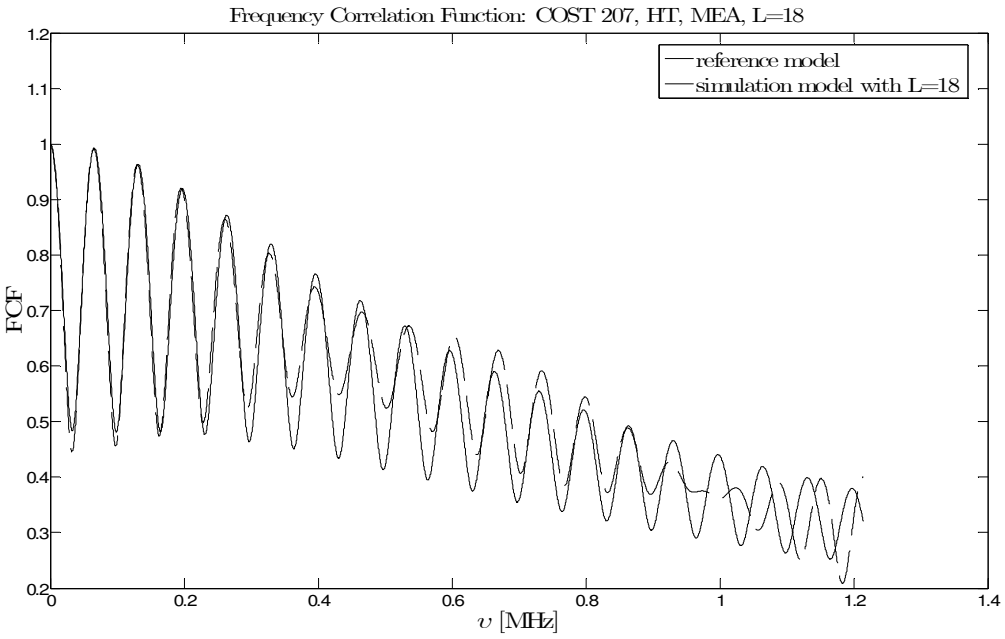

Figure 2-27: The magnitude of the FCF with (BU environment, MEA method)

Contrary to the MED, the MEA method improves the performance unless some drawbacks come up:

- The shapes didn't overlap until $\tau_{\max}/2$
- The same performance with all environment types
- The error increases when the frequency gets higher

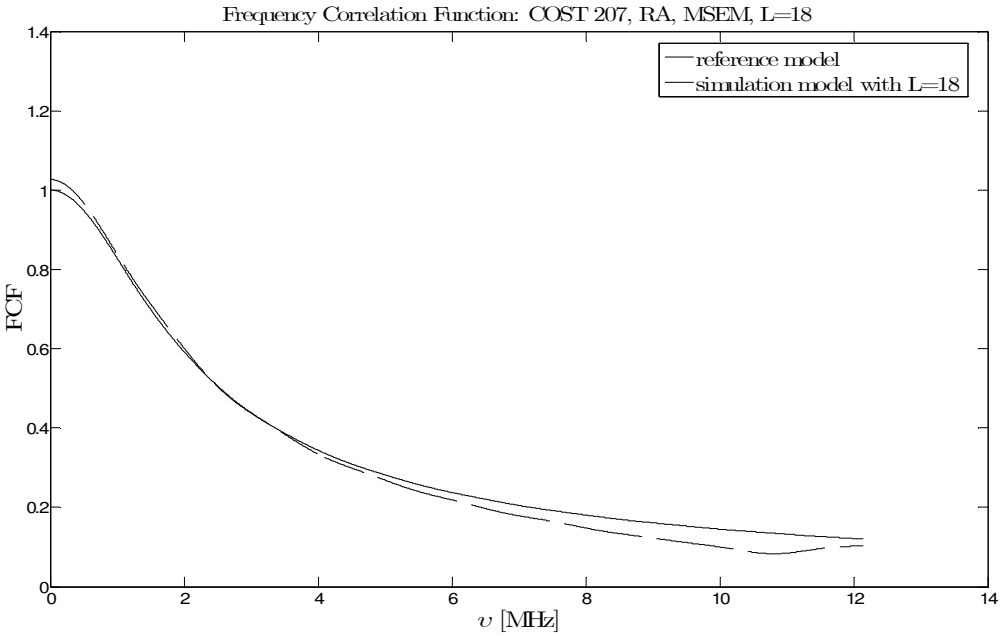

Figure 2-28: The magnitude of the FCF with (RA environment, MSEM method)

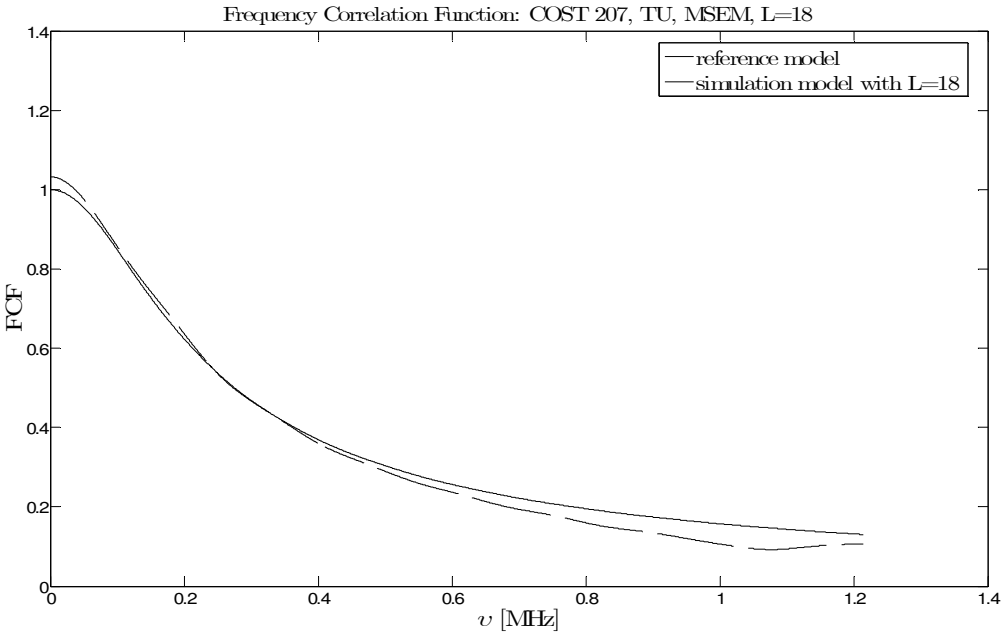

Figure 2-29: The magnitude of the FCF with (TU environment, MSEM method)

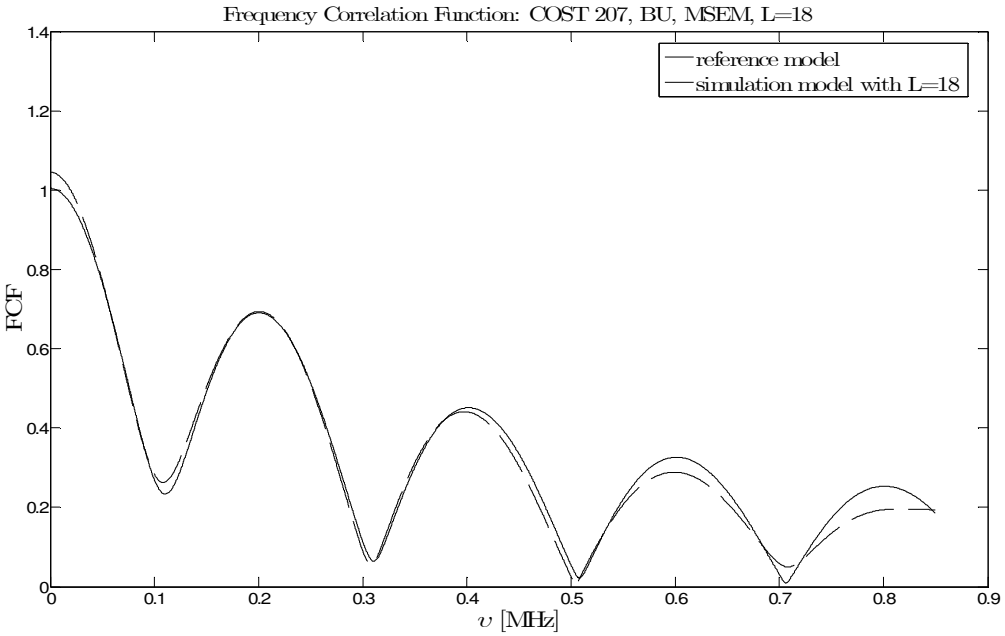

Figure 2-30: The magnitude of the FCF with (BU environment, MSEM method)

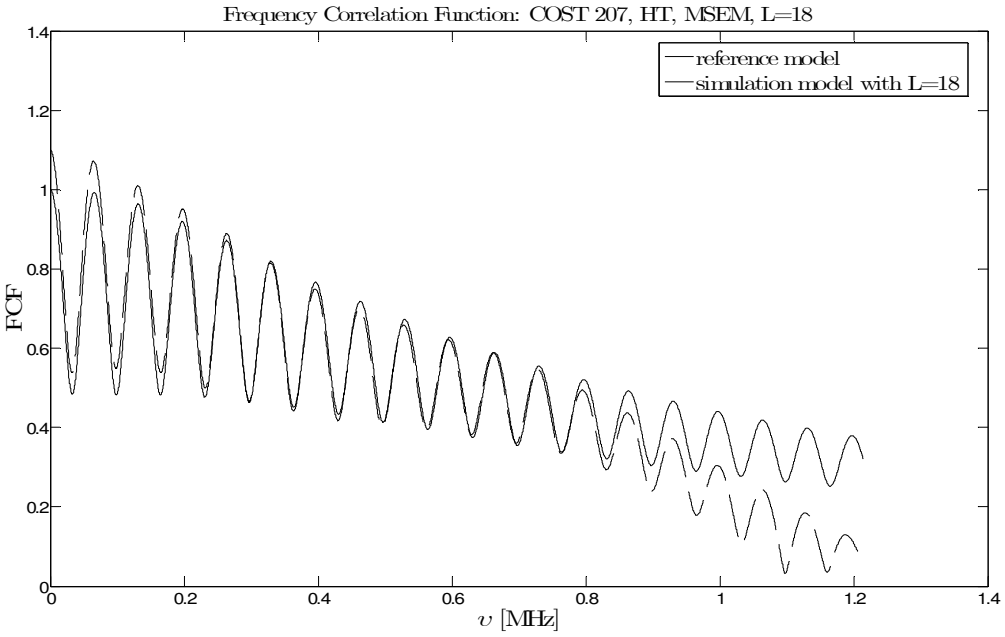

Figure 2-31: The magnitude of the FCF with (HT environment, MSEM method)

Since the MSEM method is based on the minimization of the error function between the simulation model and the reference model, the shapes must be closed. Therefore, we get almost the best fitting:

- The shapes reach a good fitting until $\tau_{max}/2$
- The same performance with all environment types
- Some errors still exist, but it could be neglected

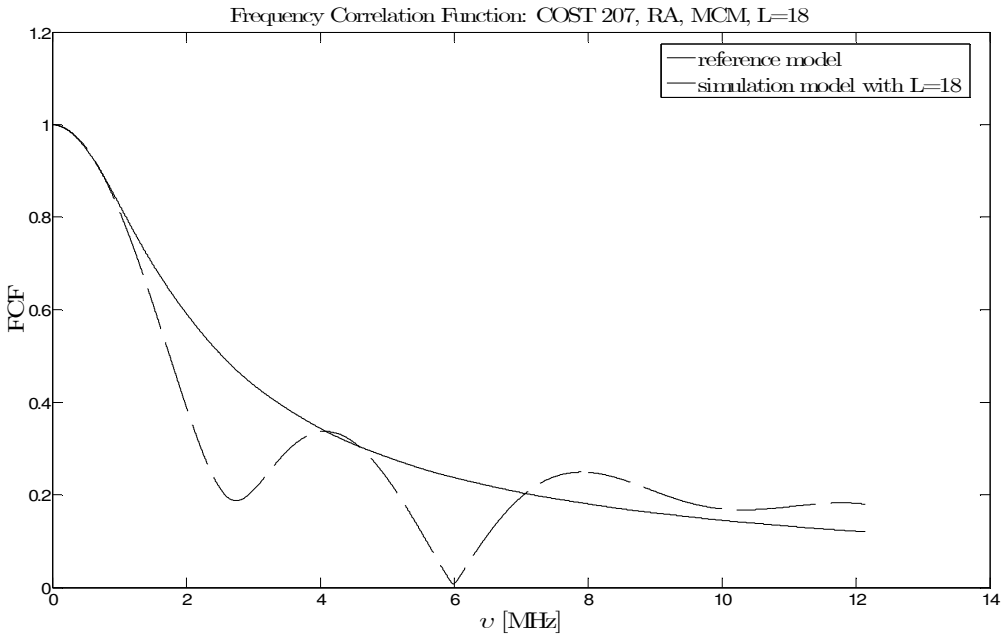

Figure 2-32: The magnitude of the FCF with (RA environment, MCM method) for one realization

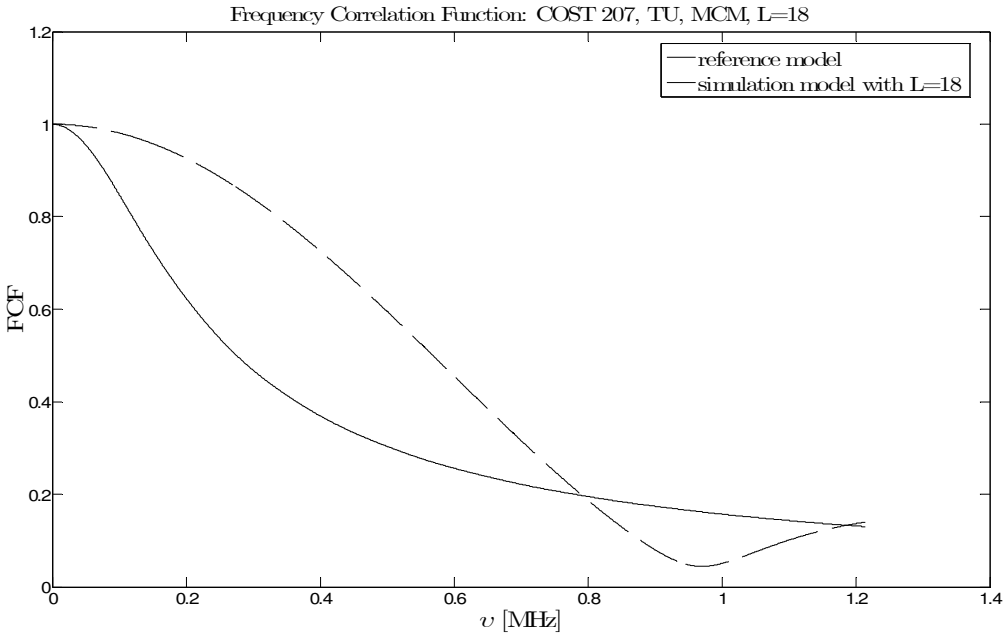

Figure 2-33: The magnitude of the FCF with (TU environment, MCM method) for one realization

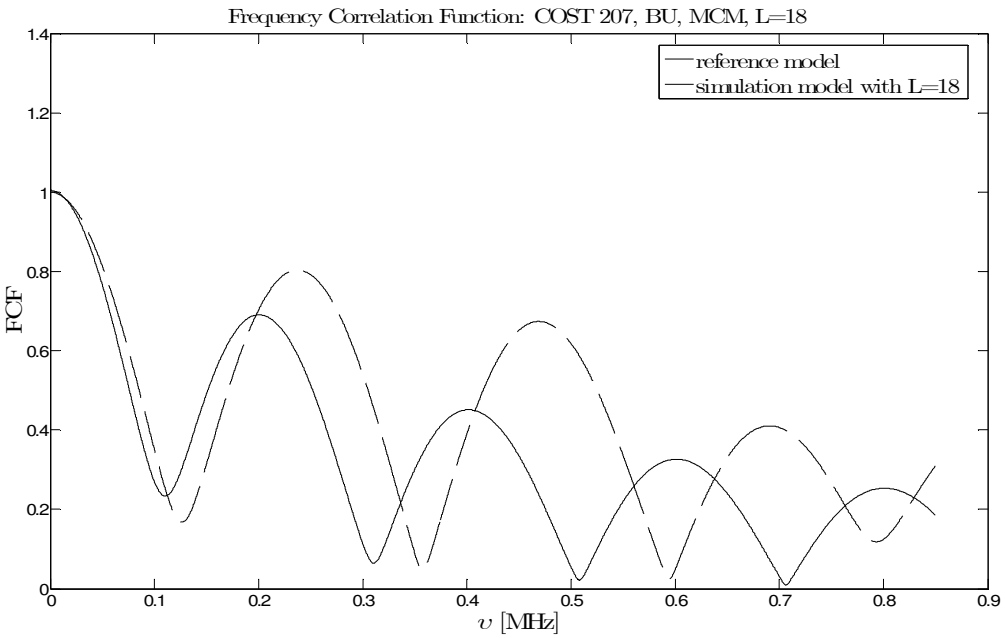

Figure 2-34: The magnitude of the FCF with (BU environment, MCM method) for one realization

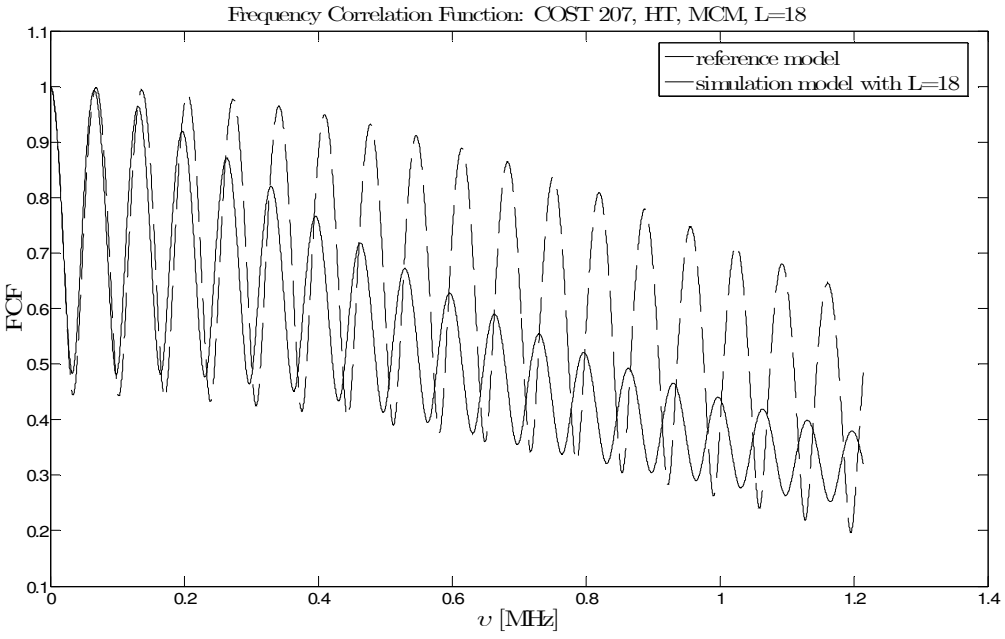

Figure 2-35: The magnitude of the FCF with (BU environment, MCM method) for one realization

The MCM method needs to be performed many realizations. The inconvenient of this method persists on time consuming when the number of realization gets higher.

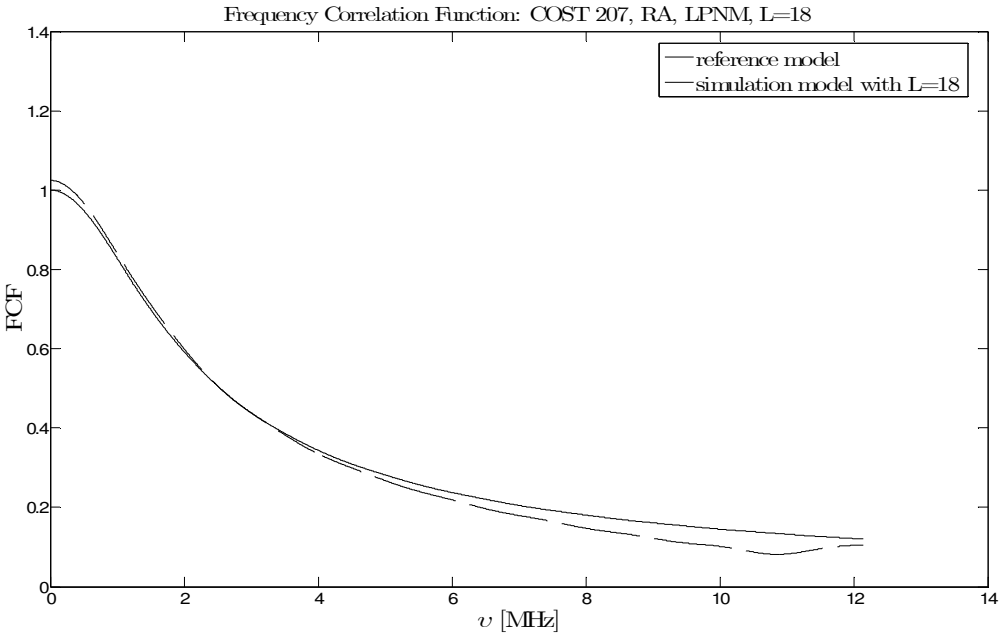

Figure 2-36: The magnitude of the FCF with (RA environment, LPNM method)

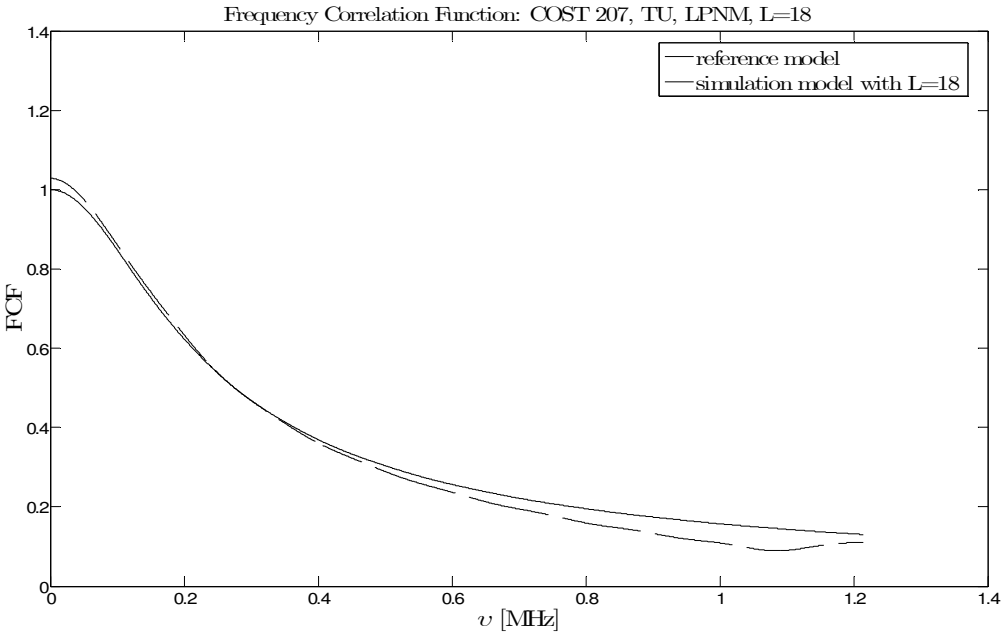

Figure 2-37: The magnitude of the FCF with (TU environment, LPNM method)

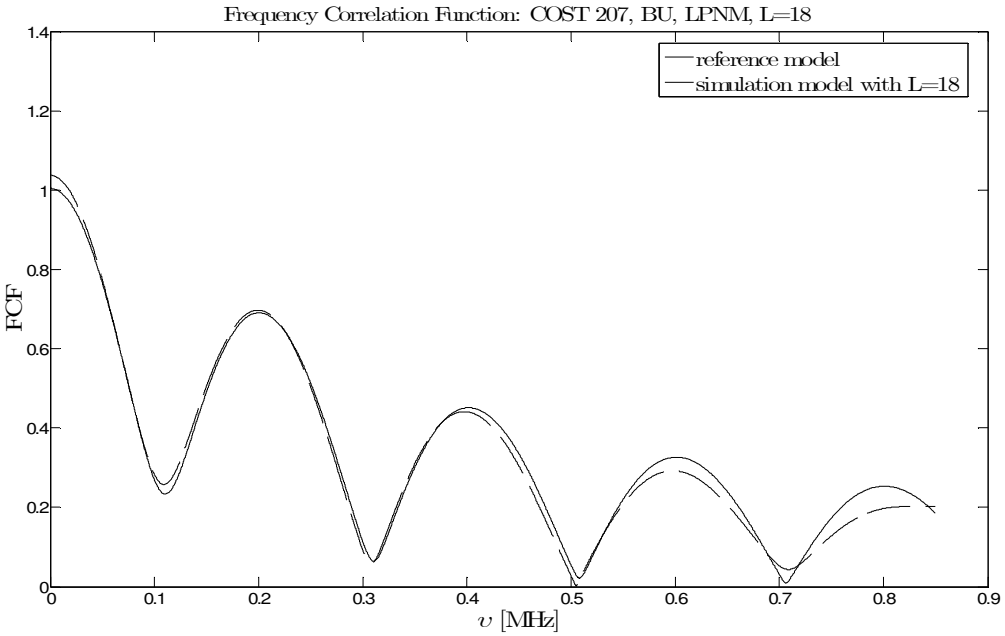

Figure 2-38: The magnitude of the FCF with (BU environment, LPNM method)

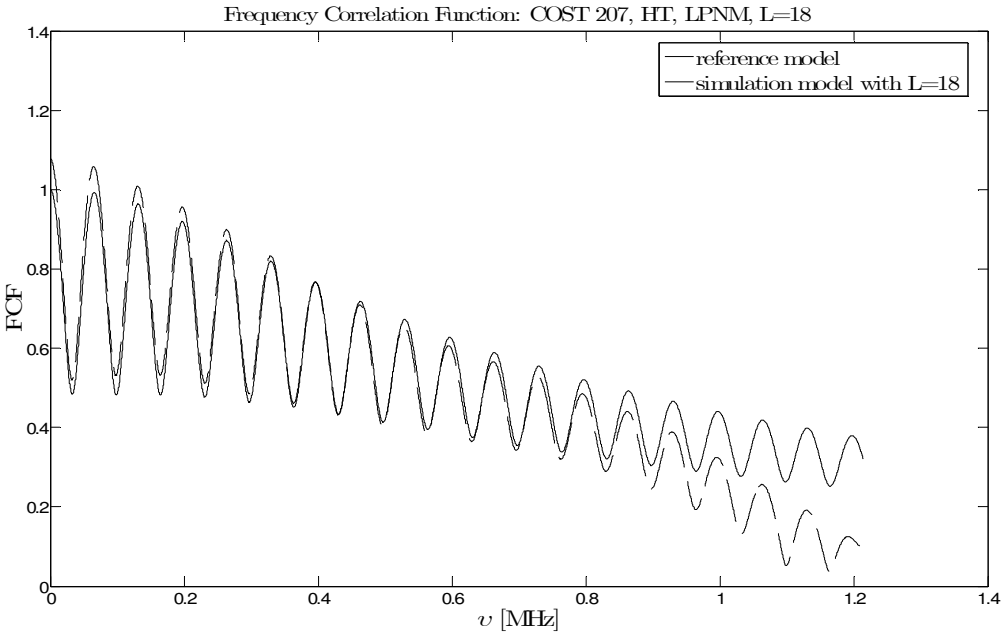

Figure 2-39: The magnitude of the FCF with (HT environment, LPNM method)

The LPNM method presents the best performance since it tries to improve the results of the MSEM method. However, a small improvement has been achieved.
All programs relatively to each method implementation are appended in Appendix 1.

2.3 CONCLUSION

As a conclusion we can assume that the mobile channel fading is recognized in two families of channels: the frequency-nonselective channels, and the frequency-selective channel. In the first part, we described the different well-known methods (MED, MEA, MSEM, MCM, LPNM, MEDS, JM, R-MEDS, MEDS-sp), which are used to compute the channels' parameters. Nonetheless, to ensure the efficiency of each method separately, we plotted the auto-correlation function of both the simulation model and the reference in order to evaluate their closeness. The second part deals with the parameters computation for the frequency-selective channels where we especially used 5 methods (MED, MEA, MSEM, MCM, LPNM). In addition, we studied their performance using the frequency correlation function of both the simulation model and the reference model.
In the next chapter, we will introduce some channel simulators using the parameters determined in this chapter.

CHAPTER 3 - MOBILE FADING CHANNEL SIMULATORS

In this chapter, we introduce the spatial shadowing processes and some adequate models like Gudmundson, Butterworth and Gauss models. The first step is to define the shadowing phenomenon. The statistical characteristics like autocorrelation function, level-crossing rate, and average duration of fades are partially shown here. Next, we simulate a shadowing process based on the channel parameters (gains and spatial frequencies) obtained from the MEA method. The second part of this chapter deals with MIMO channel simulators. In this section, we describe briefly the different procedures for finding the different models (one ring, tow ring and elliptical).
It should be noted that the models presented here are a summary work of different articles. Thus, as mentioned before, our tasks during this thesis are to adapt the previous works with the Matlab environment. Thereby, we develop a new toolbox fitted for the modelling of mobile radio channels. The last part of this chapter is concerned with the implementation of different simulators as well as the obtained results.

3.1 SPATIAL SHADOWING CHANNEL SIMULATOR

The so-called shadowing process is recognized as the long-term effects caused by the natural and artificial obstacles located through the way between the base station and the user's mobile station. This phenomenon can be modelled by lognormal processes. It has been shown in [1] that the lognormal processes can be obtained by transforming a Gaussian process. This section is divided into two parts. The first part reports the description of the reference model, while the rest deals with the simulation model.

3.1.1 Reference Model

In this section, we define the probability density function as well as the cumulative distribution function of the shadowing process. Next, we introduce the level-crossing rate and average duration of fades which are considered as advanced statistics.

3.1.1.1 The PDF and CDF of spatial shadowing process

The spatial shadowing process of the reference model is given by

$$\lambda(x)=10^{(\sigma_L \nu(x)+m_L)/20} \tag{3.1}$$

where the quantities:
m_L : stands for area mean
σ_L : stands for shadow standard deviation
The probability density function PDF of the shadowing process is expressed as the following:

$$P_\lambda(x)=\frac{20}{\sqrt{2\pi}\ln(10)\sigma_L z}e^{\frac{-(20\ln(z)-m_L)^2}{2\sigma_L^2}},\ z\geq 0 \tag{3.2}$$

From the above expression, we determine the cumulative distribution function CDF as the integration over an interval $[-\infty, x]$ of the PDF.

$$F_\lambda(r)=P_r(\lambda(x)\leq r)=\int_0^r P_r(z)\,dz \tag{3.3}$$

The lognormal process plays an important role and it is considered as a key in the performance, analysis of handover procedures as explained in [7]. These processes are merely described by two fundamental parameters. Later on, we demonstrate some statistical and theoretical properties of the spatial lognormal processes.
Neither the probability density function nor the cumulative distribution function provides any information on how fast the processes are changing from one level to another. However, the statistical computation of how fast the process is changing could be efficient information to analyse the phenomenon. Therefore, level-crossing as the number of ups and downs within as specific interval is a significant testing parameter.

3.1.1.2 Level-Crossing Rate and Average Duration of Fades

As defined in [1], the level crossing rate is the number of up or downs crossings through a given level r over a time interval. Later, this should be large enough that we obtain a good approximation of LCR. After some steps of calculation, we derive the following formula corresponding to the spatial shadowing process.

$$N_\lambda(r)=\frac{\sqrt{\gamma}}{2\pi}e^{\frac{(20\ln(r)-m_L)^2}{2\sigma_L^2}},\ r\geq 0 \tag{3.4}$$

where

$$\gamma=-\frac{d^2}{d\Delta x^2}r_{vv}(\Delta x)\big|_{\Delta x=0}=-r_{vv}^{"}(0) \tag{3.5}$$

$$T_{\lambda-}(r)=\frac{F_{\lambda-}(r)}{N_\lambda(r)} \tag{3.6}$$

3.1.2 Simulation Model

Applying the sum-of-sinusoids principal denoted by rice

$$\tilde{v}(x)=\sum_{n=1}^{N}c_n\cos(2\pi\alpha_n x+\theta_n) \tag{3.7}$$

where c_n and α_n are constants

θ_n : Random variable following a uniform distribution over $[0, 2\pi]$.

$$\begin{aligned}\tilde{r}_{\nu\nu}(\Delta x) &= E\{\tilde{\nu}(x)\tilde{\nu}(x+\Delta x)\} \\ &= \sum_{n=1}^{N}\frac{{c_n}^2}{2}\cos(2\pi\alpha_n\Delta x)\end{aligned} \tag{3.8}$$

$$\tilde{p}_\nu(x) = \begin{cases} 2\int_0^\infty\left[\prod_{n=1}^{N}J_0(2\pi c_n z)\right]\cos(2\pi xz)\,dz & ,x\in[\hat{v}_{\min},\hat{v}_{\max}] \\ 0 & ,otherwise\end{cases} \tag{3.9}$$

where $c_n = \sqrt{2/3}$.

The probability $\tilde{p}_v(x)$ equals zero if x doesn't belong to the range $[\hat{v}_{\min},\hat{v}_{\max}]$ such that $\hat{v}_{\max} = -\hat{v}_{\min} = \sqrt{2N}$.

By applying the transformation of random variable we obtain:

$$\begin{aligned}\hat{p}_\lambda(y) &= \frac{20\hat{p}_v\left(\frac{20\log_{10}y - m_L}{\sigma^2}\right)}{y\sigma_L\ln 10} \\ &= \frac{40}{\sigma_L\ln 10\, y}\int_0^\infty\left[\prod_{n=1}^{N}J_0(2\pi c_n z)\right]\cos\left[\frac{2\pi z(20\log_{10}y - m_L)}{\sigma_L}\right]dz\end{aligned} \tag{3.10}$$

And finally the level-crossing rate is defined as:

$$\hat{N}_\lambda(r) = \frac{\sqrt{\hat{\gamma}}}{2\pi}e^{\frac{-(20\log_{10}r - m_L)}{2\sigma_L^2}}\quad ,r\geq 0 \tag{3.11}$$

where $\hat{\gamma} = 2\pi^2\sum_{n=1}^{N}(c_n\alpha_n)^2$.

The schema below describe the structure of the spatial shadowing simulator

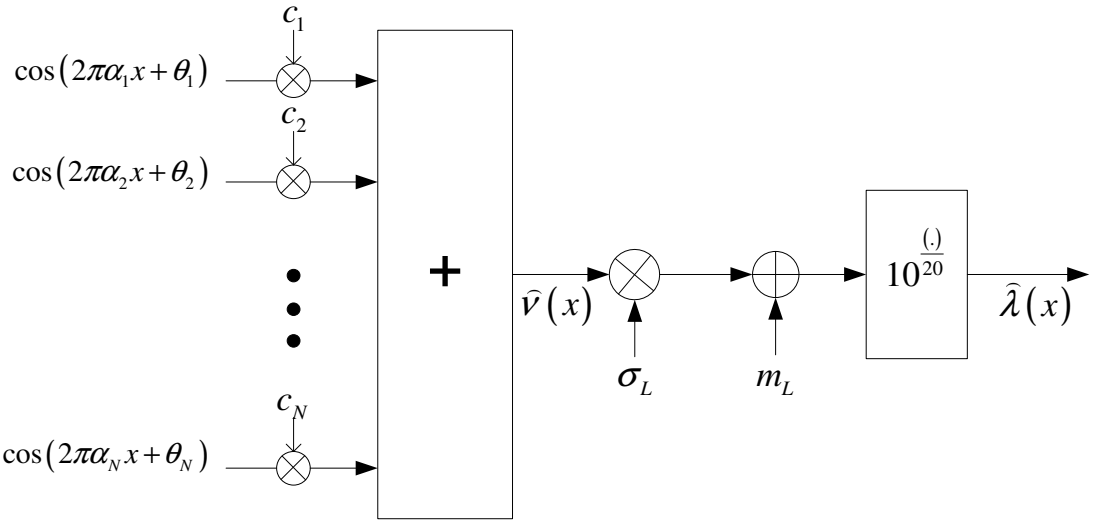

Figure 3-1: structure of a shadowing processes simulator

3.1.3 Implementation

The shadowing processes simulator shown in the figure 3.1 will be implemented under the Matlab environment as a set of m-files containing the different functions and procedures. In the figure below, we described the possible interactions between the user and our program or simulator. The implementation was organized following a logical structure.

- Computation of the parameters: the parameters aimed at here are mostly the channel gains and delays such that we considered the non-selective case of channel modelling. They are calculated based on some predefined models, such as the Gudmundson's model, Butterworth's model and Gauss's model, which are explained more in [5]. Furthermore, some other parameters are supplied as fixed entries according to the model, for instance, D, m_L and σ_L^2
- Determination of the shadowing processes: the parameters computed in the first step, using the program shadowing parameters.m, should be provided as input to the shadowing processes.m program. The user has to give only the number of scatters, model type, environment type, simulation time, and simulation time step.
- Testing of performance: in order to ensure that our models are correct and converge to the reference model, we provide some programs for testing some statistical characteristic of the process. As characteristics we implement PDF, CDF, LCR, ADF and ACF, each in a single matlab file.

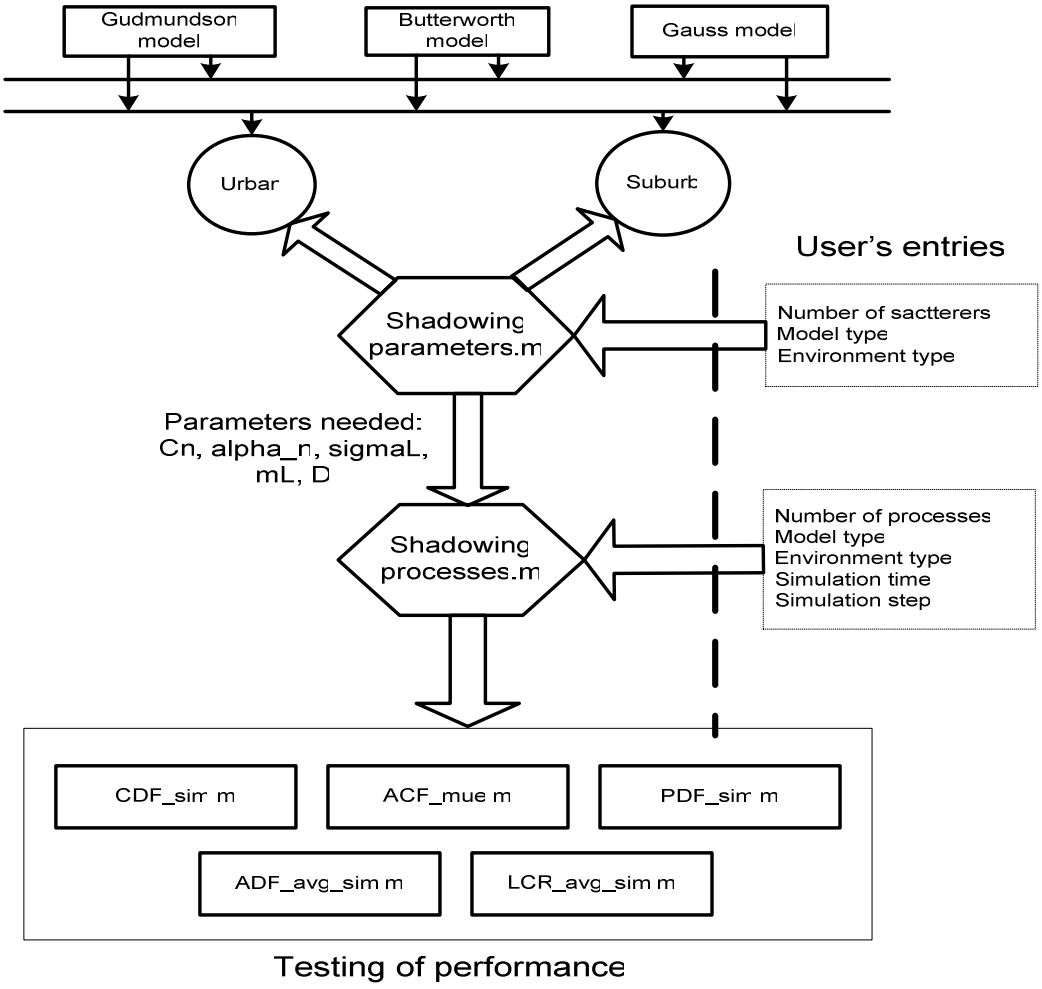

Figure 3-2: The diagram for the implementation of the spatial shadowing process

3.1.4 Results and Interpretation

As a result of the above programs, we get the following results. The figures present different shapes of the autocorrelation function referring to reference model and simulation model. The number of scatters that are supposed to exist between the transmitter and the receiver are fixed to N=20, and we picked up MEA (method of equal areas) to compute the channel parameters. The first set of ACF's shapes is plotted under the urban environment.

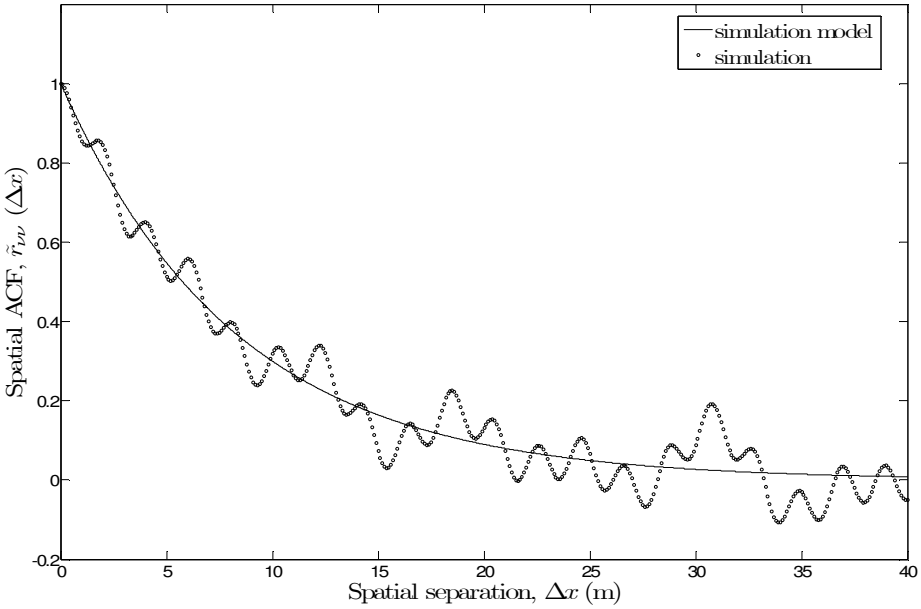

Figure 3-3: ACFs of the spatial shadowing process using N=20, MMEA method, Gudmundson computation model for Urban environment

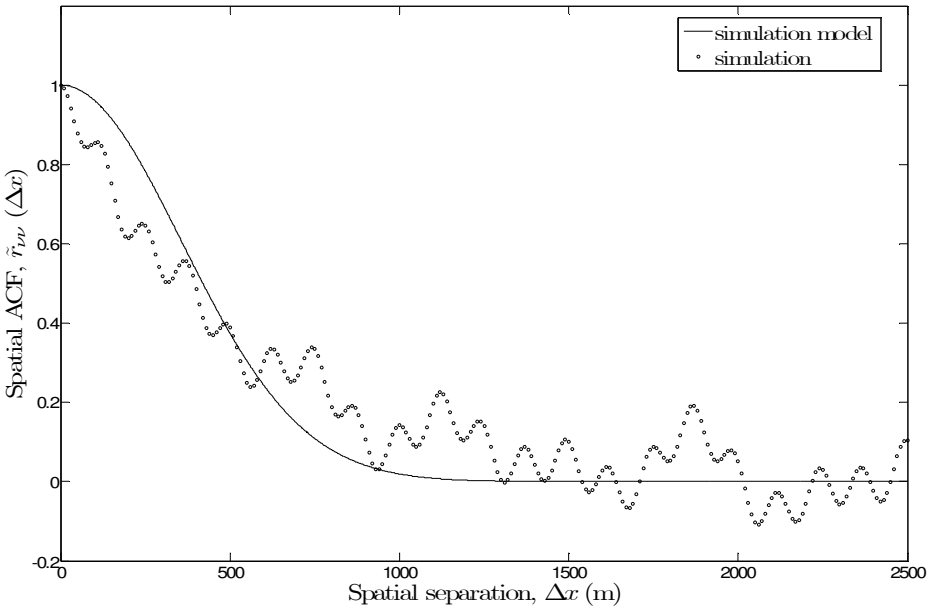

Figure 3-4: ACFs of the spatial shadowing process using N=20, MMEA method, Gudmundson computation model for Suburban environment

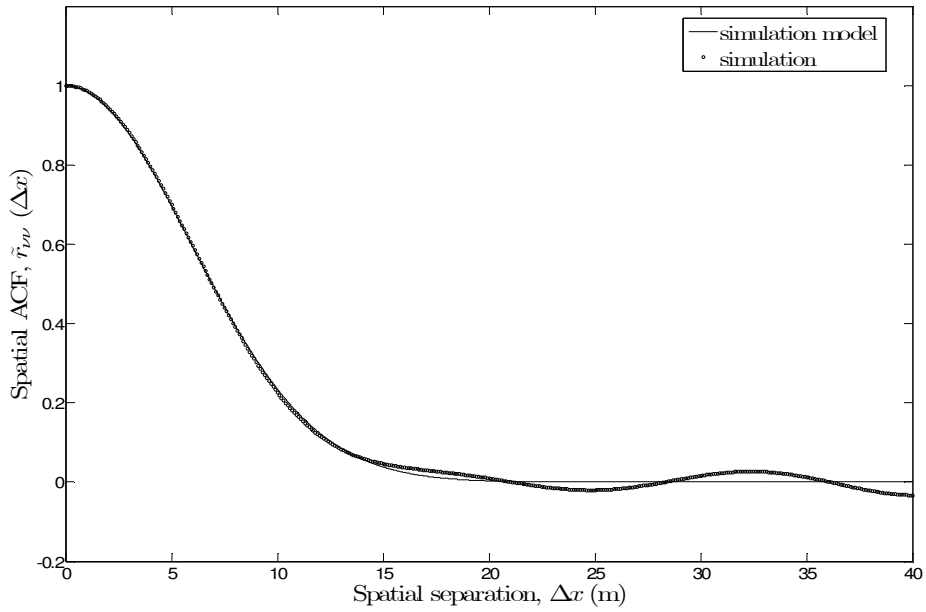

Figure 3-5: ACFs of the spatial shadowing process using N=20, MMEA method, Gaussian computation model for Urban environment

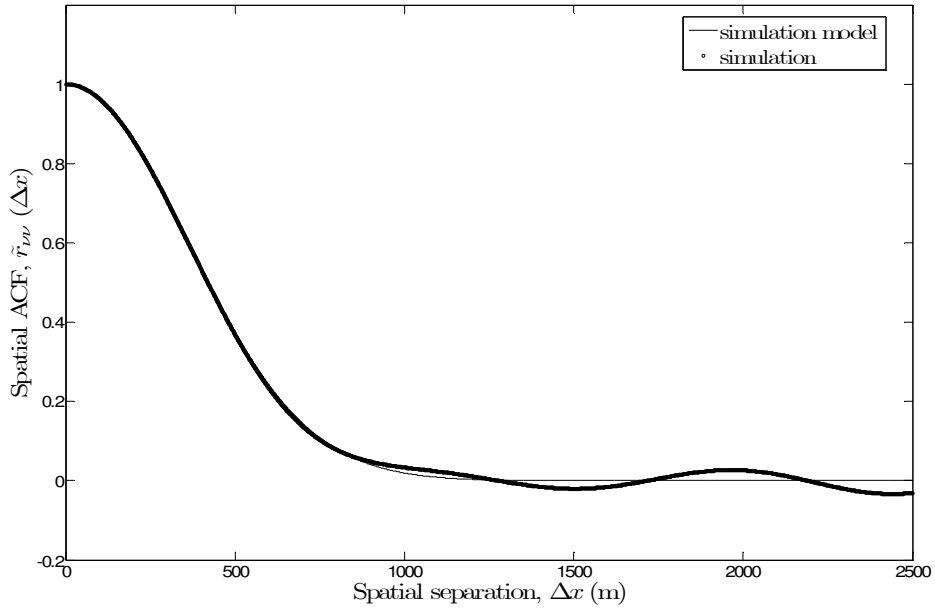

Figure 3-6: ACFs of the spatial shadowing process using N=20, MMEA method, Gaussian computation model for Suburban environment

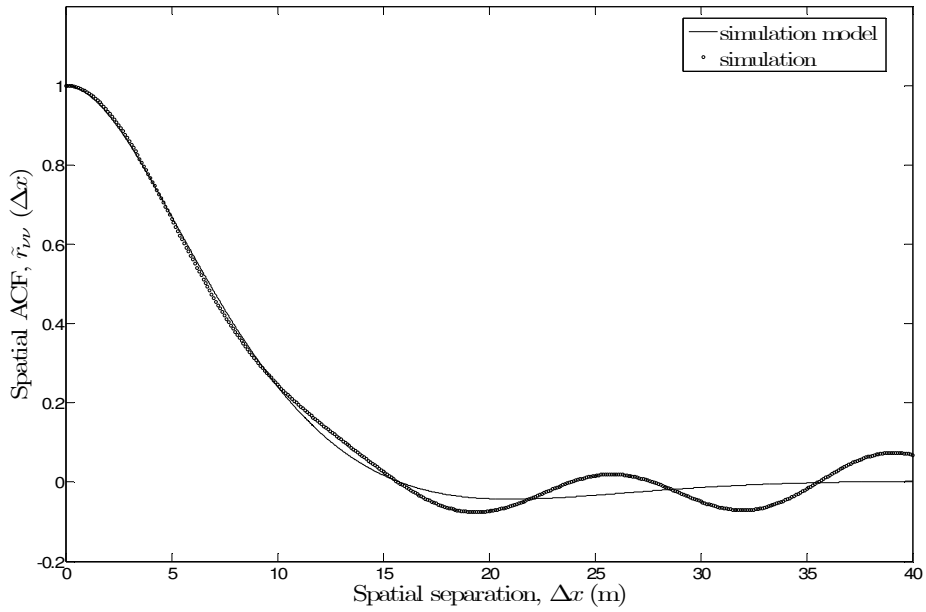

Figure 3-7: ACFs of the spatial shadowing process using N=20, MMEA method, Butterworth computation model for Urban environment

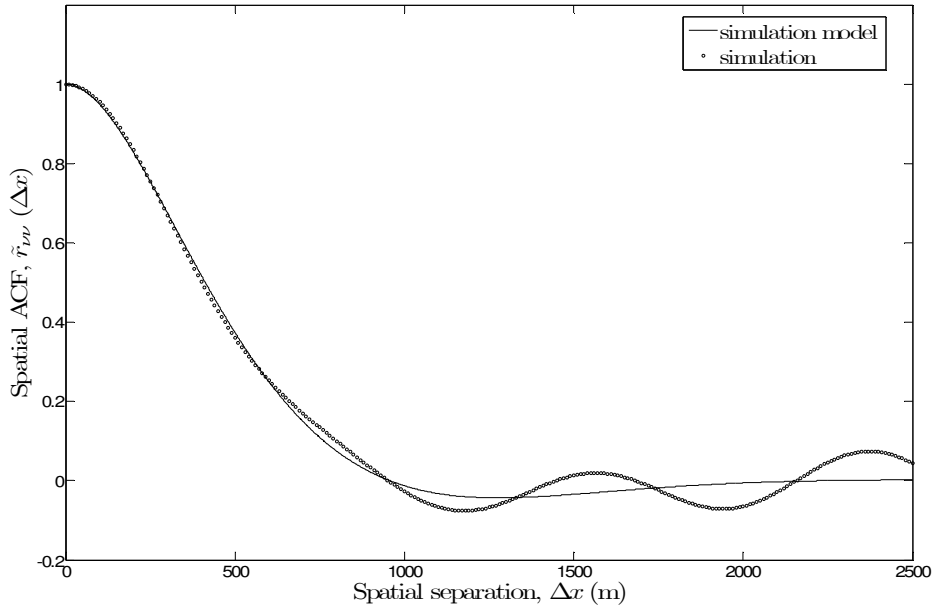

Figure 3-8: ACFs of the spatial shadowing process using N=20, MMEA method, Gaussian computation model for Suburban environment

We conclude that the spatial shadowing process simulator under certain conditions runs efficiently by using some closed models (Gaudmundson, Gauss, Butterworth). According to figure 3.5, it turned out that the Gaussian model of the shadowing process showed the best performance. On the contrary, Gudmundson model is the worst one since, as is shown, some small fluctuations exist on reference model towards simulation model. all relative programs for spatial shadowing simulator are listed in appendix 2.

3.2 MIMO FADING CHANNEL SIMULATORS

MIMO systems offered a significant improvement of quality of the signal at the receiver side. It affects the channel gains that will be increased. MIMO techniques have been invented to overcome some drawbacks on SISO systems, such as the strength of channel gains. As a consequence, many MIMO channel models have been proposed. The figure below shows the fundamental concept of MIMO systems.

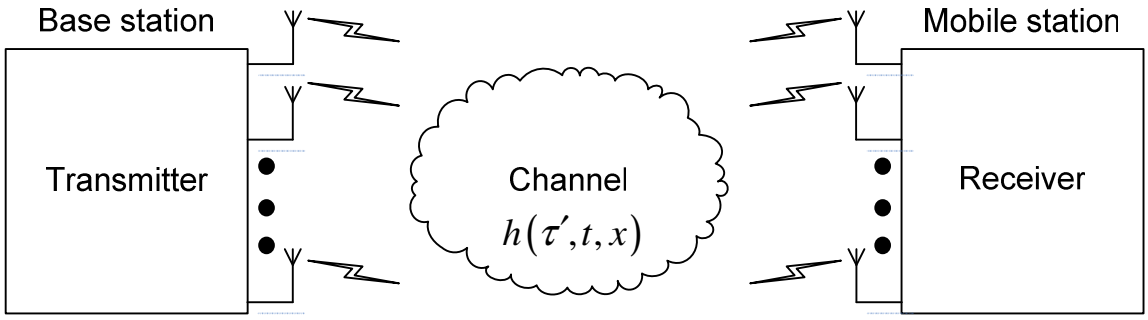

Figure 3-9: example of MIMO system based on smart antennas

The principle of the MIMO system holds onto the fact that we have multiple inputs as well as multiple outputs. The above example is the most commonly used in the mobile fading channel in order to figure out the appropriate model for MIMO channels.
In this section, we first show the one ring model and its main features. Secondly, we discuss the tow ring model. At last, we describe the elliptical model.

3.2.1 MIMO Channel Simulator Based on One-Ring Model

This model is based on the geometrical one-ring scattering model. In [10], [11] and [12] a detailed description of this model holds. The main idea is to assume that all scatterers are located on the same ring around the Mobile station. In this section, we will step by some features of one ring briefly.

3.2.1.1 Geometrical One-Ring Scattering Model

The geometrical model (one ring model) for the MIMO channel employs multi-element antenna arrays. The numbers of antennas elements on both sides vary from at least 2. The figure 2.6 contains a detailed description of all components of the model.

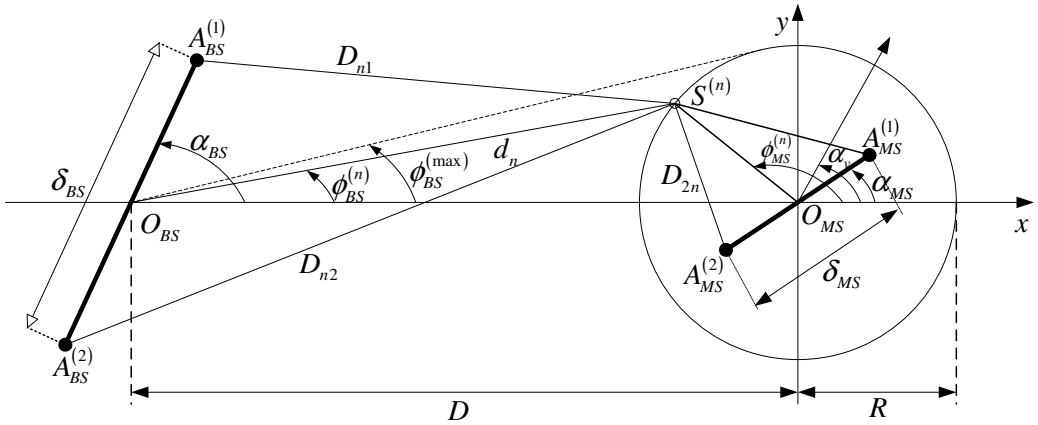

Figure 3-10: Geometrical model (one-ring model)

where:
δ_{BS} : Antenna element spacing at base station
δ_{MS} : Antenna element spacing at mobile station
α_{BS} : Multielement antenna tilt angle at base station
α_{MS} : Multielement antenna tilt angle at mobile station
α_v : Angle of motion
$\phi_{BS}^{(n)}$: Angle of departure at base station
$\phi_{MS}^{(n)}$: Angle of arrival at mobile station
$\phi_{BS}^{(\max)}$: One half of the maximum angle of departures seen at the base station
D : The distance between the base station and the mobile station
R : Radii of the ring of scatterers around the mobile station

NB: In the next section we will go through some features of this model without proof since our focus is the implementation of simulators.

3.2.1.2 Reference Model

As an example of a one-ring based MIMO channel, we take a reference model for $M_{BS} \times M_{MS}$ channel with local scatterers laying on a ring around the mobile station. Later on, we consider the Time ACF and Space-Time CCF only for 2×2 antenna elements and different scattering scenarios.
The channel gains matrix is given by:

$$H_{pq}(t) = \begin{pmatrix} h_{p1}(t) & \dots & h_{1q}(t) \\ \vdots & \ddots & \vdots \\ h_{p1}(t) & \cdots & h_{pq}(t) \end{pmatrix} \tag{3.12}$$

The matrix elements are the channel gains for every antenna element at the base station towards every antenna element at the mobile station. By consequence, the diffuse component of the $A_p^{BS} - A_q^{MS}$ link is approximated by:

$$h_{pq}(t) = \lim_{N\to\infty} \frac{1}{\sqrt{N}} \sum_{n=1}^{N} a_{n,q} b_{n,p} e^{j(2\pi f_n t + \theta_n)} \tag{3.13}$$

When implementing the model, we consider the number of scatterers N finite. Therefore, the equation (1.13) will be modified as follows:

$$\hat{h}_{pq}(t) = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} a_{n,p} b_{n,q} e^{j(2\pi f_n t + \theta_n)} \tag{3.14}$$

where

$$a_{n,q} = e^{j\pi(M_{BS}-2q+1)\frac{\delta_{BS}}{\lambda}\left[\cos(\alpha_{BS})+\phi_{\max}^{BS}\sin(\alpha_{BS})\sin\left(\phi_n^{MS}\right)\right]}, \qquad q = 1,\dots,M_{BS} \tag{3.15}$$

$$b_{n,p} = e^{j\pi(M_{MS}-2p+1)\frac{\delta_{MS}}{\lambda}\cos\left(\phi_n^{MS}-\alpha_{MS}\right)}, \qquad p = 1,\dots,M_{MS} \tag{3.16}$$

$$f_n = f_{\max} \cos\left(\phi_n^{MS} - \alpha_v\right) \tag{3.17}$$

The phases θ_n are independent and identically distributed (i.i.d) random variables uniformly distributed over $[0, 2\pi]$. λ denotes the carrier's wave length and $f_{\max}$ stands for the maximum Doppler frequency. The rest of the parameters were mentioned in the previous section.
However, the 2×2 channel gains can be easily concluded from the expression:

$$h_{11}(t) = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} a_n b_n e^{j(2\pi f_n t + \theta_n)} \tag{3.18}$$

by just substituting a_n and b_n respectively by the complex conjugate value a_n^* and b_n^* where $a_n = e^{j\pi \frac{\delta_{BS}}{\lambda}\left[\cos(\alpha_{BS}) + \phi_{\max}^{BS} \sin(\alpha_{BS}) \sin\left(\phi_n^{MS}\right)\right]}$ and $b_n = e^{j\pi \frac{\delta_{MS}}{\lambda} \cos\left(\phi_n^{MS} - \alpha_{MS}\right)}$. Thus, $h_{12}(t)$ and $h_{21}(t)$ are obtained by replacing respectively a_n and b_n by a_n^* and b_n^*. However, $h_{22}(t)$ is determined by substituting both a_n and b_n by a_n^* and b_n^*.
The cross-correlation function is an important performance tester for the MIMO channels simulator. It is defined as the expected value of the channel gains. As a result, we get the following equation for CCF:

$$\begin{aligned} \rho_{11,22}(\delta_{BS}, \delta_{MS}, \tau) &= E\left\{h_{11}(t) h_{22}^*(t+\tau)\right\} \\ &= \lim_{x\to\infty} \frac{1}{N} \sum_{n=1}^{N} a_n^2 b_n^2 e^{-j2\pi f_n \tau} \end{aligned} \tag{3.19}$$

assuming that the discrete AoA ϕ^{MS} is a continuous random variable with $p\left(\phi^{MS}\right)$. However, we consider in our case only Von Mises distribution to cover all possible cases involving uniform and non-uniform scattering distribution. Hence, the expression of space-time CCF is given by:

$$\rho_{11,22}(\delta_{BS}, \delta_{MS}, \tau) = \int_{-\pi}^{\pi} a_n^2(\delta_{BS}) b_n^2(\delta_{MS}) e^{-j2\pi f_n \tau} p\left(\phi^{MS}\right) d\phi^{MS} \tag{3.20}$$

where

$$p\left(\phi^{MS}\right) = \frac{1}{2\pi I_0(\kappa)} e^{\kappa \cos\left(\phi^{MS} - \phi_0^{MS}\right)}, \quad \phi^{MS} \in (0, 2\pi] \tag{3.21}$$

The Time auto-correlation function can be derived from the Space-Time cross-correlation function simply by setting $\delta_{BS} = \delta_{MS} = 0$. Then, the Time ACF is expressed by:

$$\begin{aligned} r_{ij}(\tau) &= E\left\{h_{ij}(t) h_{ij}^*(t+\tau)\right\} = \rho_{11,22}(0,0,\tau) \\ &= \int_{-\pi}^{\pi} e^{-j2\pi f_{\max} \cos\left(\phi^{MS} - \alpha_v\right)\tau} p\left(\phi^{MS}\right) d\phi^{MS} \end{aligned} \tag{3.22}$$

3.2.1.3 Simulation Model

In normal way, the simulation model is deducted from the reference model. Since this latter can not be realizable due to the infinite number of scatterers, we pick an acceptable number of scatterers to implement the aimed model. The expression of the simulation model is shown here. First, the diffuse component of the $A_1^{BS} - A_1^{MS}$ link is modelled as:

$$\hat{h}_{11}(t) = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} a_n b_n e^{j(2\pi f_n t + \theta_n)} \tag{3.23}$$

The phases θ_n are random variables uniformly distributed over $(0, 2\pi]$. The other channel gains are determined by substituting a_n and b_n respectively by the complex conjugate value a_n^* and b_n^* as described in the section 2.1.2.

The Space-Time cross-correlation function of the simulation is expressed by the equation below:

$$\begin{aligned} \tilde{\rho}_{11,22}(\delta_{BS}, \delta_{MS}, \tau) &= \left\langle \tilde{h}_{11}(t) \tilde{h}_{22}^*(t+\tau) \right\rangle \\ &= \frac{1}{N} \sum_{n=1}^{N} a_n^2(\delta_{BS}) b_n^2(\delta_{MS}) e^{-j2\pi f_n \tau} \end{aligned} \tag{3.24}$$

Similarly to the reference model, the Time auto-correlation function can be obtained by replacing $\delta_{BS} = \delta_{MS} = 0$ in Space-Time cross-correlation function. Thus, we obtain:

$$\begin{aligned} \tilde{r}_{h_{ij}}(\tau) &= \left\langle \tilde{h}_{ij}(t) \tilde{h}_{ij}^*(t+\tau) \right\rangle \\ &= \frac{1}{N} \sum_{n=1}^{N} e^{-j2\pi f_{\max} \cos\left(\phi_n^{MS} - \alpha_v\right)\tau} \end{aligned} \tag{3.25}$$

In our programs, we choose two well-know methods which are described in more detail in the previous chapter. These methods are the modified method of equal areas MMEA and L_p-norm method LPNM. The reasons for considering these two methods are their efficiency, as experienced in many papers and publications.

The idea behind the modified method of equal areas MMEA originated from the method of equal area is described in [6]; it holds onto the fact that we rotate the constellation of the scatterers location on the ring by a defined shift. Hence, the angles of arrivals can be expressed according to the sequel equation which depends on the probability distribution function of different scatterers:

$$\int_{-\infty}^{\tilde{\alpha}_n} p_\alpha(\alpha)\, d\alpha - \frac{1}{N}\left(n - \frac{1}{4}\right) = 0, \; n = 1, \ldots, N \tag{3.26}$$

where N refers to the number of scatterers and $p_\alpha(\alpha)$ is the Von Mises distribution shown in equation (1.20).

For the LPNM method, to compute the AoA in our case, we optimized the tow error function related respectively to the Time auto-correlation function and the Space-Time cross-correlation function. Therefore, we minimized the following two L_p-norms:

$$E_1^{(p)} = \left\{ \frac{1}{\tau_{\max}} \int_0^{\tau_{\max}} \left| r_{h_{11}}(\tau) - \tilde{r}_{h_{11}}(\tau) \right|^p d\tau \right\}^{1/p} \tag{3.27}$$

$$E_2^{(p)} = \left\{ \frac{1}{\delta_{\max}^{BS} \delta_{\max}^{MS}} \int_0^{\delta_{\max}^{BS}} \int_0^{\delta_{\max}^{MS}} \left| \rho(\delta_{BS}, \delta_{MS}) - \tilde{\rho}(\delta_{BS}, \delta_{MS}) \right|^p d\delta_{MS} d\delta_{BS} \right\}^{1/p} \tag{3.28}$$

We found out that when we implemented these tow functions, which we called weighted function $E^{(p)} = w_1 E_1^{(p)} + w_2 E_2^{(p)}$ $\left(w_1, w_2 \in [0,1]\right)$, we didn't get a satisfactory solution that leads to a good fitting between the reference model and the simulation. However, a simple optimization of the second error function is enough and more efficient. We take as starting points the values of the AoA found by the MMEA method.

3.2.1.4 An Extension to the Impulse Response of Frequency-Selective MIMO Channels

The channel gains can be extended to the impulse response of the frequency-selective $M_{BS} \times M_{MS}$ MIMO channel model. The expression is given by:

$$\tilde{h}_{pq}(\tau', t) = \sum_{\ell=1}^{L} \frac{c_\ell}{\sqrt{N_\ell}} \sum_{n=1}^{N_\ell} a_{n,q,\ell} b_{n,p,\ell} e^{j(2\pi f_{n,\ell} t + \theta_{n,\ell})} \delta(\tau' - \tau'_\ell) \tag{3.29}$$

where c_ℓ and τ'_ℓ represent respectively the discrete power delays profile and the discrete propagation delays.

3.2.1.5 Implementation

Our program approach follows some steps in order to implement the simulator in an efficient and easy way. Therefore, we proceed as follows:

- First, we determine the angles of arrivals using two fundamental methods (MMEA, which is newly published in [6]). The autocorrelation function of both the reference model and the simulation model is taken as criteria to evaluate the performance of each computation method.
- In the second step, we compute the channel gains by using the obtained parameters for multiple uncorrelated fading waveforms. To ensure this independency, we compute the parameters using the LPNM method with different values of p.
- The third step consists of computing the extended model for frequency selectivity parameters (channel gains, channel delays) according to the hiperLAN standard measurement.
- Once we have the channel gains as well as delays, we can now simulate the channel simulator.

In addition, some results derived from Matlab programs, which are presented in Appendix 3, are shown here.

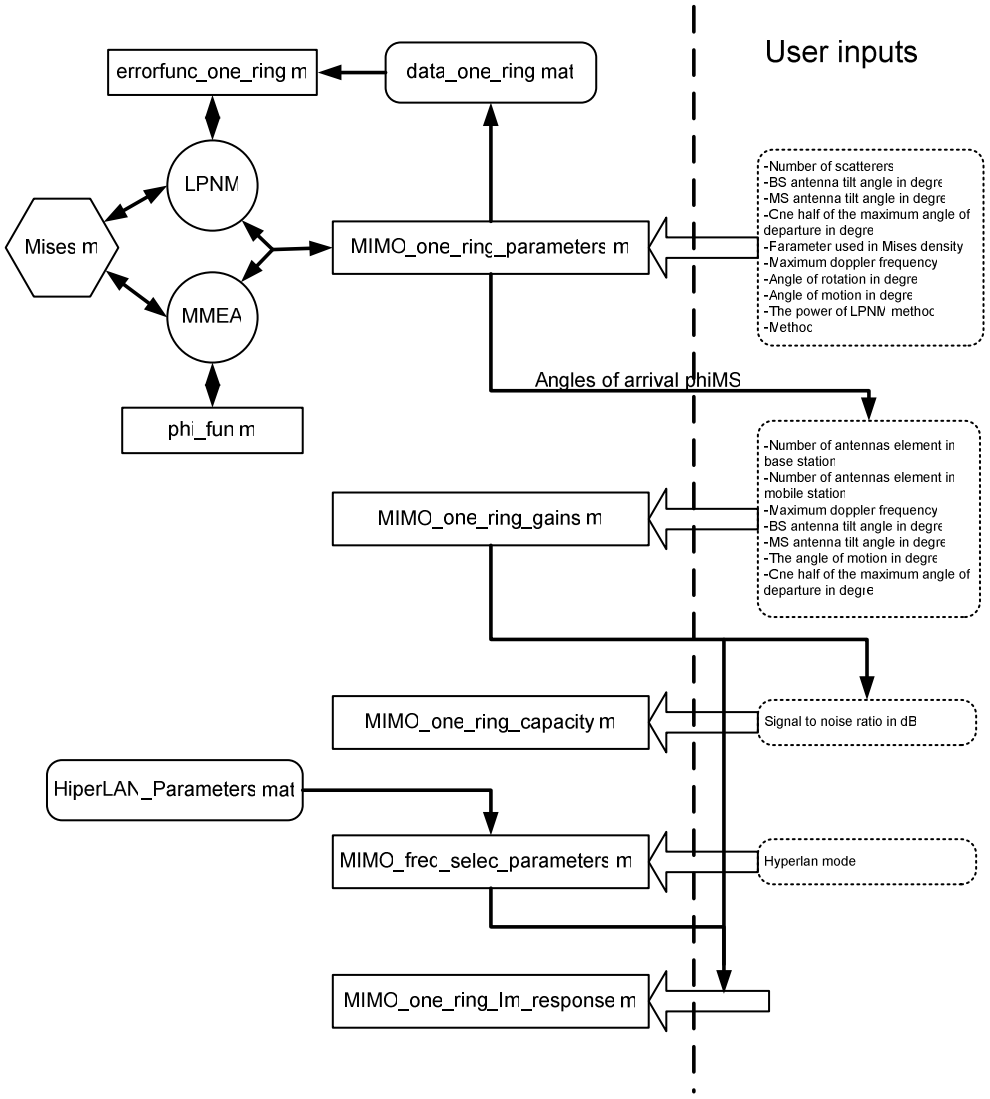

Figure 3-11: The diagram for the implementation of a MIMO channel simulator based on one-ring geometrical model

3.2.1.6 Results and Interpretation

In the following, we illustrate the simulation results for different parameters values coming from the MMEA method and the LPNM method. For each parameter computation method, we depict the ACF and CCF of the simulation and reference model as well as the error that can occur.

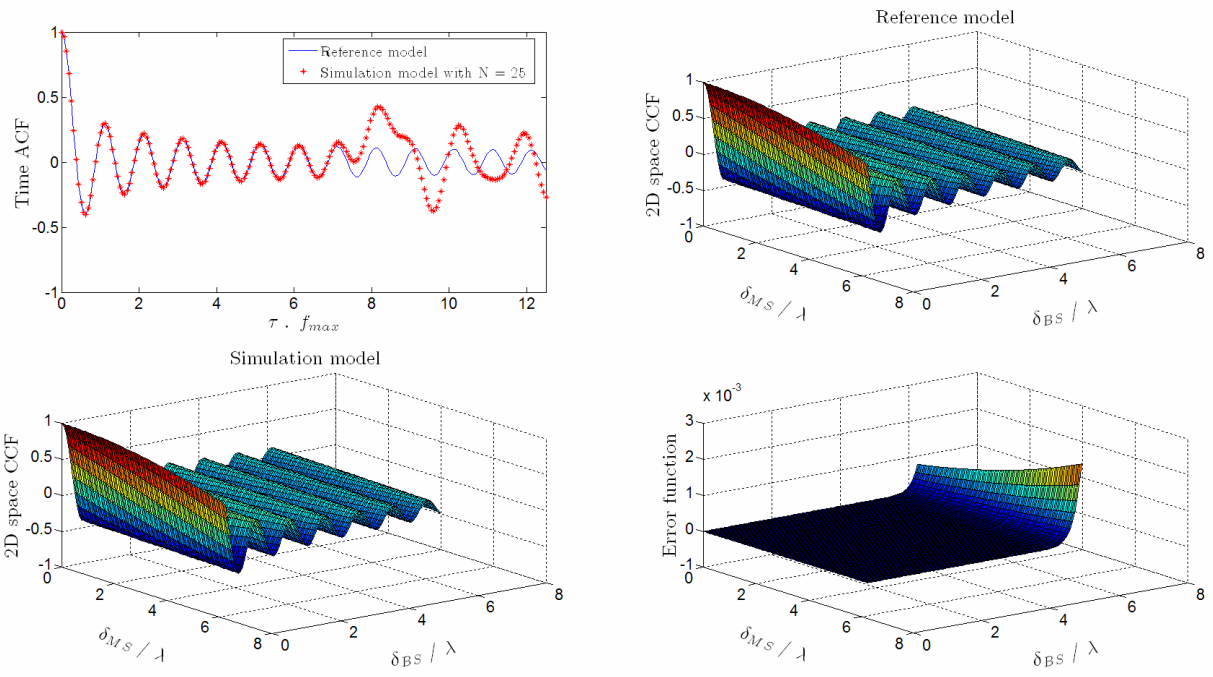

Figure 3-12: Plots for one-ring model under isotropic environment

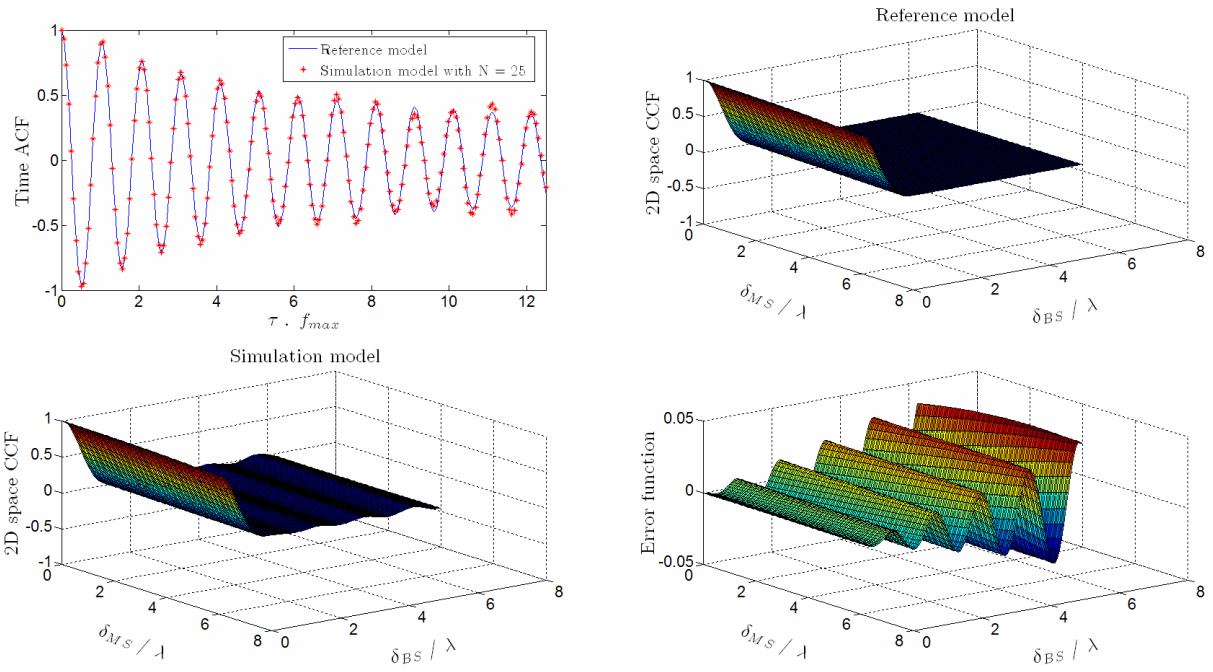

Figure 3-13: Plots for one-ring model under non-isotropic environment

The figures shown above reveal that there is an excellent agreement between the simulation model and the reference model. On the other hand, under isotropic scattering conditions we get a very good fitting until $\delta_{MS}/\lambda = N/4$ and $\delta_{BS}/\lambda = N/4$. Thus, we prove the consistency of the new method to find the best performance. However, under non-isotropic scattering conditions, the simulation model doesn't fit the reference model so much since the

distribution of the scatterers around the ring is non-uniform. To ensure a good closeness, we should increase the number of scatterers N. Hence, an efficient simulator can be achieved.

3.2.2 MIMO Channel Simulator Based on Tow-Ring Model

3.2.2.1 Geometrical Tow-Ring Scattering Model

Now, we take in account the scatterers that could hold on the transmit side. Figure 3.14 demonstrates the geometrical scenario of two-ring principal. Hence, our aim is to find a relation between the AoA and the rest of parameters. In this part, we present briefly the form of MIMO channel two-ring model based. Further details for a more in-depth understanding can be found in [13] and [14].

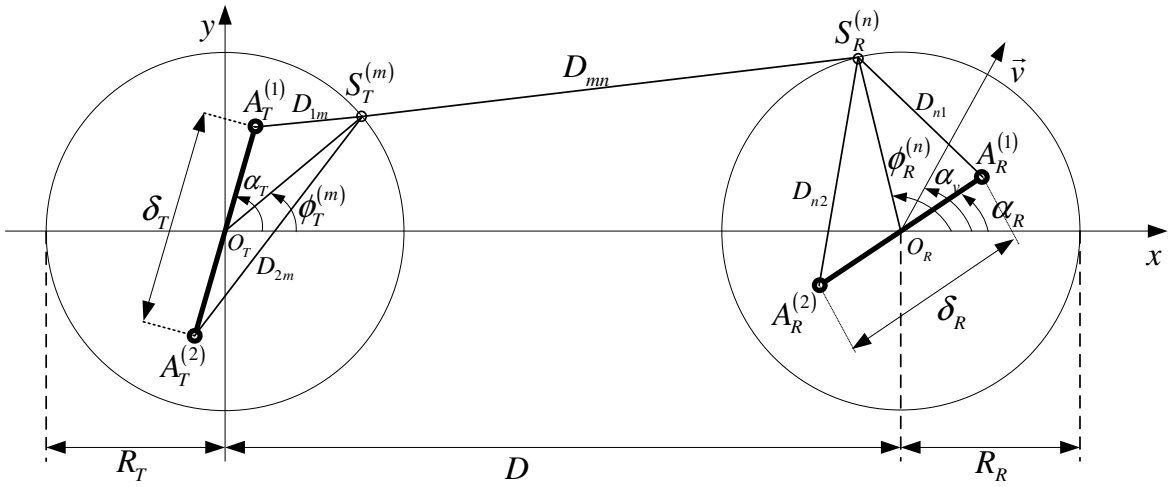

Figure 3-14: Geometrical model (two-ring model)

where

δ_T : Antenna element spacing at base station

δ_R : Antenna element spacing at mobile station

α_T : Multielement antenna tilt angle at base station

α_R : Multielement antenna tilt angle at mobile station

α_v : Angle of motion

$\phi_T^{(n)}$: Angle of departure at base station

$\phi_R^{(n)}$: Angle of arrival at mobile station

D : The distance between the base station and the mobile station

R_T : Radii of the ring of scatterers around the base station

R_R : Radii of the ring of scatterers around the mobile station

3.2.2.2 Reference Model

Starting from the geometrical two-ring model, we assume that the number of scatterers around the receiver and transmitter is infinite. We expand the general expression of the gains presented in reference [15] referring to the parameters of the two-ring model. We then get the expression below for the channel gains.

$$h_{pq}(t) = \lim_{\substack{M\to\infty \\ N\to\infty}} \frac{1}{\sqrt{MN}} \sum_{m,n=1}^{M,N} g_{pqmn} e^{j\left[\left(2\pi\left(f_T^{(m)} + f_R^{(n)}\right)t + \theta_{mn} + \theta_0\right)\right]} \tag{3.30}$$

where

$$a_{mq} = e^{j(M_T - 2q + 1)\pi(\delta_T/\lambda)\cos\left(\phi_T^{(m)} - \beta_T\right)} \tag{3.31}$$

$$b_{np} = e^{j(M_T - 2p + 1)\pi(\delta_R/\lambda)\cos\left(\phi_R^{(m)} - \beta_R\right)} \tag{3.32}$$

$$c_{mn} = e^{j\frac{2\pi}{\lambda}\left(R_T\cos\phi_T^{(m)} - R_R\cos\phi_R^{(n)}\right)} \tag{3.33}$$

$$g_{pqmn} = a_{mq} b_{np} c_{mn} \tag{3.34}$$

$$f_T^{(m)} = f_{T_{\max}} \cos\left(\phi_T^{(m)} - \alpha_T\right) \tag{3.35}$$

$$f_R^{(m)} = f_{R_{\max}} \cos\left(\phi_R^{(m)} - \alpha_R\right) \tag{3.36}$$

$$\theta_{mn} = \left(\theta_m + \theta_n\right) \bmod 2\pi \tag{3.37}$$

θ_0 is set to zero and θ_n, θ_m are random variables independent and uniformly distributed over $[0, 2\pi)$. Therefore, θ_{mn} should be uniformly distributed over the same interval. M_T and M_R consist respectively of the transmit and receive antenna elements and $M_T, M_R \geq 2$ must be respected.

Hence, by combining the diffuse components, we form the so-called stochastic channel matrix $H(t)$. In addition, the capacity that characterizes the channel is designated here:

$$C(t) = \log_2\left[\det\left(I_2 + \frac{P_T}{2N_0} H(t) H^H(t)\right)\right] \tag{3.38}$$

where

I_2: is the identity matrix with two rows and two columns.

P_T: is the total transmitted power allocated uniformly to the two antenna elements of the transmitter.

N_0: is the noise power

To evaluate the performance of the model, we make use of the cross-correlation function which is expressed by:

$$\rho_{11,22}(\delta_T, \delta_R, \tau) = \rho_T(\delta_T, \tau) \cdot \rho_R(\delta_R, \tau) \tag{3.39}$$

where

$$\rho_T(\delta_T, \tau) = \int_{-\pi}^{\pi} a^2(\delta_T, \phi_T) e^{-j2\pi f_T(\phi_T)\tau} p_{\phi_T}(\phi_T) d\phi_T \tag{3.40}$$

$$\rho_R(\delta_R,\tau)=\int_{-\pi}^{\pi} b^2(\delta_R,\phi_R)e^{-j2\pi f_R(\phi_R)\tau}p_{\phi R}(\phi_R)d\phi_R \tag{3.41}$$

where

$$a(\delta_T,\phi_T)=e^{j\pi(\delta_T/\lambda)\cos(\phi_T-\beta_T)} \tag{3.42}$$

$$b(\delta_R,\phi_R)=e^{j\pi(\delta_R/\lambda)\cos(\phi_R-\beta_R)} \tag{3.43}$$

$$f_T(\phi_T)=f_{T_{\max}}\cos(\phi_T-\alpha_T) \tag{3.44}$$

$$f_R(\phi_R)=f_{R_{\max}}\cos(\phi_R-\alpha_R) \tag{3.45}$$

The temporal autocorrelation function ACF can easily be obtained by just setting δ_T and δ_R to zero.

$$r_{h_{i\,j}}(\tau)=\rho_T(0,\tau)\cdot\rho_R(0,\tau) \tag{3.46}$$

3.2.2.3 Simulation Model

The simulation model should be as closed as possible to the reference model. Therefore, we fix the number of scatterers to finite number. Then, the diffuse component of the $A_T^{(q)}-A_R^{(p)}$ link can be expressed as

$$h_{pq}(t)=\frac{1}{\sqrt{MN}}\sum_{m,n=1}^{M,N} g_{pqmn}e^{j\left[\left(2\pi\left(f_T^{(m)}+f_R^{(n)}\right)t+\theta_{mn}\right)\right]} \tag{3.47}$$

q and p vary respectively from 1 till M_T and M_R. g_{pqmn}, $f_T^{(m)}$, $f_R^{(n)}$, and θ_{mn} keep the same expressions as in the previous section.
Like for the reference model, we take the cross-correlation function CCF in order to evaluate the performance. The CCF of the simulation model is given by

$$\hat{\rho}_{pq,p'q'}(\delta_T,\delta_R,\tau)=\hat{\rho}_{T_{qq'}}(\delta_T,\tau)\cdot\hat{\rho}_{R_{pp'}}(\delta_R,\tau) \tag{3.48}$$

where

$$\hat{\rho}_{T_{qq'}}(\delta_T,\tau)=\frac{1}{M}\sum_{m=1}^{M} a^2_{m,q-q'+\frac{1}{2}(M_T+1)}(\delta_T)e^{-j2\pi f_T^{(m)}\tau} \tag{3.49}$$

$$\hat{\rho}_{R_{pp'}}(\delta_R,\tau)=\frac{1}{N}\sum_{n=1}^{N} b^2_{n,p-p'+\frac{1}{2}(M_R+1)}(\delta_R)e^{-j2\pi f_R^{(n)}\tau} \tag{3.50}$$

The temporal autocorrelation function can be derived from the CCF, so we obtain the following expression

$$r_{h_{pq}}(\tau)=\hat{\rho}_{T_{qq'}}(0,\tau)\cdot\hat{\rho}_{R_{pp'}}(0,\tau) \tag{3.51}$$

Similarly to one-ring model, we proceed the same way to compute the AoA and AoD using the same parameters computation methods. Thus, these two methods (MMEA, LPNM) were

explained previously. However, in the LPNM method, we are intended to optimize two error functions relative to the transmit and receive sides. We determine the AoA by minimizing

$$E_1^{(p)} = \left\{ \frac{1}{\delta_{T_{\max}} \tau_{T_{\max}}} \int_0^{\delta_{T_{\max}}} \int_0^{\tau_{T_{\max}}} \left| \rho_{T_{qq'}}(\delta_T, \tau) - \hat{\rho}_{T_{qq'}}(\delta_T, \tau) \right|^p d\delta_T d\tau \right\}^{1/p} \tag{3.52}$$

and

$$E_1^{(p)} = \left\{ \frac{1}{\delta_{R_{\max}} \tau_{R_{\max}}} \int_0^{\delta_{R_{\max}}} \int_0^{\tau_{R_{\max}}} \left| \rho_{T_{pp'}}(\delta_R, \tau) - \hat{\rho}_{T_{pp'}}(\delta_R, \tau) \right|^p d\delta_R d\tau \right\}^{1/p} \tag{3.53}$$

In order to generalize our Matlab toolbox, we consider isotropic and non-isotropic scattering by using Von Mises distribution.

3.2.2.4 Implementation

The implementation of the MIMO channel simulator is divided into two parts. One part deals with the computation of the angles of departure at the transmit side, and the other determines the angles of arrivals (the angles of departures are totally independent of the angles of arrivals i.e. each one is computed individually). Then, we combine these two results so that we determine the channels' gains as well as the impulse response which characterizes the behaviour of the model.
The figure 3-15 sketches the way of implementation of our simulator under the Matlab environment. We instruct in a simple way the different steps on who we run the simulator:

- The functions gen_phiT_two_ring.m and gen_phiR_two_ring.m are used to compute the angles of departure and arrivals. The user has simply to pick the right parameters.
- The channel gains are determined by running MIMO_two_gains.m. To make sure that the angles are different in order to de-correlate the channels, we use the LPNM method with different values of p.
- The option PLOT is provided to sketch the performance criteria, such as temporal ACF and space-time CCF.

NB: An extension to the frequency selectivity could be the subject of future work.

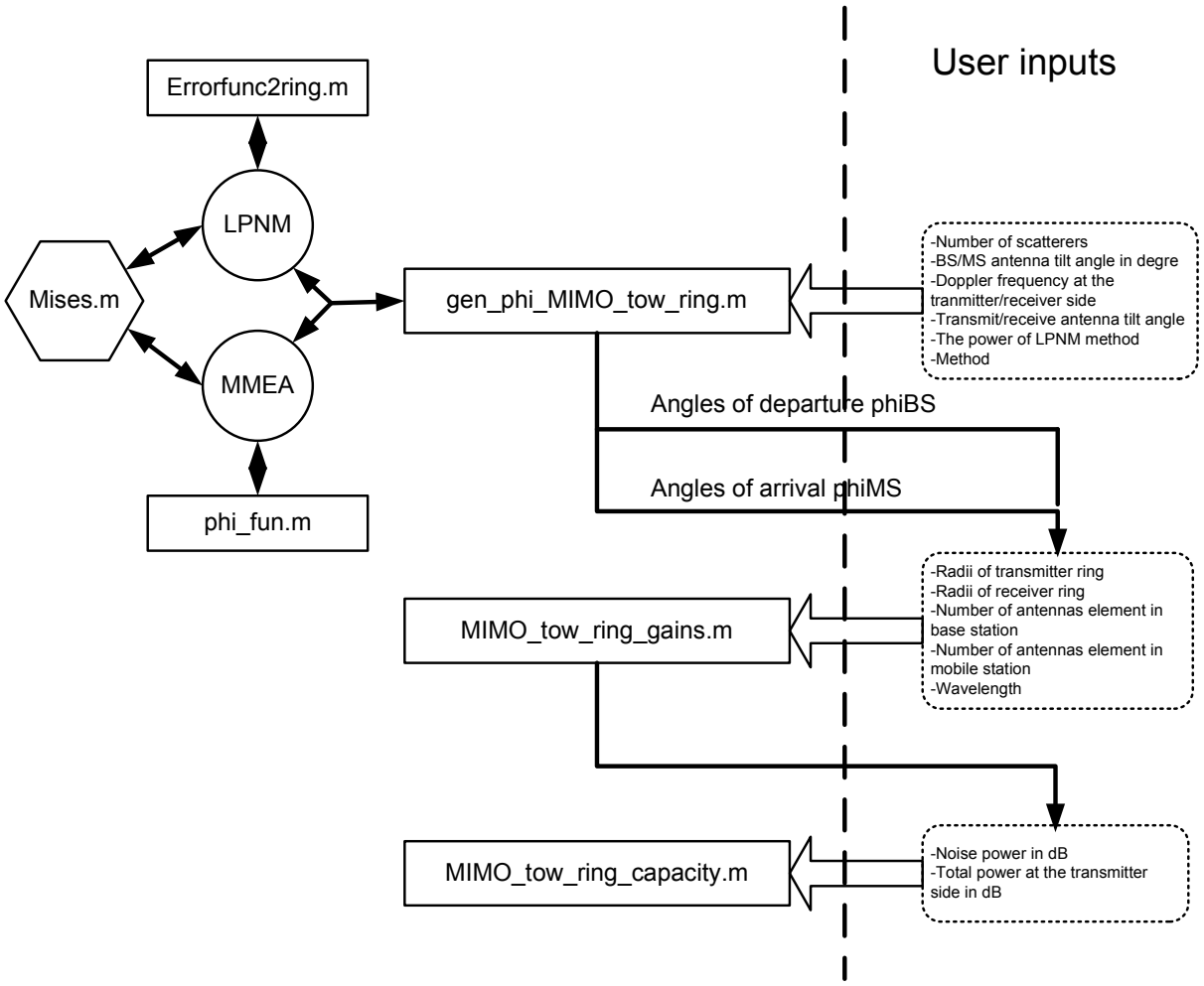

Figure 3-15: The diagram for the implementation of a MIMO channel simulator based on tow-ring geometrical model

3.2.2.5 Results and Interpretation

Following the diagram shown above, we expound the simulation results for different parameter values coming from the MMEA method and the LPNM method similarly as one-ring model. For each parameter computation method, we draw the ACF and CCF of the simulation and reference model as well as the error that can occur. The first set of figures are for the computation of the angels of arrivals for the transmit side with $\alpha_T = 90^\circ$, $\beta_T = 180^\circ$, $\kappa = 0 \, and \, 10$ and $f_{T_{\max}} = 91 Hz$.

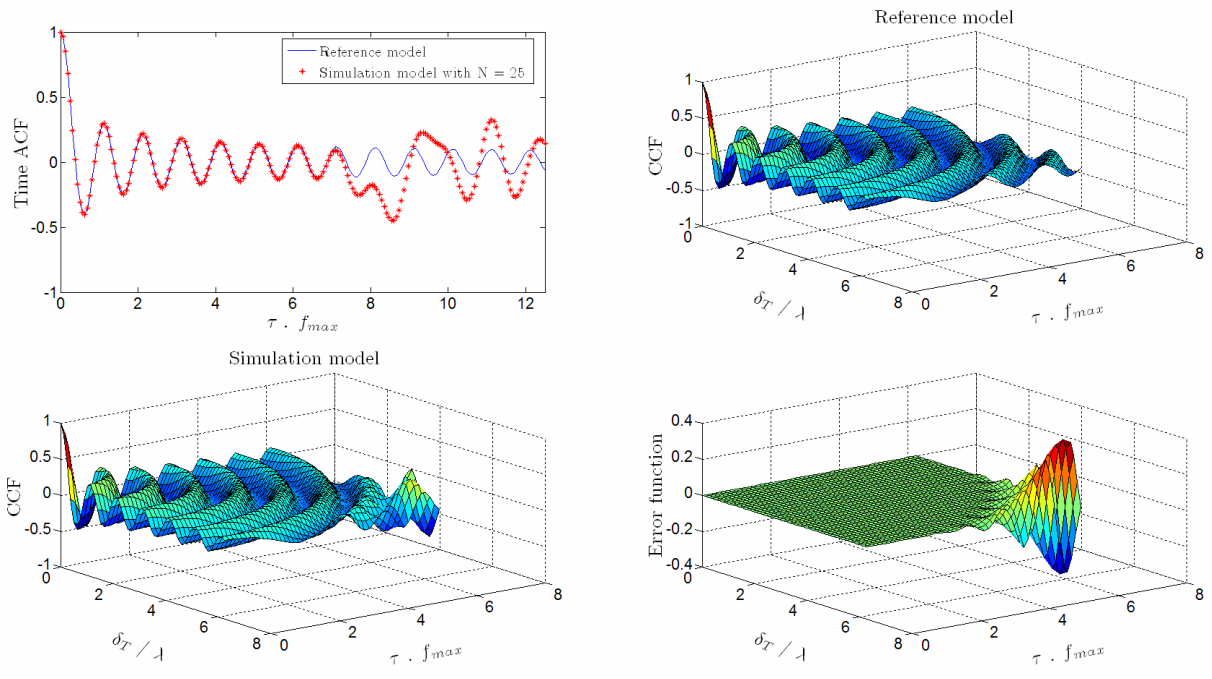

Figure 3-16: Plots for two-ring model under isotropic environment (on transmitter)

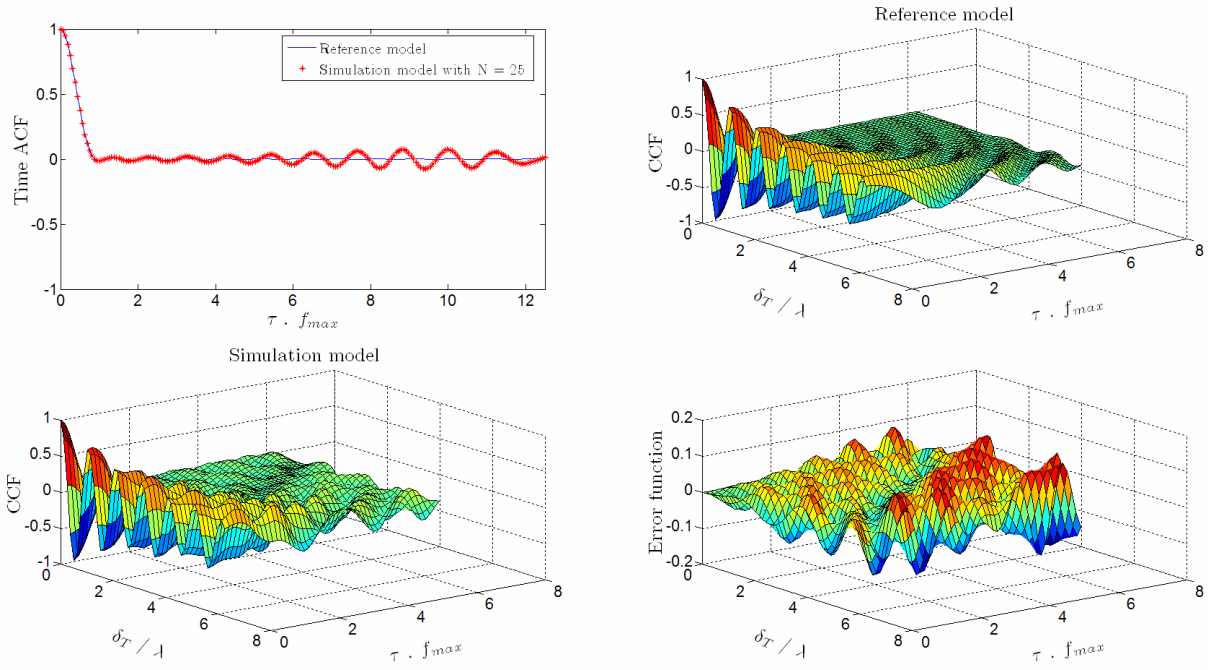

Figure 3-17: Plots for two-ring model under non-isotropic environment (on transmitter)

The second set of figures are for the computation of the angels of departures for the receive side with $\alpha_R = 90^\circ$, $\beta_R = 180^\circ$, $\kappa = 0 \, and \, 10$ and $f_{R_{\max}} = 91 Hz$.

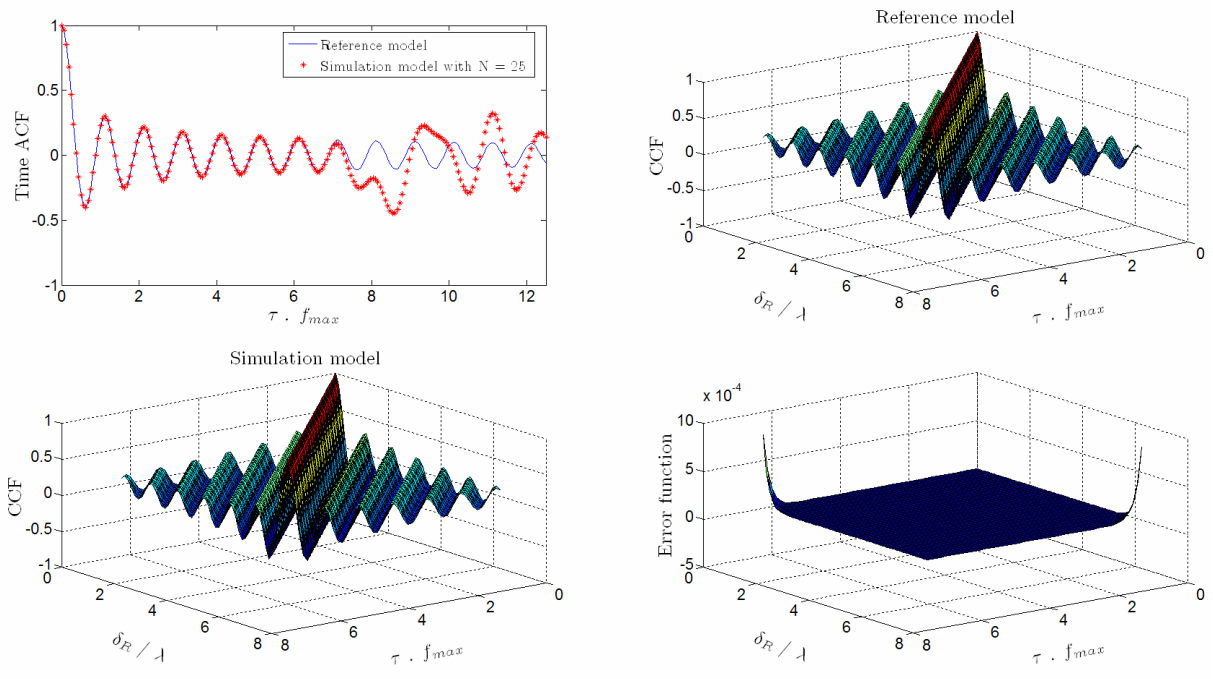

Figure 3-18: Plots for two-ring model under isotropic environment (on receiver)

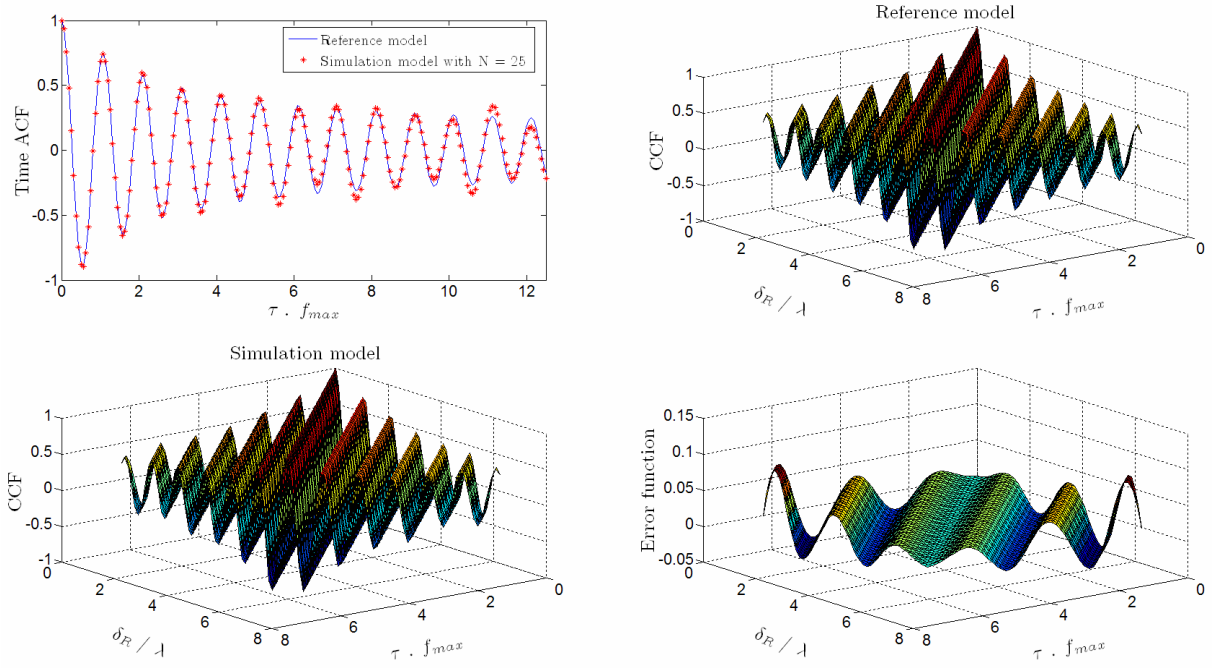

Figure 3-19: Plots for two-ring model under non-isotropic environment (on receiver)

During the simulation of several examples, we remarked that the angles β and α have a big impact on the results as mentioned above in the figures. We find out after several tests that the computation of the angles influences the results due to the variations caused by the original MEDS method. Therefore, further research work will be investigated on this topic.
A perfect fitting has been achieved between the simulation model and the reference model due to the high performance approved by the MMEA method. This method allows us to reach

a shape overlapping until $N/4$. Once again, the insurance of getting a good fitting leads to an efficient simulator since these parameters are the key elements in the model. All necessary programs for the generation of the MIMO channel simulator based on the two ring model can be found in Appendix 3.

3.2.3 MIMO Channel Simulator Based on Elliptical Model

3.2.3.1 Geometrical Elliptical Scattering Model

For the elliptical model for MIMO channels, we assume that all scatterers are located on an ellipse where the transmitter and the receiver hold onto the focal points. The figure below describes the principle idea behind this model. More information about this concept can be found in [18] and [19].

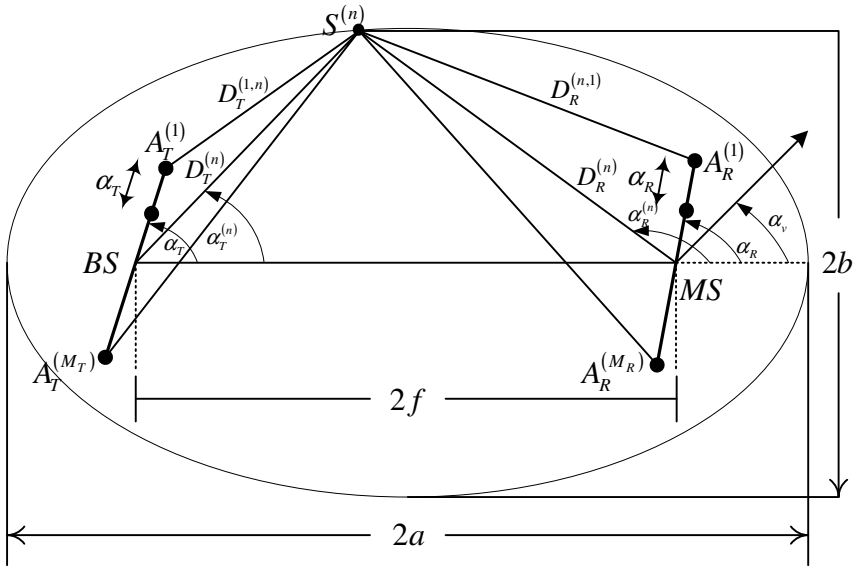

Figure 3-20: Geometrical model (elliptical model)

where

δ_T : Antenna element spacing at base station

δ_R : Antenna element spacing at mobile station

α_T : Multielement antenna tilt angle at base station

α_R : Multielement antenna tilt angle at mobile station

α_v : Angle of motion

$\phi_T^{(n)}$: Angle of departure at base station

$\phi_R^{(n)}$: Angle of arrival at mobile station

M_T : Number of antenna elements at base station

M_R : Number of antenna elements at mobile station

3.2.3.2 Reference Model

As shown in [18], the channel gains of MIMO based on elliptical model is given by the following expression

$$g_{kl}(t) = \lim_{N\to\infty} \frac{1}{\sqrt{N}} \sum_{n=1}^{N} a_{1n} b_{kn} e^{j(2\pi f_n t + \theta_n + \theta_0)} \tag{3.54}$$

where

$$a_{ln} = e^{j\pi(M_T - 2l + 1)(\delta_T/\lambda)\cos\left(\phi_T^{(n)} - \alpha_T\right)} \tag{3.55}$$

$$b_{kn} = e^{j\pi(M_R - 2k + 1)(\delta_R/\lambda)\cos\left(\phi_R^{(n)} - \alpha_R\right)} \tag{3.56}$$

$$f_n = f_{\max}\cos\left(\phi_R^{(n)} - \alpha_v\right) \tag{3.57}$$

In contrast to the one-ring model, the AoD in the elliptical model depends on the behaviour of the AoA. According to the geometrical model and after some demonstration steps, the AoD can be obtained by

$$\phi_T^{(n)} = \begin{cases} f\left(\phi_R^{(n)}\right) & \text{if} \quad -\pi < \phi_R^{(n)} \le -\phi_0 \\ f\left(\phi_R^{(n)}\right) - \pi & \text{if} \quad -\phi_0 < \phi_R^{(n)} \le 0 \\ f\left(\phi_R^{(n)}\right) + \pi & \text{if} \quad 0 < \phi_R^{(n)} \le \phi_0 \\ f\left(\phi_R^{(n)}\right) & \text{if} \quad \phi_0 < \phi_R^{(n)} \le \pi \end{cases} \tag{3.58}$$

where

$$f\left(\phi_R^{(n)}\right) = \arctan\left[\frac{(\kappa_0^2 - 1)\sin\left(\phi_R^{(n)}\right)}{2\kappa_0 + \left(\kappa_0^2 + 1\right)\cos\left(\phi_R^{(n)}\right)}\right] \tag{3.59}$$

and

$$\phi_0 = \arctan\left(\frac{\kappa_0^2 - 1}{2\kappa_0}\right) \tag{3.60}$$

The parameter κ_0 designates the reciprocal value of the eccentricity e of the ellipse, i.e. $\kappa_0 = 1/e = a/f$.

We consider the probability $p_{\phi_R}(\phi_R)$, which defines the scattering type, as the Von Mises distribution given in (3.21). Then, the 3D space-time CCF of the reference model demonstrated in [18] is written as

$$\rho_{kl,k'l'}(\delta_T, \delta_R, \tau) = \int_{-\pi}^{\pi} c_{ll'}(\delta_T, \phi_T)\, d_{kk'}(\delta_R, \phi_R)\, e^{-j2\pi f(\phi_R)\tau} p_{\phi_R}(\phi_R)\, d\phi_R \tag{3.61}$$

where

$$c_{ll'}(\delta_T, \phi_T) = e^{-j2\pi(l-l')(\delta_T/\lambda)\cos(\phi_T - \alpha_T)} \tag{3.62}$$

$$d_{kk'}\left(\delta_R,\phi_R\right)=e^{-j2\pi(k-k')(\delta_R/\lambda)\cos(\phi_R-\alpha_R)} \tag{3.63}$$

$$f\left(\phi_R\right)=f_{\max}\cos\left(\phi_R-\alpha_v\right) \tag{3.64}$$

The temporal ACF can be derived as for the two-ring model from the 3D space-time CCF by setting δ_T and δ_R to zero.

$$r_{g_{kl}}\left(\tau\right)=\int_{-\pi}^{\pi}e^{-j2\pi f_{\max}\cos(\phi_R-\alpha_v)\tau}p_{\phi_R}\left(\phi_R\right)d\phi_R \tag{3.65}$$

3.2.3.3 Simulation Model

The simulation model is derived for the reference model by just reducing the number of scatterers around the transmitter and the receiver to a finite number. Therefore, the expression of the channel gains or the diffuse component of the link from the transmit antenna element $A_T^{(l)}\left(l=1,\ldots,M_T\right)$ to the receive antenna element $A_R^{(l)}\left(l=1,\ldots,M_R\right)$ as mentioned in [19] are

$$\hat{g}_{kl}\left(t\right)=\frac{1}{\sqrt{N}}\sum_{n=1}^{N}a_{\ln}b_{kn}e^{j(2\pi f_n t+\theta_n)} \tag{3.66}$$

The 3D space-time CCF of the simulation model is given by

$$\hat{\rho}_{kl,k'l'}\left(\delta_T,\delta_R,\tau\right)=\frac{1}{N}\sum_{n=1}^{N}c_{ll'}d_{kk'}e^{-j2\pi f(\phi_R)\tau} \tag{3.67}$$

where

$$c_{ll'n}=e^{-j2\pi(l-l')(\delta_T/\lambda)\cos\left(\phi_T^{(n)}-\alpha_T\right)} \tag{3.68}$$

$$d_{kk'n}=e^{-j2\pi(k-k')(\delta_R/\lambda)\cos\left(\phi_R^{(n)}-\alpha_R\right)} \tag{3.69}$$

The temporal ACF of the simulation model is given by

$$r_{g_{kl}}\left(\tau\right)=\frac{1}{N}\sum_{n=1}^{N}e^{-j2\pi f_{\max}\cos(\phi_R^{(n)}-\alpha_v)\tau} \tag{3.70}$$

To compute the AoA used to generate the channel gains, we used two fundamental methods approved by their efficiency as described in the previous sections. The MMEA method is defined by solving (3.26). The LPNM method is supposed to optimize the two following errors.

$$E_1^{(p)}=\left\{\frac{1}{\tau_{\max}}\int_0^{\tau_{\max}}\left|r_{g_{11}}\left(\tau\right)-\tilde{r}_{g_{11}}\left(\tau\right)\right|^p d\tau\right\}^{1/p} \tag{3.71}$$

$$E_2^{(p)}=\left\{\frac{1}{\delta_T^{\max}\delta_R^{\max}}\int_0^{\delta_T^{\max}}\int_0^{\delta_R^{\max}}\left|\rho_{kl,k'l'}\left(\delta_T,\delta_R\right)-\tilde{\rho}_{kl,k'l'}\left(\delta_T,\delta_R\right)\right|^p d\delta_R d\delta_T\right\}^{1/p} \tag{3.72}$$

The elliptical model can be extended to a frequency-selective MIMO channel. Hence, the channels gains ought to be

$$\tilde{h}_{kl}(\tau',t)=\sum_{\ell=0}^{L-1}\sum_{c=1}^{C}\tilde{a}_{\ell}d_{c}\tilde{g}_{kl,c}(t)\delta(\tau'-\tilde{\tau}'_{\ell}) \tag{3.73}$$

where C designates the number of channels gains that should be generated with different AoA. However, to obey this condition, the LPNM method can be used for different values of p. d_c is a positive real value that fulfils the condition $\sum_{c=1}^{C} d_c^2 = 1$.

3.2.3.4 Implementation

The implementation of the MIMO channel simulator based on the elliptical model is similar to the one-ring model, unless we assume that in the one-ring model the angles of departure are predefined and constant. However, for the elliptical model these angles are related to the angles of departures as demonstrated in (3.58).
The figure 3.21 illustrates the sketch of our simulator under the Matlab environment. Here follows a simple guidance on how the user can simulate the elliptical model:

- To compute the angles of arrival, the user has to run the program called MIMO_elliptical_parameters.m. This program is related to gen_phiT.m which generates the angles of departure starting from the angles of arrival. When picking the LPNM method, it may take a few minutes.
- The channel gains are determined by running MIMO_elliptical_gains.m. To make sure that the angles are different in order to de-correlate the channels, we use the LPNM method with different values of p.
- The option PLOT is provided to sketch the performance criteria like temporal ACF and space-time CCF.
- The frequency-selective parameters (gains and delays) are specified by the standards (HiperLAN-A, HiperLAN-B, HiperLAN-C, HiperLAN-D, HiperLAN-E) shown in the Appendix 2.
- Once we get the channel gains and the frequency-selective parameters, we can see the impulse response of the channel by running MIMO_Im_response.m.

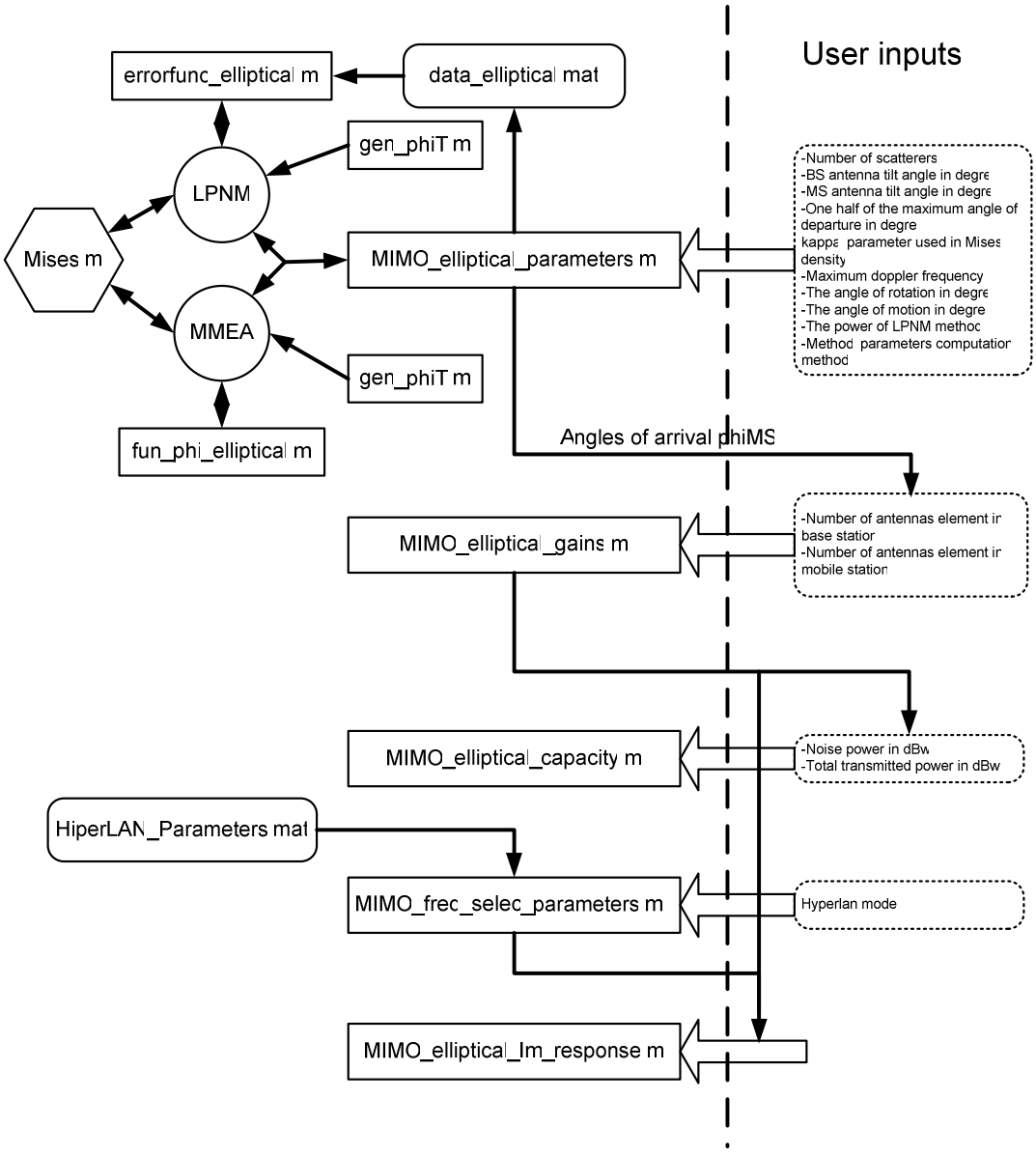

Figure 3-21: The diagram for the implementation of a MIMO channel simulator based on elliptical geometrical model

3.2.3.5 Results and Interpretation

Here, we show some results for the simulation of the elliptical simulator. Moreover, we take into account an isotropic and non-isotropic scattering environment.

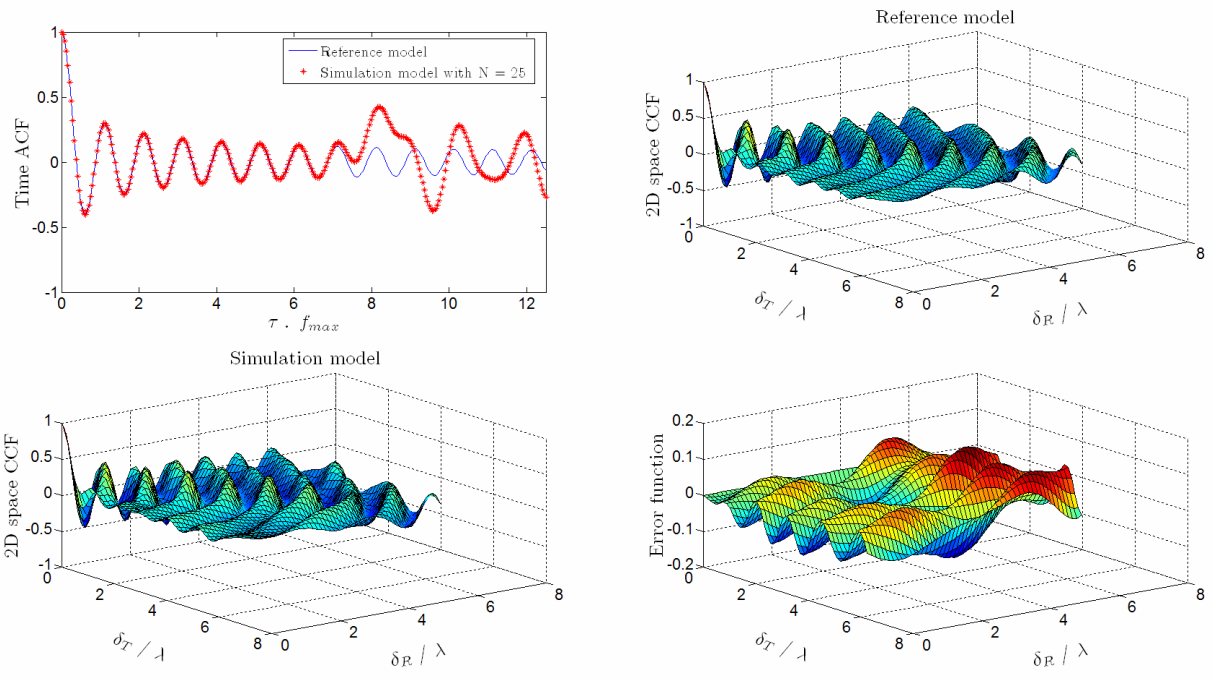

Figure 3-22: Plots for elliptical model under isotropic environment

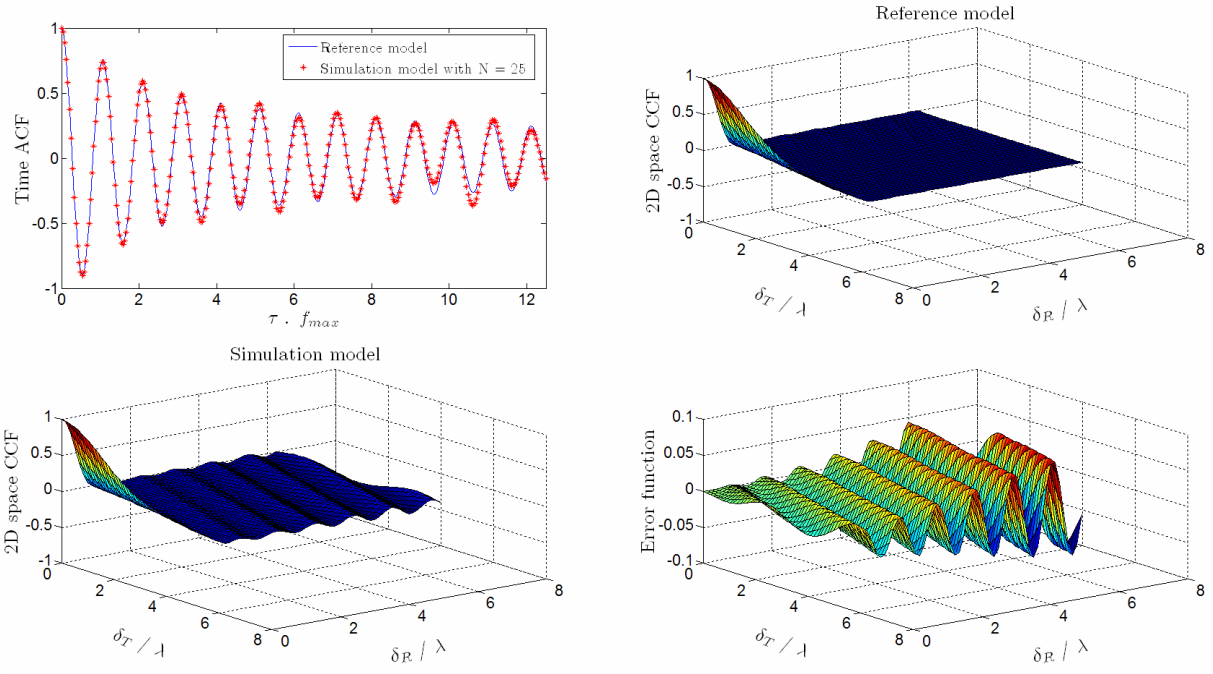

Figure 3-23: Plots for elliptical model under non-isotropic environment

Due the fact that the angles of departure are dependant on the angles of arrival, we couldn't achieve a perfect fitting between the simulation model and the reference model. In addition, the MEDS method supposes that the environment should be isotropic. In other words, the scatterers must be located on the ring around the mobile station in a uniform fashion. Therefore, as depicted in the figures above, the best fitting we get is on the receiver side.

3.3 CONCLUSION

In this chapter, different Mobile fading channels have been treated. First, we briefly described the spatial shadowing process and its allied characteristics. A new computation method has been proposed. It is called Modified MEA, and presents a high efficiency together with the well-known LPNM method. We then proposed an optimized way for the implementation of the channel simulator. By plotting the temporal ACF and 3D space-time CCF, we evaluated the degree of fitness of the simulation model towards the reference model. Secondly, we introduced the MIMO channels (one-ring geometrical model, two-ring geometrical model and elliptical geometrical model). For each model, we presented shortly the fundamental concept and the main features (channel gains, temporal ACF, 3D space-time CCF, impulse response, etc.). Once again, to evaluate the performance of the simulator, we made use of the stochastic properties mentioned earlier. The next chapter will deal with the discussion and a brief conclusion recapitulating our work. Finally, we open some perspectives for further work in this domain, counted as an extension of the present work.

CHAPTER 4 - CONCLUSION AND PERSPECTIVES

4.1 DISCUSSION AND CONCLUSION

The aim of this thesis was to develop a new toolbox for Matlab software involving the mobile fading channels and to provide some useful functions for testing the performance. Therefore, we created a new package from scratch that can be integrated into the new release of Matlab software.

Based on researching work carried out by masters and ph.d student working under the supervision of prof. Matthias Pätzold, we proposed an optimized solution for the implementation of nearly all mobile fading channel simulators. The toolbox consists of a user/friendly interface that can be invoked by the user. The user has to simply input (?) the right parameters.

We started by looking at some articles related to channel fading domains; we then tried to pinpoint the main formula and concepts that lead us to achieving our solution. It is very important for us to define a toolbox that should be as simple as possible and suitable for channel fading

To make our toolbox consistent and adaptable to the Matlab environment, we first divided each channel simulator into several subfunctions. Each function performs an operation. Some functions deal with channel parameters computation and others ensure the correctness of channel models.

In the first part of our work, we implemented all parameter computation methods for both frequency-nonselective and frequency-selective channels using different methods (MED, MEA, MSEM, MCM, MEDS, MEDS-sp, LPNM,…). We pointed out the main difference between these methods and their performance by plotting the ACF of the simulation model in regards to the reference model as well as the FCF.

The second part of this thesis put emphasis on the mobile fading channel simulators. We briefly gave an overview on each simulator. The mathematical concepts have been presented. Using MMEA and LPNM methods for computing the channel parameters, we developed a function for each simulator to determine the convenient parameters. In addition, we developed the impulse response of each channel which merely characterizes the concept of the simulator.

New methods for parameter computation have been introduced and they confirmed their efficiency. Therefore, improved results have been reached and could be published in an updated version of MIMO models.

Our toolbox tied simplicity together with higher performance. Hence, in this thesis we accomplished:

- An optimized toolbox
- Simple user/friendly interface
- Improvement of existent models
- Introduction of new methods
- Universal testing functions

4.2 PERSPECTIVES

Further work and issues that could be interesting for further research topics:

- Implementation of a mobile fading channel simulator based on COST 273 (new European measurement for 3G mobile systems).
- Investigation of parameter computation methods for the non-isotropic scattering environment case.
- Development of DSP codes for mobile fading channel simulator issued from proposed models (the spatial shadowing, MIMO, Rayleigh, etc…).
- Development of electronic circuits integrating channel functionalities.
- In this thesis we considered only fixed-to-fixed systems. Further work could take into account the mobility of the two parts (transmitter, receiver).
- Optimized time consumption could be of interest if we migrate the code to some machine programming language like C or FORTRAN.

REFERENCES

[1] M. Pätzold: "**Mobile Fading Channels**," Chichester: John Wiley & Sons, February 2002, 428 pages.

[2] M. Pätzold, A. Szczepanski, N. Youssef: "**Methods for Modelling of Specified and Measured Multipath Power Delay Profiles,**" IEEE Trans. Veh. Technol., vol. VT-51, no. 5, Sept. 2002, pp. 978988.

[3] M. Pätzold, A. Szczepanski: "**Methods for Modeling of Specified and Measured Multipath Power Delay Profiles,**" Proc. IEEE 51st Vehicular Technology Conf., IEEE VTC 2000-Spring, Tokyo, Japan, 15. - 18. May 2000, pp. 18281834.

[4] Yahong R. Zheng and Chengshan Xiao: **"Improved Models for the Generation of Multiple Uncorrelated Rayleigh Fading Waveforms,"** IEEE COMMUNICATIONS LETTERS, VOL. 6, NO. 6, JUNE 2002.

[5] M. Pätzold, B. O. Hogstad, D. Kim, S. Kim: "**A New Design Concept for High-Performance Fading Channel Simulators Using Set Partitioning,**" Proc. 8th International Symposium on Wireless Personal Multimedia Communications, WPMC 2005, Aalborg, Denmark, 18. - 22. September 2005, pp. 496–502.

[6] Carlos A. Gutiérrez-Día-de-León, M. Pätzold: "**Sum-of-sinusoids-based simulation of flat fading wireless propagation channels under non-isotropic scattering conditions**" not published yet.

[7] M M. Pätzold, V. D. Nguyen: "**A Spatial Simulation Model for Shadow Fading Processes in Mobile Radio Channels,**" Proc. 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, IEEE PIMRC 2004, Barcelona, Spain, 05. - 08. Sept. 2004, vol. 3, pp.1832–1838.

[8] M. Pätzold and Y. Kun: "**An Exact Solution for the Level-Crossing Rate of Shadow Fading Processes Modelled by Using the Sum-of-Sinusoids Principle,**" Proc. 9th International Symposium on Wireless Multimedia Communications, WPMC 2006, San Diego, USA, Sept. 2006, pp. 188–193.

[9] M. Pätzold, B. O. Hogstad: "**A Space-Time Channel Simulator for MIMO Channels Based on the Geometrical One-Ring Scattering Model,**" Wireless Communications and Mobile Computing, Special Issue on Multiple-Input Multiple-Output (MIMO) Communications, vol. 4, no. 7, Nov. 2004, pp. 727737.

[10] M. Pätzold, B. O. Hogstad: "**A Space-Time Channel Simulator for MIMO Channels Based on the Geometrical One-Ring Scattering Model,**" Proc. 60th IEEE Semiannual Vehicular Technology Conference, IEEE VTC 2004-Fall, Los Angeles, CA, USA, 26. - 29. Sept. 2004, Vol. 1, pp. 144–149.

[11] M. Pätzold and B. O. Hogstad : "**A Wideband Space-Time MIMO Channel Simulator Based on the Geometrical One-Ring Model,**" Proc. 64th IEEE

Semiannual Vehicular Technology Conference, IEEE VTC 2006-Fall, Montreal, Canada, Sept. 2006.

[12] Haixia Zhang, Dongfeng Yuan, M. Pätzold, Yi Wu and Van Duc Nguyen: "**A novel wideband space-time channel simulator based on the geometrical one-ring model with application in MIMO-OFDM systems**", not published yet.

[13] M. Pätzold, B. O. Hogstad, N. Youssef, D. Kim: "**A MIMO Mobile-to-Mobile Channel Model: Part I - The Reference Model,**" Proc. 16th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2005, Berlin, Germany, 11. - 14. September 2005, vol. 1, pp. 573–578.

[14] B. O. Hogstad, M. Pätzold, N. Youssef, D. Kim: "**A MIMO Mobile-to-Mobile Channel Model: Part II - The Simulation Model,**" Proc. 16th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2005, Berlin, Germany, 11. - 14. September 2005, vol. 1, pp. 562–567.

[15] B. O. Hogstad, M. Pätzold, N. Youssef : "**Modelling , analysis, and simulation of MIMO mobile-to-mobile fading channels**", not published yet.

[16] M. Pätzold, B. O. Hogstad: "**Two New Methods for the Generation of Multiple Uncorrelated Rayleigh Fading Waveforms,**" Proc. 63rd Semiannual Vehicular Technology Conference, IEEE VTC 2006-Spring, Melbourne, Australia, May 2006, vol. 6, pp. 2782–2786.

[17] C.-X. Wang, M. Pätzold: "**Methods of Generating Multiple Uncorrelated Rayleigh Fading Processes,**" Proc. 57th IEEE Vehicular Technology Conference, IEEE VTC 2003 - Spring, Jeju, Korea, 22. - 25. April 2003, pp. 510514.

[18] B. O. Hogstad, M. Pätzold, A. Chopra, D. Kim, K. B. Yeom : "**A Wideband MIMO Channel Simulation Model Based On the Geometrical Elliptical Scattering Model,**" Proc. 15th Wireless World Research Forum Meeting, WWRF15, 8. - 9. December 2005, Paris, France.

[19] M. Pätzold and B. O. Hogstad: "**A Wideband MIMO Channel Model Derived From the Geometric Elliptical Scattering Model,**" Proc. 3rd International Symposium on Wireless Communication System, ISWCS'06, Valencia, Spain, Sept. 2006, pp. 138–143.

[20] "**Matlab documentation,"** version 7.0.1.24704 (R14) September 13, 2004.

[21] Arkadius Szczepanski: "**Modelling of Frequency-Selective Fading Channels by Using Deterministic Gaussion Uncorrelated Scattering Processes**" Diploma thesis, September 1999.

ACRONYMS AND ABBREVIATIONS

ACF	autocorrelation function
ADF	average duration of fades
AoA	angle of arrival
AoD	angle of departure
API	application programming interface
ASK	amplitude shift keying
BU	bad urban
CCF	cross-correlation function
CDF	cumulative distribution function
COST	European Cooperation in the Field of Scientific and Technical Research
DSP	digital signal processing
FSK	frequency shift keying
FCF	frequency correlation function
FORTRAN	earlier programming language
HT	hilly terrain
Matlab	mathematics laboratory
MCM	Monte Carlo method
MED	method of equal distances
MEDS	method of exact Doppler spread
MEDS-sp	method of exact Doppler spread with set partitioning
MEA	method of equal area
MIMO	multiple-input multiple-output
MMEA	modified method of equal area
MSEM	mean-square error method
LAN	local area network
LCR	level-crossing rate
LPNM	L_p-norm method
PAM	phase-amplitude modulation
PDF	probability density function
PSD	Doppler power spectral density function
PSK	phase shift keying
TU	typical urban
RA	rural
R-MEDS	randomized method of exact Doppler spread
RTM	radio telephony mobile
QAM	quadrature amplitude modulation

SYMBOLS

$\in$	is an element of
$[a,b]$	set of real numbers within the closed interval from a to b
$(a,b]$	set of real numbers within the left-hand side open interval from a to b
$\{x_n\}_{n=1}^{N}$	set of elements $x_1, x_2, ..., x_N$
e^x	exponential function
$E\{x\}$	(statistical) mean value or expected value value of x
lim	limit
$\ln x$	natural logarithm
$\log_a x$	logarithm of x to base of a
mod	modulo operation
$P(\mu \le x)$	probability that the event μ is less than or equal to x
$\mathrm{Re}\{x\}$	real part of $x = x_1 + j\,x_2$
$round\{x\}$	nearest integer to x
x^*	complex conjugate of the number $x = x_1 + j\,x_2$
$\lvert x \rvert$	absolute value of x
$\sqrt{x}$	principal value of the square root of x
$\prod_{n=1}^{N}$	multiple product
$\sum_{n=1}^{N}$	multiple sum
$\int_a^b x(t)\,dt$	integral of the function $x(t)$ over the interval $[a,b]$
$x \to a$	x tends to a
$\lfloor x \rfloor$	floor function, the greatest integer less than or equal to x
~	distributed according to (statistic) or asymptotically equal (analysis)
≤	less than or equal to
=	equal
$a_{m,n}$	the mth row and the nth of the matrix $(a_{m,n})$
A^T	transpose matrix of the matrix A
A^{-1}	inverse matrix of the matrix A
$\det A$	determinant of the matrix A
$erf(\cdot)$	error function
$\delta(\cdot)$	delta function
f	Doppler frequency
f_c	cut-off frequency
$f_{\max}$	maximum Doppler frequency
$h(\tau', t)$	time-variant impulse response
r	amplitude level

$r_{\mu_i\mu_i}(t)$	autocorrelation function of $\mu_i(t)$
$r_{\tau'\tau'}(\nu')$	frequency correlation function
$S_{\mu_i\mu_i}(f)$	power spectral density of $\mu_i(t)$
u_n	random variable, uniformly distributed in the interval $(0,1]$
$\lambda(t)$	lognormal process
$\mu_i(t)$	real Gaussian random process (stochastic reference model)
$\hat{\mu}_i(t)$	real stochastic process (stochastic simulation model)
σ_0^2	mean power of $\mu_i(t)$
τ	time difference between t_2 and t_1
τ'	continuous propagation delay
$\tilde{\tau}'_\ell$	discrete propagation delay of the ℓth path
$\tau'_{\max}$	maximum propagation delay
$\Delta\tilde{\tau}'_\ell$	propagation delay difference between τ'_ℓ and $\tau'_{\ell-1}$
K	number of simulated samples of a discrete deterministic process
$\tilde{a}_\ell$	delay coefficient of the ℓth path
$c_{i,n}$	Doppler coefficient of the nth component of $\tilde{\mu}_i(t)$
$c_{i,n,\ell}$	Doppler coefficient of the nth component of $\tilde{\mu}_{i,\ell}(t)$
$f_{i,n}$	discrete Doppler frequency of the nth component of $\tilde{\mu}_i(t)$
$f_{i,n,\ell}$	discrete Doppler frequency of the nth component of $\tilde{\mu}_{i,\ell}(t)$
N_i	number of harmonic functions of $\tilde{\mu}_i(t)$
$\tilde{r}_{\mu_i\mu_i}(t)$	autocorrelation function of $\tilde{\mu}_i(t)$
$\tilde{S}_{\mu_i\mu_i}(f)$	power spectral density of $\tilde{\mu}_i(t)$
$\theta_{i,n}$	Doppler phase of the nth component of $\tilde{\mu}_i(t)$
$\theta_{i,n,\ell}$	Doppler phase of the nth component of $\tilde{\mu}_{i,\ell}(t)$

APPENDIX 1 - MATLAB PROGRAMS FOR PARAMETERS COMPUTATION METHODS

MATLAB-PROGRAMS of parameters computation methods for frequency-nonselective Channels

```
%--------------------------------------------------------------------
% parameter_Jakes.m -------------------------------------------------
%
% Program for the computation of the discrete Doppler frequencies,
% Doppler coefficients and Doppler phases by using the Jakes power
% spectral density.
%
% Used m-files: LPNM_opt_Jakes.m, fun_Jakes.m,
%                          grad_Jakes.m, acf_mue.m
%--------------------------------------------------------------------
% [f_i_n,c_i_n,theta_i_n]=parameter_Jakes(METHOD,N_i,...
%                                  sigma_0_2,f_max,PHASE,K,PLOT)
%--------------------------------------------------------------------
% Explanation of the input parameters:
%
% METHOD:
% |-----------------------------------------------|------------------|
% | Methods for the computation of the discrete   |      Input       |
% | Doppler frequencies and Doppler coefficients  |                  |
% |-----------------------------------------------|------------------|
% |-----------------------------------------------|------------------|
% | Method of equal distances (MED)               |     'ed_j'       |
% |-----------------------------------------------|------------------|
% | Mean square error method  (MSEM)              |     'ms_j'       |
% |-----------------------------------------------|------------------|
% | Method of equal areas (MEA)                   |     'ea_j'       |
% |-----------------------------------------------|------------------|
% | Monte Carlo method (MCM)                      |     'mc_j'       |
% |-----------------------------------------------|------------------|
% | Lp-norm method (LPNM)                         |     'lp_j'       |
% |-----------------------------------------------|------------------|
% | Method of exact Doppler spread (MEDS)         |     'es_j'       |
% |-----------------------------------------------|------------------|
% | Jakes method (JM)                             |     'jm_j'       |
% |-----------------------------------------------|------------------|
% | Randomized MEDS (R-MEDS)                      |     'rm_j'       |
% |-----------------------------------------------|------------------|
% | MEDS with set partitionning (MEDS-sp)         |     'sp_j'       |
% |-----------------------------------------------|------------------|
% N_i: number of harmonic functions
% sigma_0_2: average power of the real deterministic Gaussian
%            process mu_i(t)
% f_max: maximum Doppler frequency
%
% PHASE:
% |-----------------------------------------------|------------------|
% | Methods for the computation of the Doppler    |      Input       |
% | phases                                        |                  |
% |-----------------------------------------------|------------------|
% |-----------------------------------------------|------------------|
% | Random Doppler phases                         |     'rand'       |
% |-----------------------------------------------|------------------|
% | Permuted Doppler phases                       |     'perm'       |
% |-----------------------------------------------|------------------|
```

```
%
% K: number of sets (partitions)
% PLOT: plot of the ACF and the PSD of mu_i(t), if PLOT==1

function [f_i_n,c_i_n,theta_i_n]=parameter_Jakes(METHOD,N_i,...
                                 sigma_0_2,f_max,PHASE,K,PLOT)

if nargin<7,
   error('Not enough input parameters')
end

sigma_0=sqrt(sigma_0_2);

% Method of equal distances (MED)
if     METHOD=='ed_j',
       n=(1:N_i)';
       f_i_n=f_max/(2*N_i)*(2*n-1);
       c_i_n=2*sigma_0/sqrt(pi)*(asin(n/N_i)-asin((n-1)/N_i)).^0.5;

% Mean square error method (MSEM)
elseif METHOD=='ms_j',
       n=(1:N_i)';
       f_i_n=f_max/(2*N_i)*(2*n-1);
       Tp=1/(2*f_max/N_i);
       t=linspace(0,Tp,5E3);
       Jo=besselj(0,2*pi*f_max*t);
       c_i_n=zeros(size(f_i_n));
       for k=1:length(f_i_n),
           c_i_n(k)=2*sigma_0*...
                    sqrt(1/Tp*( trapz( t,Jo.*...
                    cos(2*pi*f_i_n(k)*t )) ));
       end

% Method of equal areas (MEA)
elseif METHOD=='ea_j'
       n=(1:N_i)';
       f_i_n=f_max*sin(pi*n/(2*N_i));
       c_i_n=sigma_0*sqrt(2/N_i)*ones(size(n));

% Monte Carlo method (MCM)
elseif METHOD=='mc_j'
       n=rand(N_i,1);
       f_i_n=f_max*sin(pi*n/2);
       c_i_n=sigma_0*sqrt(2/N_i)*ones(size(n));


% Lp-norm method (LPNM)
elseif METHOD=='lp_j',
       if   exist('fminsearch')~=2
            disp([' =====> This method requires ',...
                  'the Optimization Toolbox !!'])
            return
       else
            N=25;
            p=2;   % Norm
            PLOT=1;
            [f_i_n,c_i_n]=LPNM_opt_Jakes(N,f_max,sigma_0,p,N_i,K,PLOT);
       end
```

```
% Method of exact Doppler spread (MEDS)
elseif METHOD=='es_j',
       n=(1:N_i)';
       f_i_n=f_max*sin(pi/(2*N_i)*(n-1/2));
       c_i_n=sigma_0*sqrt(2/(N_i))*ones(size(f_i_n));

% Jakes method (JM)
elseif METHOD=='jm_j',
       n=1:N_i-1;
       f_i_n=f_max*[[cos(pi*n/(2*(N_i-1/2))),1]',...
                    [cos(pi*n/(2*(N_i-1/2))),1]'];
       c_i_n=2*sigma_0/sqrt(N_i-1/2)*[[sin(pi*n/(N_i-1)),1/2]',...
                                      [cos(pi*n/(N_i-1)),1/2]'];
       theta_i_n=zeros(size(f_i_n));
       PHASE='none';

% Randomized method of exact Doppler spread (R-MEDS)
elseif METHOD=='rm_j'
       n=(1:N_i);
       f_i=[];
       for i=1:K
          f_i_n(i,:)=f_max*sin(pi/(2*N_i)*(n-1/2) +...
                unifrnd(-pi,pi,1,N_i)/(4*N_i));
          f_i=[f_i,f_i_n(i,:)];
       end
       c_i_n=sigma_0*sqrt(2/(N_i))*ones(size(f_i_n));
       c_i=sigma_0*sqrt(2/(N_i))*ones(size(f_i));

% Method of exact doppler spread with set partitioning (MEDS-SP)
elseif METHOD=='sp_j'
       n=(1:N_i);
       f_i=[];
       for i=1:K
           f_i_n(i,:)=f_max*sin(pi/(2*N_i)*(n-1/2) +...
                pi*(i-(K+1)/2)/(2*K*N_i));
           f_i=[f_i,f_i_n(i,:)];
       end
       c_i_n=sigma_0*sqrt(2/(N_i))*ones(size(f_i_n));
       c_i=sigma_0*sqrt(2/(N_i))*ones(size(f_i));
else
       error('Method is unknown')
end

% Computation of the Doppler phases:
if     PHASE=='rand',
       theta_i_n=rand(N_i,1)*2*pi;

elseif PHASE=='perm',
       n=(1:N_i)';
       Z=rand(size(n));
       [dummy,I]=sort(Z);
       theta_i_n=2*pi*n(I)/(N_i+1);
end;

if PLOT==1,
   if  METHOD=='jm_j'
       subplot(2,3,1)
       stem([-f_i_n(N_i:-1:1,1);f_i_n(:,1)],...
             1/4*[c_i_n(N_i:-1:1,1);c_i_n(:,1)].^2,'k')
       set(0,'DefaultAxesFontSize',16);
```

```
        title('i=1')
        xlabel({'f (Hz)'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
        ylabel({'PSD'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
        subplot(2,3,2)
        stem([-f_i_n(N_i:-1:1,2);f_i_n(:,2)],...
              1/4*[c_i_n(N_i:-1:1,2);c_i_n(:,2)].^2,'k')
        set(0,'DefaultAxesFontSize',16);
        title('i=2')
        xlabel({'f (Hz)'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
        ylabel({'PSD'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
        tau_max=N_i/(K*f_max);
        tau=linspace(0,tau_max,500);
        r_mm=sigma_0^2*besselj(0,2*pi*f_max*tau);
        r_mm_tilde1=acf_mue(f_i_n(:,1),c_i_n(:,1),tau);
        subplot(2,3,4)
        plot(tau*f_max,r_mm,'k-',tau*f_max,r_mm_tilde1,'k--')
        xlim([0 tau_max*f_max])
        set(0,'DefaultAxesFontSize',16);
        title('i=1')
        xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
        ylabel({'ACF'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
        r_mm_tilde2=acf_mue(f_i_n(:,2),c_i_n(:,2),tau);
        subplot(2,3,5)
        plot(tau*f_max,r_mm,'k-',tau*f_max,r_mm_tilde2,'k--')
        xlim([0 tau_max*f_max])
        set(0,'DefaultAxesFontSize',16);
        title('i=2')
        xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
       ylabel({'ACF'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
        subplot(2,3,3)
        stem([-f_i_n(N_i:-1:1,1);f_i_n(:,1)],...
              1/4*[c_i_n(N_i:-1:1,1);c_i_n(:,1)].^2+...
              1/4*[c_i_n(N_i:-1:1,2);c_i_n(:,2)].^2,'k')
        set(0,'DefaultAxesFontSize',16);
        title('i=1,2')
        xlabel({'f (Hz)'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
       ylabel({'PSD'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
        subplot(2,3,6)
        plot(tau*f_max,2*r_mm,'k-',tau*f_max,r_mm_tilde1+r_mm_tilde2,'k--')
        xlim([0 tau_max*f_max])
        set(0,'DefaultAxesFontSize',16);
        title('i=1,2')
        xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
       ylabel({'ACF'},'FontName','Arial','FontSize',
             24,'Interpreter','latex')
    elseif METHOD=='sp_j'|METHOD=='rm_j'
        subplot(1,2,1)
        stem([-f_i(N_i*K:-1:1);f_i],...
              1/4*[c_i(N_i*K:-1:1);c_i].^2,'k')
        set(0,'DefaultAxesFontSize',16);
```

```
        xlabel({'f (Hz)'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')

ylabel({'LDS'},'FontName','Arial','FontSize',24,'Interpreter','latex')
        tau_max=N_i*K/(2*f_max);
        tau=linspace(0,tau_max,500);
        r_mm=sigma_0^2*besselj(0,2*pi*f_max*tau);
        r_mm_tilde=acf_mue_avg(f_i_n,c_i_n,N_i,K,tau);
        subplot(1,2,2)
        plot(tau*f_max,r_mm,'k-',tau*f_max,r_mm_tilde,'k--')
        xlim([0 tau_max*f_max])
        set(0,'DefaultAxesFontSize',16);
        xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
        ylabel({'samples mean ACF, $\overline{r}_{\mu_i\mu_i}(\tau)$'},...
            'FontName','Arial','FontSize',24,'Interpreter','latex')
    else
        subplot(1,2,1)
        stem([-f_i_n(N_i:-1:1);f_i_n],...
              1/4*[c_i_n(N_i:-1:1);c_i_n].^2,'k')
        set(0,'DefaultAxesFontSize',16);
        xlabel({'f (Hz)'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
       ylabel({'LDS'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
        tau_max=N_i/(K*f_max);
        tau=linspace(0,tau_max,500);
        r_mm=sigma_0^2*besselj(0,2*pi*f_max*tau);
        r_mm_tilde=acf_mue(f_i_n,c_i_n,tau);
        subplot(1,2,2)
        plot(tau*f_max,r_mm,'k-',tau*f_max,r_mm_tilde,'k--')
        xlim([0 tau_max*f_max])
        set(0,'DefaultAxesFontSize',16);
        xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',...
             24,'Interpreter','latex')
      label({'ACF'},'FontName','Arial','FontSize',24,'Interpreter','latex')
    end
end

%-------------------------------------------------------------------
% fun_Jakes.m -------------------------------------------------------
%
% Computation of the error function for the optimization of the discrete
%   Doppler frequencies (Jakes PSD).
%
% Used m-file: acf_mue.m
%-------------------------------------------------------------------
% F=fun_Jakes(x)
%-------------------------------------------------------------------
% Explanation of the input parameters:
%
% x: parameter vector to be optimized

function F=fun_Jakes(x)

load data

f_i_n=x;
r=acf_mue(f_i_n,c_i_n,tau);
F=norm(Jo-r,p);
```

```
f_i_n=x;
r=acf_mue(f_i_n,c_i_n,tau);
if PLOT==1,
   subplot(1,2,1)
   stem(f_i_n,c_i_n)
   set(0,'DefaultAxesFontSize',16);

xlabel({'$f_{in}$'},'FontName','Arial','FontSize',24,'Interpreter','latex')

ylabel({'$c_{in}$'},'FontName','Arial','FontSize',24,'Interpreter','latex')
   title({['$N_i$ = ',num2str(N_i)]},'FontName','Arial','FontSize',...
            24,'Interpreter','latex')
   subplot(1,2,2)
   plot(tau,Jo,tau,r)
   set(0,'DefaultAxesFontSize',16);
   xlabel({'$\tau$ (s)'},'FontName','Arial','FontSize',...
            24,'Interpreter','latex')
   ylabel({'ACF'},'FontName','Arial','FontSize',24,'Interpreter','latex')
   title({['$Error-norm$ =',num2str(F)]},'FontName','Arial','FontSize',...
            24,'Interpreter','latex')
   pause(0)
end

save x x

%-------------------------------------------------------------------
% LPNM_opt_Jakes.m ------------------------------------------------
%
% Program for the computation of the discrete Doppler frequencies
% employing the Jakes PSD by using a numerical optimization method.
%
% Used m-files: parameter_Jakes.m, fun_Jakes.m,
%                           grad_Jakes.m, acf_mue.m
%-------------------------------------------------------------------
% [f_i_n,c_i_n]=LPNM_opt_Jakes(N,f_max,sigma_0_2,p,N_i,PLOT)
%-------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: length of vector tau
% f_max: maximum Doppler frequency
% sigma_0_2: average power of the real Gaussian process mu_i(t)
% p: parameter of the Lp-norm (here: p=2,4,6,...)
% N_i: number of harmonic functions
% K: number of constellation schemes
% PLOT: display of the intermediate optimization results, if PLOT==1

function [f_i_n,c_i_n]=LPNM_opt_Jakes(N,f_max,sigma_0_2,p,N_i,K,PLOT)

tau=linspace(0,N_i/(2*f_max),N);
Jo=sigma_0_2*besselj(0,2*pi*f_max*tau);
c_i_n=sqrt(sigma_0_2)*sqrt(2/N_i)*ones(N_i,1);

save data Jo tau N_i c_i_n p PLOT

% Initial values:
[f_i_n,dummy1]=parameter_Jakes('es_j',N_i,sqrt(sigma_0_2),f_max,'none',K,0)
;

xo=f_i_n;
options = optimset('Display','iter','MaxIter',2000,'TypicalX',xo);
```

```
x=fminsearch('fun_jakes',xo,options);
load x
f_i_n=x;

%-----------------------------------------------------------------------
% parameter_Gauss.m ----------------------------------------------------
%
% Program for the computation of the discrete Doppler frequencies,
% Doppler coefficients, and Doppler phases by using the Gaussian
% power spectral density.
%
% Used m-files: LPNM_opt_Gauss.m, fun_Gauss.m,
%                          grad_Gauss.m, acf_mue.m
%-----------------------------------------------------------------------
% [f_i_n,c_i_n,theta_i_n]=parameter_Gauss(METHOD,N_i,sigma_0_2,...
%                                         f_max,f_c,PHASE,PLOT)
%-----------------------------------------------------------------------
% Explanation of the input parameters:
%
% METHOD:
% |-----------------------------------------------|------------------|
% | Methods for the computation of the discrete   |      Input       |
% | Doppler frequencies and Doppler coefficients  |                  |
% |-----------------------------------------------|------------------|
% |-----------------------------------------------|------------------|
% | Method of equal distances (MED)               |     'ed_g'       |
% |-----------------------------------------------|------------------|
% | Mean square error method  (MSEM)              |     'ms_g'       |
% |-----------------------------------------------|------------------|
% | Method of equal areas (MEA)                   |     'ea_g'       |
% |-----------------------------------------------|------------------|
% | Monte Carlo method (MCM)                      |     'mc_g'       |
% |-----------------------------------------------|------------------|
% | Lp-norm method (LPNM)                         |     'lp_g'       |
% |-----------------------------------------------|------------------|
% | Method of exact Doppler spread (MEDS)         |     'es_g'       |
% |-----------------------------------------------|------------------|
%
% N_i: number of harmonic functions
% sigma_0_2: average power of the real deterministic Gaussian
%            process mu_i(t)
% f_max: maximum Doppler frequency
% f_c: 3-dB-cutoff frequency
%
% PHASE:
% |-----------------------------------------------|------------------|
% | Methods for the computation of the Doppler    |      Input       |
% | phases                                        |                  |
% |-----------------------------------------------|------------------|
% |-----------------------------------------------|------------------|
% | Random Doppler phases                         |     'rand'       |
% |-----------------------------------------------|------------------|
% | Permuted Doppler phases                       |     'perm'       |
% |-----------------------------------------------|------------------|
%
% PLOT: plot of the ACF and the PSD of mu_i(t), if PLOT==1

function [f_i_n,c_i_n,theta_i_n]=parameter_Gauss(METHOD,N_i,...
                                 sigma_0_2,f_max,f_c,PHASE,PLOT)

if nargin<7,
```

```
   error('Not enough input parameters')
end

sigma_0=sqrt(sigma_0_2);
kappa_c=f_max/f_c;

% Method of equal distances (MED)
if      METHOD=='ed_g',
        n=(1:N_i)';
        f_i_n=kappa_c*f_c/(2*N_i)*(2*n-1);
        c_i_n=sigma_0*sqrt(2)*sqrt(erf(n*kappa_c*...
              sqrt(log(2))/N_i)-erf((n-1)*kappa_c*...
              sqrt(log(2))/N_i) );
        K=1;

% Mean square error method (MSEM)
elseif METHOD=='ms_g',
        n=(1:N_i)';
        f_i_n=kappa_c*f_c/(2*N_i)*(2*n-1);
        tau_max=N_i/(2*kappa_c*f_c);
        N=1E3;
        tau=linspace(0,tau_max,N);
        f1=exp(-(pi*f_c*tau).^2/log(2));
        c_i_n=zeros(size(f_i_n));
        for k=1:length(c_i_n),
            c_i_n(k)=2*sigma_0*sqrt(trapz(tau,f1.*...
                     cos(2*pi*f_i_n(k)*tau))/tau_max);
        end
        K=1;

% Method of equal areas (MEA)
elseif METHOD=='ea_g'
        n=(1:N_i)';
        c_i_n=sigma_0*sqrt(2/N_i)*ones(size(n));
        f_i_n=f_c/sqrt(log(2))*erfinv(n/N_i);
        f_i_n(N_i)=f_c/sqrt(log(2))*erfinv(0.9999999);
        K=1;

% Monte Carlo method (MCM)
elseif METHOD=='mc_g'
        n=rand(N_i,1);
        f_i_n=f_c/sqrt(log(2))*erfinv(n);
        c_i_n=sigma_0*sqrt(2/N_i)*ones(size(n));
        K=1;

% Lp-norm method (LPNM)
elseif METHOD=='lp_g',

        if   exist('fminunc')~=2
             disp([' =====> This method requires ',...
                   'the Optimization Toolbox !!'])
             return
        else
             N=1e2;
             p=2;
             [f_i_n,c_i_n]=LPNM_opt_Gauss(N,f_max,f_c,...
                           sigma_0_2,p,N_i,PLOT);
             K=2;
        end
```

```
% Method of exact Doppler spread (MEDS)
elseif METHOD=='es_g',
       n=(1:N_i)';
       c_i_n=sigma_0*sqrt(2/N_i)*ones(size(n));
       f_i_n=f_c/sqrt(log(2))*erfinv((2*n-1)/(2*N_i));
       K=1;
else
       error([setstr(10),'Method is unknown'])
end

% Computation of the Doppler phases:
if     PHASE=='rand',
       theta_i_n=rand(N_i,1)*2*pi;
elseif PHASE=='perm',
       n=(1:N_i)';
       Z=rand(size(n));
       [dummy,I]=sort(Z);
       theta_i_n=2*pi*n(I)/(N_i+1);
end

if PLOT==1,
   subplot(1,2,1)
   stem([-f_i_n(N_i:-1:1);f_i_n],...
        1/4*[c_i_n(N_i:-1:1);c_i_n].^2,'k')
   set(0,'DefaultAxesFontSize',16);
   xlabel({'f (Hz)'},'FontName','Arial','FontSize',...
            24,'Interpreter','latex')
   ylabel({'PSD'},'FontName','Arial','FontSize',24,'Interpreter','latex')
   tau_max=N_i/(K*kappa_c*f_c);
   tau=linspace(0,tau_max,500);
   r_mm=sigma_0_2*exp(-(pi*f_c/sqrt(log(2))*tau).^2);
   r_mm_tilde=acf_mue(f_i_n,c_i_n,tau);
   subplot(1,2,2)
   plot(tau*f_max,r_mm,'k-',tau*f_max,r_mm_tilde,'k--')
   xlim([0 tau_max*f_max])
   xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',...
            24,'Interpreter','latex')
   ylabel({'ACF'},'FontName','Arial','FontSize',24,'Interpreter','latex')
end

%-----------------------------------------------------------------------
% fun_Gauss.m ----------------------------------------------------------
%
% Computation of the error function for the optimization of the
%  discrete Doppler frequencies (Gaussian PSD).
%
% Used m-file: acf_mue.m
%-----------------------------------------------------------------------
% F=fun_Gauss(x)
%-----------------------------------------------------------------------
% Explanation of the input parameters:
%
% x: parameter vector to be optimized

function F=fun_Gauss(x)

load data

f_i_n=x;
```

```
r=acf_mue(f_i_n,c_i_n,tau);
F=norm(abs(r_mm-r),p);
if PLOT==1,
   subplot(1,2,1)
   stem(f_i_n,c_i_n)
   set(0,'DefaultAxesFontSize',16);
   xlabel({'$f_{in}$'},'FontName','Arial','FontSize',...
      24,'Interpreter','latex')
   ylabel({'$c_{in}$'},'FontName','Arial','FontSize',...
      24,'Interpreter','latex')
   title({['$N_i$ = ',num2str(N_i)]},'FontName','Arial','FontSize',...
            24,'Interpreter','latex')
   subplot(1,2,2)
   plot(tau,r_mm,tau,r)
  set(0,'DefaultAxesFontSize',16);
   xlabel({'$\tau$ (s)'},'FontName','Arial','FontSize',...
            24,'Interpreter','latex')
   ylabel({'ACF'},'FontName','Arial','FontSize',24,'Interpreter','latex')
   title({['$Error-norm$ =',num2str(F)]},'FontName','Arial','FontSize',...
            24,'Interpreter','latex')
   pause(0)

end

save x x

%-----------------------------------------------------------------------
% LPNM_opt_Gauss.m ----------------------------------------------------
%
% Program for the computation of the discrete Doppler frequencies
% employing the Gaussian PSD by using a numerical optimization
% method.
%
% Used m-files: parameter_Gauss.m, fun_Gauss.m,
%               grad_Gauss.m, acf_mue.m
%-----------------------------------------------------------------------
% [f_i_n,c_i_n]=LPNM_opt_Gauss(N,f_max,f_c,sigma_0_2,p,N_i,PLOT)
%-----------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: length of vector tau
% f_max: maximum Doppler frequency
% f_c: 3-dB-cutoff frequency
% sigma_0_2: average power of the real Gaussian process mu_i(t)
% p: parameter of the Lp-norm (here: p=2,4,6,...)
% N_i: number of harmonic functions
% PLOT: display of the intermediate optimization results, if PLOT==1

function [f_i_n,c_i_n]=LPNM_opt_Gauss(N,f_max,f_c,sigma_0_2,...
                                      p,N_i,PLOT)

kappa_c=f_max/f_c;

F_list=[];
save F_list F_list

tau_max=N_i/(2*kappa_c*f_c);
tau=linspace(0,tau_max,N);
r_mm=sigma_0_2*exp(-(pi*f_c/sqrt(log(2))*tau).^2);
```

```
[f_i_n,c_i_n]=parameter_Gauss('es_g',N_i,sigma_0_2,f_max,...
                                    f_c,'none',PLOT);

save data r_mm tau N_i c_i_n p PLOT

xo=f_i_n;
options = optimset('Display','iter','MaxIter',2000,'TypicalX',xo);
x=fminsearch('fun_Gauss',xo,options);

load x

f_i_n=sort(abs(x));
```

MATLAB-PROGRAMS of parameters computation methods for frequency-selective Channels

```
%-------------------------------------------------------------------------
% Parameters_freq_selective.m ----------------------------------------
%
% Program for the computation of the gains and the delays
% of frequency-selective channels using different methods
%
% used files: RA_data.mat, TU_data.mat, BU_data.mat, HT_data.mat
%-------------------------------------------------------------------------
% [tau_teld,a_teld]=Paramters_freq_selective(L,Area,method,PLOT)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% L: number of path in the tapped-delay-line structure
% Area: According to COST 207, 4 types of channels are specified:
%           1) Rural Area:    'RA'
%           2) Typical Urban: 'TU'
%           3) Bad Urban:     'BU'
%           4) Hilly Terrain: 'HT'
% method: the most commonly used method for parameter computation founded
% in litterature:
%           1) Method of Equal distances: 'MED'
%           2) Method of Equal Areas:     'MEA'
%           3) Mean Square Error Method:  'MSEM'
%           4) Monte Carlo Method:        'MCM'
%           4) Lp-nom Method:             'LPNM'
% PLOT: plot of the resulting autocrrelation function r_tau(v),
%       if PLOT==1

function [tau_teld,a_teld]=Paramters_freq_selective(L,Area,method,PLOT)
if nargin==3,
    PLOT=0;
end
if all(upper(Area)=='RA')
    tau_max=0.7;
    b=9.2;
    c=9.2/(1-exp(-6.44)); % c_RA
    delta_tau=tau_max/(L-1);
    v_max=1/(2*delta_tau);
    v=linspace(0,v_max,5000);
    save RA_data.mat L tau_max b c delta_tau v_max v;
end
if all(upper(Area)=='TU')
```

```
    tau_max=7;
    b=1;
    c=1/(1-exp(-7)); % c_TU
    delta_tau=tau_max/(L-1);
    v_max=1/(2*delta_tau);
    v=linspace(0,v_max,5000);
    save TU_data.mat L tau_max b c delta_tau v_max v;
end
if all(upper(Area)=='BU')
    tau_max=10;
    tau1=5;
    tau2=5;
    b1=1;
    b2=5;
    b3=1;
    c1=2/(3-3*exp(-5)); % c_BU
    c2=0.5*c1;
    delta_tau=tau_max/(L-1);
    v_max=1/(2*delta_tau);
    v=linspace(0,v_max,5000);
    L1=floor(tau1/delta_tau);
    L2=round(tau2/delta_tau);
    L3=round(tau1/delta_tau);
    A=c1/b1*(1-exp(-b1*tau1));
    L4=floor(A*L);
    save BU_data.mat L tau_max tau1 tau2 b1 b2 b3 ...
             c1 c2 delta_tau v_max v L1 L2 L3 L4 A;
end
if all(upper(Area)=='HT')
    tau_max=20;
    tau1=2;
    tau2=15;
    b1=3.5;
    b2=15;
    b3=1;
    c1=1/((1-exp(-7))/3.5+(1-exp(-5))/10); % c_HT
    c2=0.1*c1;
    delta_tau=(tau_max-tau2+tau1)/(L-1);
    v_max=1/(2*delta_tau);
    v=linspace(0,v_max,5000);
    L1=floor(tau1/delta_tau);
    L2=round(tau2/delta_tau);
    A=c1/b1*(1-exp(-b1*tau1));
    L4=floor(A*L);
    save HT_data.mat L tau_max tau1 tau2 b1 b2 b3 ...
             c1 c2 delta_tau v_max v L1 L2 L4 A;
end
% computation of the gains and the delays according to different methods
switch method
    case 'MED'
        if all(upper(Area)=='RA')|all(upper(Area)=='TU')
            r_tau=c./(b+j*2*pi*v).*(1-exp(-tau_max*(b+j*2*pi.*v)));
            % determination of the delays tau_teld_l
            tau_teld=[0:L-1]*delta_tau;
            % determination of the gains a_teld_l
            a_teld=ones(1,L);
            a_teld(1)=sqrt(c./b*(1-exp(-b*delta_tau/2)));
            a_teld(L)=sqrt(c./b*(exp(-b*...
            (tau_max-delta_tau/2)-exp(-b*tau_max))));
            a_teld(2:L-1)=sqrt(c./b*(exp(-b*...
            (tau_teld(2:L-1)-delta_tau/2))-...
```

```
            exp(-b*(tau_teld(2:L-1)+delta_tau/2))));
    elseif all(upper(Area)=='BU')
        r_tau=c1./(b1+j*2*pi*v).*(1-exp(-tau_max*(b1+j*2*pi*v)))+...
            c2*exp(b2).*(exp(-tau2*(b3+j*2*pi*v))-...
              exp(-tau_max*(b3+j*2*pi*v))./(b3+j*pi*v);
        % determination of the delays tau_teld_l
        tau_teld=zeros(1,L);
        tau_teld(1:L1+1)=[0:L1]*delta_tau;
        % l index on tau
        l=[L1+1:L-1];
        tau_teld(L1+2:L)=(l+L2-L1-1)*delta_tau;
        % determination of the gains a_teld_l
        a_teld=zeros(1,L);
        a_teld(1)=sqrt(c1/b1*(1-exp(-b1*delta_tau/2)));
        a_teld(2:L3)=sqrt(c1./b1*(exp(-b1*...
              (tau_teld(2:L3)- delta_tau/2))-...
            exp(-b1*(tau_teld(2:L3)+delta_tau/2))));
        a_teld(L3+1)=sqrt(c1/b1*(exp(-b1*...
              (tau_teld(L3+1)-delta_tau/2))-exp(-b1*tau1))+...
              c2/b3*exp(b2)*(-exp(-b3*(tau_teld(L3+1)+...
                    delta_tau/2))+exp(-b3*tau2)));
        a_teld(L3+2:L-1)=sqrt(c2./b3*exp(b2)*...
        (exp(-b3*(tau_teld(L3+2:L-1)-delta_tau/2))-...
            exp(-b3*(tau_teld(L3+2:L-1)+delta_tau/2))));
        a_teld(L)=sqrt(c2/b3*exp(b2)*(exp(-b3*...
        (tau_max-delta_tau/2))-exp(-b3*tau_max)));
    elseif all(upper(Area)=='HT')
        r_tau=c1./(b1+j*2*pi*v).*(1-exp(-tau_max*(b1+j*2*pi*v)))+...
         c2*exp(b2).*(exp(-tau2*(b3+j*2*pi*v))-exp(-
        tau_max*(b3+j*2*pi*v))./(b3+j*pi*v);
        % determination of the delays tau_teld_l
        tau_teld=zeros(1,L);
        tau_teld(1:L1+1)=[0:L1]*delta_tau;
        % l index on tau
        l=[L1+1:L-1];
        tau_teld(L1+2:L)=(l+L2-L1-1)*delta_tau;
        % determination of the gains a_teld_l
        a_teld(1)=sqrt(c1/b1*(1-exp(-b1*delta_tau/2)));
        a_teld(2:L1)=sqrt(c1./b1*(exp(-b1*...
              (tau_teld(2:L1)-delta_tau/2))-...
            exp(-b1*(tau_teld(2:L1)+delta_tau/2))));
        a_teld(L1+1)=sqrt(c1/b1*(exp(-b1*...
              (tau_teld(L1+1)-delta_tau/2))-exp(-b1*tau1)));
        a_teld(L1+2)=sqrt(c2/b3*exp(b2)*...
              (exp(-b3*tau2)-exp(-b3*(tau_teld(L1+2)+delta_tau/2))));
        a_teld(L1+3:L-1)=sqrt(c2./b3*exp(b2)*...
              (exp(-b3*(tau_teld(L1+3:L-1)-delta_tau/2))-...
            exp(-b3*(tau_teld(L1+3:L-1)+delta_tau/2))));
        a_teld(L)=sqrt(c2/b3*exp(b2)*(exp(-b3*...
              (tau_teld(L)-delta_tau/2))-exp(-b3*tau_max)));
    else
        disp('Please input a right Area (RA,TU,BU,HT)');
    end
case 'MEA'
    if all(upper(Area)=='RA')|all(upper(Area)=='TU')
        r_tau=c./(b+j*2*pi*v).*(1-exp(-tau_max*(b+j*2*pi*v)));
        % determination of the gains a_teld_l
        a_teld=zeros(1,L);
        a_teld(1:L)=1/sqrt(L);
        % determination of the delays tau_teld_l
        tau_teld=(-1/b)*log(1-((b*[0:L-1])/(c*L)));
```

```
    elseif all(upper(Area)=='BU')|all(upper(Area)=='HT')
        r_tau=c1./(b1+j*2*pi*v).*(1-exp(-tau_max*(b1+j*2*pi*v)))+...
            c2*exp(b2).*(exp(-tau2*(b3+j*2*pi*v))...
        -exp(-tau_max*(b3+j*2*pi*v))./(b3+j*pi*v);
        % determination of the gains a_teld_l
        a_teld=zeros(1,L);
        a_teld(1:L)=1/sqrt(L);
        % determination of the delays tau_teld_l
        tau_teld=zeros(1,L);
        tau_teld(1:L4)=(-1/b1).*log(1-((b1.*[0:L4-1])./(c1*L)));
        tau_teld(L4+1:L)=-1/b3.*log(b3*...
        (A-[L4:L-1]./L)/(c2*exp(b2))+exp(-b3*tau2));
    else
        disp('Please input a right Area (RA,TU,BU,HT)');
    end
case 'MSEM'
    if all(upper(Area)=='RA')|all(upper(Area)=='TU')
        % determination of the delays tau_teld_l
        tau_teld=[0:L-1]*delta_tau;
        % computation of the frequency correlation function FCF
        r_tau=c./(b+j*2*pi*v).*(1-exp(-tau_max*(b+j*2*pi*v)));
        % determination of the gains a_teld
        for i=1:L,
              a_teld(i)=sqrt(real(1/v_max*...
              trapz(v,r_tau.*exp(j*2*pi*tau_teld(i)*v))));
        end
        a_teld=abs(a_teld);
    elseif all(upper(Area)=='BU')
        % determination of delays tau_teld_l
        tau_teld=[0:L-1]*delta_tau;
        % computation of the frequency correlation function FCF
        r_tau=c1./(b1+j*2*pi*v).*(1-exp(-tau_max*(b1+j*2*pi*v)))+...
            c2*exp(b2).*(exp(-tau2*(b3+j*2*pi*v))-...
              exp(-tau_max*(b3+j*2*pi*v))./(b3+j*pi*v);
        % determination of the gains a_teld
        for i=1:L,
              a_teld(i)=sqrt(real(1/v_max*...
              trapz(v,r_tau.*exp(j*2*pi*tau_teld(i)*v))));
        end
        a_teld=abs(a_teld);
    elseif all(upper(Area)=='HT')
        % determination of delays tau_teld_l
        tau_teld=zeros(1,L);
        tau_teld(1:L1+1)=[0:L1]*delta_tau;
        % l index on tau
        l=[L1+1:L-1];
        tau_teld(L1+2:L)=(l+L2-L1-1)*delta_tau;
        % computation of the frequency correlation function FCF
        r_tau=c1./(b1+j*2*pi*v).*(1-exp(-tau_max*(b1+j*2*pi*v)))+...
            c2*exp(b2).*(exp(-tau2*(b3+j*2*pi*v))-...
              exp(-tau_max*(b3+j*2*pi*v))./(b3+j*pi*v);
        % determination of the gains a_teld
        for i=1:L,
              a_teld(i)=sqrt(real(1/v_max*trapz(v,r_tau.*...
              exp(j*2*pi*tau_teld(i)*v))));
        end
        a_teld=abs(a_teld);
    else
        disp('Please input a right Area (RA,TU,BU,HT)');
    end
case 'MCM'
```

```
        if all(upper(Area)=='RA')|all(upper(Area)=='TU')
            r_tau=c./(b+j*2*pi*v).*(1-exp(-tau_max*(b+j*2*pi*v)));
            % computation of the real-valued c_tau
            c_tau=1/trapz(v,c*exp(-b*v));
            % generation of uniform distribution u_L
            u_L=rand(1,L);
            % mapping of u_L into tau_delta
            tau_teld=(-1/b)*log(1-((b*u_L)/(c*c_tau)));
            % computation of the gains a_teld
            a_teld=zeros(1,L);
            a_teld(1:L)=1/sqrt(L);
        elseif all(upper(Area)=='BU')|all(upper(Area)=='HT')
            r_tau=c1./(b1+j*2*pi*v).*(1-exp(-tau_max*(b1+j*2*pi*v)))+...
                c2*exp(b2).*(exp(-tau2*(b3+j*2*pi*v))-...
                  exp(-tau_max*(b3+j*2*pi*v))./(b3+j*pi*v);
            tau_teld=zeros(1,L);
            % generation of uniform distribution u_L
            u_L=unifrnd(0,1,1,L);
            % mapping of u_L into tau_delta
            tau_teld(1:L4)=(-1/b1)*log(1-((b1*A*u_L(1:L4))/c1));
            tau_teld(L4+1:L)=(-1/b3)*...
            log(exp(-b3*tau2)+((b3*A*u_L(L4+1:L))/(c1*exp(b2))));
            % computation of the gains a_teld
            a_teld=zeros(1,L);
            a_teld(1:L)=1/sqrt(L);
        else
            disp('Please input a right Area (RA,TU,BU,HT)');
        end
    case 'LPNM'
        if all(upper(Area)=='RA')
            % start values (parameter computed by using MSEM)
            r_tau=c./(b+j*2*pi*v).*(1-exp(-tau_max*(b+j*2*pi*v)));
            tau_teld=[0:L-1]*delta_tau;
            for i=1:L,
                  a_teld(i)=sqrt(real(1/v_max*...
                  trapz(v,r_tau.*exp(j*2*pi*tau_teld(i)*v))));
            end
            a_teld0=abs(a_teld);
            r_tau_teld=sum((a_teld.^2).'*ones(1,length(v)).*...
            exp((-j*2*pi*tau_teld).'*v));
            options = optimset('Display','iter','TolFun',...
            1*10^(-12),'MaxIter',1000,...
            'MaxFunEvals',100000000,'TypicalX',a_teld0,'TolX',1*10^(-12));
            x=fminsearch('errorfuncRA',a_teld0,options);
            a_teld=abs(x);
        elseif all(upper(Area)=='TU')
            % start values (parameter computed by using MSEM)
            r_tau=c./(b+j*2*pi*v).*(1-exp(-tau_max*(b+j*2*pi*v)));
            tau_teld=[0:L-1]*delta_tau;
            for i=1:L,
                  a_teld(i)=sqrt(real(1/v_max*...
                  trapz(v,r_tau.*exp(j*2*pi*tau_teld(i)*v))));
            end
            a_teld0=abs(a_teld);
            r_tau_teld=sum((a_teld.^2).'*ones(1,length(v)).*...
            exp((-j*2*pi*tau_teld).'*v));
            options = optimset('Display','iter','TolFun',...
            1*10^(-12),'MaxIter',1000,...
            'MaxFunEvals',100000000,'TypicalX',a_teld0,'TolX',1*10^(-12));
            x=fminsearch('errorfuncTU',a_teld0,options);
            a_teld=abs(x);
```

```
        elseif all(upper(Area)=='BU')
            % start values (parameter computed by using MSEM
            r_tau=c1./(b1+j*2*pi*v).*(1-exp(-tau_max*(b1+j*2*pi*v)))+...
                c2*exp(b2).*(exp(-tau2*(b3+j*2*pi*v))-...
                  exp(-tau_max*(b3+j*2*pi*v))./(b3+j*pi*v);
            tau_teld=[0:L-1]*delta_tau;
            for i=1:L,
              a_teld(i)=sqrt(real(1/v_max*...
                  trapz(v,r_tau.*exp(j*2*pi*tau_teld(i)*v))));
            end
            a_teld0=abs(a_teld);
            r_tau_teld=sum((a_teld.^2).'*ones(1,length(v)).*...
            exp((-j*2*pi*tau_teld).'*v));
            % determination of the gains a_teld
            options = optimset('Display','iter','TolFun',...
            1*10^(-12),'MaxIter',1000,...
            'MaxFunEvals',100000000,'TypicalX',a_teld0,'TolX',1*10^(-12));
            x=fminsearch('errorfuncBU',a_teld0,options);
            a_teld=abs(x);
        elseif all(upper(Area)=='HT')
            tau_teld=zeros(1,L);
            tau_teld(1:L1+1)=[0:L1]*delta_tau;
            % l index on tau
            l=[L1+1:L-1];
            tau_teld(L1+2:L)=(l+L2-L1-1)*delta_tau;
            % start values (parameter computed by using MSEM
            r_tau=c1./(b1+j*2*pi*v).*(1-exp(-tau_max*(b1+j*2*pi*v)))+...
                c2*exp(b2).*(exp(-tau2*(b3+j*2*pi*v))-...
                  exp(-tau_max*(b3+j*2*pi*v))./(b3+j*pi*v);
            for i=1:L,
              a_teld(i)=sqrt(real(1/v_max*...
                  trapz(v,r_tau.*exp(j*2*pi*tau_teld(i)*v))));
            end
            a_teld0=abs(a_teld);
            r_tau_teld=sum((a_teld.^2).'*ones(1,length(v)).*...
            exp((-j*2*pi*tau_teld).'*v));
            % determination of the gains a_teld
            options = optimset('Display','iter','TolFun',...
            1*10^(-12),'MaxIter',1000,...
            'MaxFunEvals',100000000,'TypicalX',a_teld0,'TolX',1*10^(-12));
            x=fminsearch('errorfuncHT',a_teld0,options);
            a_teld=abs(x);
        else
            disp('Please input a right Area (RA,TU,BU,HT)');
        end
    otherwise
        disp('Please input a right method (MED,MEA,MSEM,MCM,LPNM)');
end
if PLOT==1,
    r_tau_teld=sum((a_teld.^2).'*ones(1,length(v)).*...
      exp((-j*2*pi*tau_teld).'*v));
    plot(v,abs(r_tau),'k--');
    hold on;
    plot(v,abs(r_tau_teld));
    set(0,'DefaultAxesFontSize',16)
    title(['Frequency Correlation Function (magnitude): COST 207, ',...
      upper(Area),', ',upper(method),...
       ',L=',num2str(L)],'Interpreter','latex','FontSize',24);
    legend({['reference model, with L=',num2str(L)],...
```

```
            ['simulation model, with L=',num2str(L)]},...
      'FontName','Arial','Location','NorthEast','FontSize',22,'Interpreter'
      ,'latex');
    xlabel('$\upsilon\ [MHz]$','Interpreter','latex','FontSize',24);
    ylabel('$FCF$','Interpreter','latex','FontSize',24);
end

%-------------------------------------------------------------------------
% errorfuncRA.m-----------------------------------------------------------
%
% Program for the computation of the error function of the reference
% model and the simulation model that will be used in LPNM method
%
% used files: RA_data.mat
%-------------------------------------------------------------------------
% [E]=errorfuncRA(a_teld)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% a_teld: gains of different ellipses

function [E]=errorfuncRA(a_teld)
p=2;
load RA_data.mat;
tau_teld=[0:L-1]*delta_tau;
r_tau=c./(b+j*2*pi*v).*(1-exp(-tau_max*(b+j*2*pi*v)));
r_tau_teld=sum((a_teld.^2).'*ones(1,length(v)).*exp((-
j*2*pi*tau_teld).'*v));
% The error function between time ACF reference and time ACF simulation
E=norm(abs(r_tau-r_tau_teld),p);

%-------------------------------------------------------------------------
% errorfuncTU.m-----------------------------------------------------------
%
% Program for the computation of the error function of the reference
% model and the simulation model that will be used in LPNM method
%
% used files: TU_data.mat
%-------------------------------------------------------------------------
% [E]=errorfuncTU(a_teld)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% a_teld: gains of different ellipses

function [E]=errorfuncTU(a_teld)
p=2;
load TU_data.mat;
tau_teld=[0:L-1]*delta_tau;
r_tau=c./(b+j*2*pi*v).*(1-exp(-tau_max*(b+j*2*pi*v)));
r_tau_teld=sum((a_teld.^2).'*ones(1,length(v)).*exp((-
j*2*pi*tau_teld).'*v));
% The error function between time ACF reference and time ACF simulation
E=norm(abs(r_tau-r_tau_teld),p);

%-------------------------------------------------------------------------
% errorfuncBU.m-----------------------------------------------------------
%
% Program for the computation of the error function of the reference
% model and the simulation model that will be used in LPNM method
```

```
%
% used files: RA_data.mat
%-------------------------------------------------------------------------
% [E]=errorfuncBU(a_teld)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% a_teld: gains of different ellipses

function [E]=errorfuncBU(a_teld)
p=2;
load BU_data.mat;
tau_teld=[0:L-1]*delta_tau;
r_tau=c1./(b1+j*2*pi*v).*(1-exp(-tau_max*(b1+j*2*pi*v)))+...
                c2*exp(b2).*(exp(-tau2*(b3+j*2*pi*v))-exp(-
tau_max*(b3+j*2*pi*v))./(b3+j*pi*v);
r_tau_teld=sum((a_teld.^2).'*ones(1,length(v)).*exp((-
j*2*pi*tau_teld).'*v));
% The error function between time ACF reference and time ACF simulation
E=norm(abs(r_tau-r_tau_teld),p);

%-------------------------------------------------------------------------
% errorfuncHT.m-----------------------------------------------------------
%
% Program for the computation of the error function of the reference
% model and the simulation model that will be used in LPNM method
%
% used files: RA_data.mat
%-------------------------------------------------------------------------
% [E]=errorfuncHT(a_teld)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% a_teld: gains of different ellipses

function [E]=errorfuncHT(a_teld)
p=2;
load HT_data.mat;
tau_teld=zeros(1,L);
tau_teld(1:L1+1)=[0:L1]*delta_tau;
r_tau=c1./(b1+j*2*pi*v).*(1-exp(-tau_max*(b1+j*2*pi*v)))+...
                c2*exp(b2).*(exp(-tau2*(b3+j*2*pi*v))-exp(-
tau_max*(b3+j*2*pi*v))./(b3+j*pi*v);
r_tau_teld=sum((a_teld.^2).'*ones(1,length(v)).*exp((-
j*2*pi*tau_teld).'*v));
% The error function between time ACF reference and time ACF simulation
E=norm(abs(r_tau-r_tau_teld),p);

%-------------------------------------------------------------------
% F_S_K_p.m ---------------------------------------------------------
%
% Program for the generation of the matrices used in F_S_K.m.
%
% Used m-file: pCOST207.m
%-------------------------------------------------------------------
% [C1,F1,TH1,C2,F2,TH2,F01,F02,RHO,F_RHO,q_l,T]=
%                       F_S_K_p(N_1,AREA,f_max,psf,method)
%-------------------------------------------------------------------
% Explanation of the input parameters:
%
```

```
% N_1: minimum number of discrete Doppler frequencies
% AREA: according to COST 207, 4 types of channels are specified:
%            1) Rural Area:    'RA'
%            2) Typical Urban: 'TU'
%            3) Bad Urban:     'BU'
%            4) Hilly Terrain: 'HT'
% f_max: maximum Doppler frequency
% psf: parameters of selective-frequency channels which can be taken from
% specification or computation:
%           1) specification 'SP'
%           2) computation 'CM'
% method: the most commonly used method for parameter computation founded
% in litterature:
%           1) Method of Equal distances: 'MED'
%           2) Method of Equal Areas:     'MEA'
%           3) Mean Square Error Method:  'MSEM'
%           4) Monte Carlo Method:        'MCM'
%           4) Lp-nom Method:             'LPNM'

function [C1,F1,TH1,C2,F2,TH2,F01,F02,RHO,F_RHO,q_l,T]=...
         F_S_K_p(N_1,Area,f_max,psf,method)

% The greatest common divisor of the discrete propagation delays
% defines the sampling interval T_s:

T_s=0.2E-6;
L_RA=4; % number of paths in the tapped-line-delays structure
L=6; % number of paths in the tapped-line-delays structure
if  all(upper(Area)=='RA'),
    DOPP_KAT=['RI';'JA';'JA';'JA'];
    if all(upper(psf)=='SP'),
        a_l=[1,0.63,0.1,0.01];
        tau_l=[0,0.2,0.4,0.6]*1E-6;
    elseif all(upper(psf)=='CM'),
        [tau_l,a_l]=Paramters_freq_selective(L_RA,Area,method);
    end

elseif all(upper(Area)=='TU'),
    DOPP_KAT=['JA';'JA';'G1';'G1';'G2';'G2'];
    if all(upper(psf)=='SP'),
       a_l=[0.5,1,0.63,0.25,0.16,0.1];
       tau_l=[0,0.2,0.6,1.6,2.4,5]*1E-6;
    elseif all(upper(psf)=='CM'),
       [tau_l,a_l]=Paramters_freq_selective(L,Area,method);
    end

elseif all(upper(Area)=='BU'),
    DOPP_KAT=['JA';'JA';'G1';'G1';'G2';'G2'];
    if all(upper(psf)=='SP'),
        a_l=[0.5,1,0.5,0.32,0.63,0.4];
        tau_l=[0,0.4,1.0,1.6,5.0,6.6]*1E-6;
    elseif all(upper(psf)=='CM'),
        [tau_l,a_l]=Paramters_freq_selective(L,Area,method);
    end

elseif all(uppper(Area)=='HT'),
    DOPP_KAT=['JA';'JA';'JA';'JA';'G2';'G2'];
    if all(upper(psf)=='SP'),
        a_l=[1,0.63,0.4,0.2,0.25,0.06];
        tau_l=[0,0.2,0.4,0.6,15,17.2]*1E-6;
```

```
    elseif all(upper(psf)=='CM'),
       [tau_l,a_l]=Paramters_freq_selective(L,Area,method);
    end
end

% Generate the parameters and assign them to the matrices:
num_of_taps=length(DOPP_KAT);
F1=zeros(num_of_taps,N_1+2*num_of_taps-1);
F2=F1;C1=F1;C2=F1;TH1=F1;TH2=F1;
F01=zeros(1,num_of_taps);F02=F01;
RHO=zeros(1,num_of_taps);F_RHO=RHO;
NN1=N_1+2*(num_of_taps-1):-2:N_1;
for k=1:num_of_taps,
    [f1,f2,c1,c2,th1,th2,rho,f_rho,f01,f02]=...
    pCOST207(DOPP_KAT(k,:),NN1(k));
    F1(k,1:NN1(k))=f1;
    C1(k,1:NN1(k))=c1*sqrt(a_l(k));
    TH1(k,1:NN1(k))=th1;
    F2(k,1:NN1(k)+1)=f2;
    C2(k,1:NN1(k)+1)=c2*sqrt(a_l(k));
    TH2(k,1:NN1(k)+1)=th2;
    F01(k)=f01;F02(k)=f02;
    RHO(k)=rho;F_RHO(k)=f_rho;
end

% Determine indices of the delay elements of the FIR filter:
q_l=tau_l/T_s+1;
% Initialization of the delay elements of the FIR filter:
T=zeros(1,max(q_l));



%-------------------------------------------------------------------
% pCOST207.m --------------------------------------------------------
%
% Program for the derivation of the channel parameters of the
% Doppler PSDs defined by COST 207.
%
%-------------------------------------------------------------------
%[f1,f2,c1,c2,th1,th2,rho,f_rho,f01,f02]=pCOST207(D_S_T,N_i)
%-------------------------------------------------------------------
% Explanation of the input parameters:
%
% D_S_T: type of the Doppler PSD:
%             Jakes:    D_S_T='JA'
%             Rice:     D_S_T='RI'
%             Gauss  I: D_S_T='G1'
%             Gauss II: D_S_T='G2'
% N_i: number of harmonic functions

function [f1,f2,c1,c2,th1,th2,rho,f_rho,f01,f02]=pCOST207(D_S_T,N_i)

if     all(lower(D_S_T)=='ri'), % RICE
       n=(1:N_i);
       f1=sin(pi/(2*N_i)*(n-1/2));
       c1=0.41*sqrt(1/N_i)*ones(1,N_i);
       th1=rand(1,N_i)*2*pi;
       n=(1:N_i+1);
       f2=sin(pi/(2*(N_i+1))*(n-1/2));
       c2=0.41*sqrt(1/(N_i+1))*ones(1,N_i+1);
       th2=rand(1,N_i+1)*2*pi;
```

```
        f01=0;f02=0;
        rho=0.91;f_rho=0.7;
elseif all(lower(D_S_T)=='ja'), % JAKES
        n=(1:N_i);
        f1=sin(pi/(2*N_i)*(n-1/2));
        c1=sqrt(1/N_i)*ones(1,N_i);
        th1=rand(1,N_i)*2*pi;
        n=(1:N_i+1);
        f2=sin(pi/(2*(N_i+1))*(n-1/2));
        c2=sqrt(1/(N_i+1))*ones(1,N_i+1);
        th2=rand(1,N_i+1)*2*pi;
        f01=0;f02=0;
        rho=0;f_rho=0;
elseif all(lower(D_S_T)=='g1'), % GAUSS I
        n=(1:N_i);
        sgm_0_2=5/6;
        c1=sqrt(sgm_0_2*2/N_i)*ones(1,N_i);
        f1=sqrt(2)*0.05*erfinv((2*n-1)/(2*N_i));
        th1=rand(1,N_i)*2*pi;
        sgm_0_2=1/6;
        c2=[sqrt(sgm_0_2*2/N_i)*ones(1,N_i),0]/j;
        f2=[sqrt(2)*0.1*erfinv((2*n-1)/(2*N_i)),0];
        th2=[rand(1,N_i)*2*pi,0];
        f01=0.8;f02=-0.4;
        rho=0;f_rho=0;
elseif all(lower(D_S_T)=='g2'), % GAUSS II
        n=(1:N_i);
        sgm_0_2=10^0.5/(sqrt(10)+0.15);
        c1=sqrt(sgm_0_2*2/N_i)*ones(1,N_i);
        f1=sqrt(2)*0.1*erfinv((2*n-1)/(2*N_i));
        th1=rand(1,N_i)*2*pi;
        sgm_0_2=0.15/(sqrt(10)+0.15);
        c2=[sqrt(sgm_0_2*2/N_i)*ones(1,N_i),0]/j;
        f2=[sqrt(2)*0.15*erfinv((2*n-1)/(2*N_i)),0];
        th2=[rand(1,N_i)*2*pi,0];
        f01=-0.7;f02=0.4;
        rho=0;f_rho=0;
end
```

APPENDIX 2 - MATLAB PROGRAMS FOR MOBILE FADING CHANNEL SIMULATORS

MATLAB-PROGRAMS of the fundamental channel simulators

```
%-------------------------------------------------------------------
% Mu_i_t.m ---------------------------------------------------------
%
% Program for the simulation of real deterministic Gaussian processes
%-------------------------------------------------------------------
% mu_i_t=Mu_i_t(c,f,th,T_s,T_sim,PLOT)
%-------------------------------------------------------------------
% Explanation of the input parameters:
%
% f:  discrete Doppler frequencies
% c:  Doppler coefficients
% th: Doppler phases
% T_s: sampling interval
% T_sim: duration of the simulation
% PLOT: plot of the deterministic Gaussian process mu_i(t),
%       if PLOT==1

function mu_i_t=Mu_i_t(c,f,th,T_s,T_sim,PLOT)

if nargin==5,
   PLOT=0;
end

N=ceil(T_sim/T_s);
t=(0:N-1)*T_s;
mu_i_t=0;
for k=1:length(f),
    mu_i_t=mu_i_t+c(k)*cos(2*pi*f(k)*t+th(k));
end

if PLOT==1,
   plot(t,mu_i_t)
   set(0,'DefaultAxesFontSize',16);
   xlabel({'t (s)'},'FontName','Arial','FontSize',24,'Interpreter','latex')
   ylabel({'$\mu_i$ (t)'},'FontName','Arial','FontSize',...
      24,'Interpreter','latex')
 end

%-------------------------------------------------------------------
% Rice_proc.m ------------------------------------------------------
%
% Program for the simulation of deterministic Rice processes xi(t)
%
% Used m-file: Mu_i_t.m
%-------------------------------------------------------------------
% xi_t=Rice_proc(f1,c1,th1,f2,c2,th2,rho,f_rho,theta_rho,...
%                                                    T_s,T_sim,PLOT)
%-------------------------------------------------------------------
% Explanation of the input parameters:
%
% f1, c1, th1: discrete Doppler frequencies, Doppler coefficients,
%              and Doppler phases of mu_1(t)
% f2, c2, th2: discrete Doppler frequencies, Doppler coefficients,
%              and Doppler phases of mu_2(t)
```

```
% rho: amplitude of the LOS component m(t)
% f_rho: Doppler frequency of the LOS component m(t)
% theta_rho: phase of the LOS component m(t)
% T_s: sampling interval
% T_sim: duration of the simulation
% PLOT: plot of the deterministic Rice process xi(t), if PLOT==1

function xi_t=Rice_proc(f1,c1,th1,f2,c2,th2,rho,f_rho,theta_rho,...
                        T_s,T_sim,PLOT)

if nargin==10,
   PLOT=0;
end

N=ceil(T_sim/T_s);
t=(0:N-1)*T_s;
arg=2*pi*f_rho*t+theta_rho;

xi_t=abs(Mu_i_t(c1,f1,th1,T_s,T_sim)+rho*cos(arg)+...
         j*(Mu_i_t(c2,f2,th2,T_s,T_sim)+rho*sin(arg)) );

if PLOT==1,
   plot(t,20*log10(xi_t))
   set(0,'DefaultAxesFontSize',16)
   xlabel('$t(s)$','Interpreter','latex','FontSize',24)
   ylabel('20 log $x_i$ (t)','Interpreter','latex','FontSize',24)
end

%-------------------------------------------------------------------
% gen_Rice_proc.m ---------------------------------------------------
%
% Program for the simulation of deterministic generalized Rice
% processes
%
% Used m-files: parameter_Jakes.m, Mu_i_t.m
%-------------------------------------------------------------------
% xi_t=gen_Rice_proc(N_1,N_2,sigma_1_2,sigma_2_2,kappa_0,...
%                    theta_0,rho,theta_rho,f_max,...
%                    T_s,T_sim,PLOT)
%-------------------------------------------------------------------
% Explanation of the input parameters:
%
% N_1, N_2: number of harmonic functions of the real deterministic
%           Gaussian processes nu_1(t) and nu_2(t), respectively
% sigma_1_2: average power of the real deterministic Gaussian
%            process nu_1(t)
% sigma_2_2: average power of the real deterministic Gaussian
%            process nu_2(t)
% kappa_0: frequency ratio f_min/f_max (0<=kappa_0<=1)
% theta_0: phase shift between mu_1_n(t) and mu_2_n(t)
% rho: amplitude of the LOS component m(t)
% theta_rho: phase of the LOS component m(t)
% f_max: maximum Doppler frequency
% T_s: sampling interval
% T_sim: duration of the simulation
% K: number of sets (partitions)
% PLOT: plot of the deterministic generalized Suzuki process xi(t),
%       if PLOT==1

function xi_t=gen_Rice_proc(N_1,N_2,sigma_1_2,sigma_2_2,kappa_0,...
```

```
                              theta_0,rho,theta_rho,f_max,T_s,...
                              T_sim,K,PLOT)

if nargin==11,
   PLOT=0;
end

[f1,c1,th1]=parameter_Jakes('es_j',N_1,sigma_1_2,f_max,'rand',K,0);
c1=c1/sqrt(2);

N_2_s=ceil(N_2/(2/pi*asin(kappa_0)));
[f2,c2,th2]=parameter_Jakes('es_j',N_2_s,sigma_2_2,f_max,'rand',K,0);
f2 =f2(1:N_2);
c2 =c2(1:N_2)/sqrt(2);
th2=th2(1:N_2);

N=ceil(T_sim/T_s);
t=(0:N-1)*T_s;

xi_t=abs(Mu_i_t(c1,f1,th1,T_s,T_sim)+...
         Mu_i_t(c2,f2,th2,T_s,T_sim)+rho*cos(theta_rho)+...
         j*(Mu_i_t(c1,f1,th1-theta_0,T_s,T_sim)+...
         Mu_i_t(c2,f2,th2+theta_0,T_s,T_sim)+...
         rho*sin(theta_rho)));

if PLOT==1,
   plot(t,20*log10(xi_t),'b-')
   set(0,'DefaultAxesFontSize',16);
   xlabel({'t (s)'},'FontName','Arial','FontSize',24,'Interpreter','latex')
   ylabel({'20 log($x_i$
(t))'},'FontName','Arial','FontSize',24,'Interpreter','latex')
end

%-----------------------------------------------------------------------
% Suzuki_Type_I.m ------------------------------------------------------
%
% Program for the simulation of deterministic extended Suzuki
% processes of Type I
%
% Used m-files: parameter_Jakes.m, parameter_Gauss.m, Mu_i_t.m
%-----------------------------------------------------------------------
% eta_t=Suzuki_Type_I(N_1,N_2,N_3,sigma_0_2,kappa_0,f_max,sigma_3,...
%                     m_3,rho,f_rho,theta_rho,f_c,T_s,T_sim,PLOT)
%-----------------------------------------------------------------------
% Explanation of the input parameters:
%
% N_1, N_2, N_3: number of harmonic functions of the real deter-
%                ministic Gaussian processes nu_1(t), nu_2(t),
%                and nu_3(t), respectively
% sigma_0_2: average power of the real deterministic Gaussian
%            processes mu_1(t) and mu_2(t)
% kappa_0: frequency ratio f_min/f_max (0<=kappa_0<=1)
% f_max: maximum Doppler frequency
% sigma_3: square root of the average power of the real deterministic
%          Gaussian process nu_3(t)
% m_3: average value of the third real deterministic Gaussian
%      process mu_3(t)
% rho: amplitude of the LOS component m(t)
% f_rho: Doppler frequency of the LOS component m(t)
% theta_rho: phase of the LOS component m(t)
```

```
% f_c: 3-dB-cut-off frequency
% T_s: sampling interval
% T_sim: duration of the simulation
% PLOT: plot of the deterministic extended Suzuki process eta(t) of
%       Type I, if PLOT==1

function eta_t=Suzuki_Type_I(N_1,N_2,N_3,sigma_0_2,kappa_0,f_max,...
               sigma_3,m_3,rho,f_rho,theta_rho,f_c,T_s,T_sim,PLOT)

if nargin==14,
   PLOT=0;
end

[f1,c1,th1]=parameter_Jakes('es_j',N_1,sigma_0_2,f_max,'rand',0);
c1=c1/sqrt(2);

N_2_s=ceil(N_2/(2/pi*asin(kappa_0)));
[f2,c2,th2]=parameter_Jakes('es_j',N_2_s,sigma_0_2,f_max,'rand',0);
f2 =f2(1:N_2);
c2 =c2(1:N_2)/sqrt(2);
th2=th2(1:N_2);

[f3,c3,th3]=parameter_Gauss('es_g',N_3,1,f_max,f_c,'rand',0);
gaMma=(2*pi*f_c/sqrt(2*log(2)))^2;
f3(N_3)=sqrt(gaMma*N_3/(2*pi)^2-sum(f3(1:N_3-1).^2));

N=ceil(T_sim/T_s);
t=(0:N-1)*T_s;

arg=2*pi*f_rho*t+theta_rho;

xi_t=abs(Mu_i_t(c1,f1,th1,T_s,T_sim)+...
         Mu_i_t(c2,f2,th2,T_s,T_sim)+rho*cos(arg)+...
         j*(Mu_i_t(c1,f1,th1-pi/2,T_s,T_sim)-...
         Mu_i_t(c2,f2,th2-pi/2,T_s,T_sim)+rho*sin(arg)));
lambda_t=exp(Mu_i_t(c3,f3,th3,T_s,T_sim)*sigma_3+m_3);

eta_t=xi_t.*lambda_t;

if PLOT==1,
   plot(t,20*log10(eta_t),'b-')
   set(0,'DefaultAxesFontSize',16)
   xlabel('$t(s)$','Interpreter','latex','FontSize',24)
   ylabel('20 log $\eta$ (t)','Interpreter','latex','FontSize',24)
end

%-----------------------------------------------------------------------
% Suzuki_Type_II.m ------------------------------------------------------
%
% Program for the simulation of deterministic extended Suzuki
% processes of Type II
%
% Used m-files: parameter_Jakes.m, parameter_Gauss.m, Mu_i_t.m
%-----------------------------------------------------------------------
% eta_t=Suzuki_Type_II(N_1,N_3,sigma_0_2,kappa_0,theta_0,f_max,...
%                  sigma_3,m_3,rho,theta_rho,f_c,T_s,T_sim,PLOT)
%-----------------------------------------------------------------------
% Explanation of the input parameters:
%
```

```
% N_1, N_3: number of harmonic functions of the real deterministic
%           Gaussian processes nu_0(t) and nu_3(t), respectively
% sigma_0_2: average power of the real deterministic Gaussian
%            process mu_0(t)  (for kappa_0=1)
% kappa_0: frequency ratio f_min/f_max (0<=kappa_0<=1)
% theta_0: phase shift between mu_1_n(t) and mu_2_n(t)
% f_max: maximum Doppler frequency
% sigma_3: square root of the average power of the real deterministic
%          Gaussian process nu_3(t)
% m_3: average value of the real deterministic Gaussian
%      process mu_3(t)
% rho: amplitude of the LOS component m(t)
% theta_rho: phase of the LOS component m(t)
% f_c: 3-dB-cut-off freqquency
% T_s: sampling interval
% T_sim: duration of the simulation
% PLOT: plot of the deterministic extended Suzuki process eta(t) of
%       Type II, if PLOT==1

function eta_t=Suzuki_Type_II(N_1,N_3,sigma_0_2,kappa_0,theta_0,...
                              f_max,sigma_3,m_3,rho,theta_rho,f_c,...
                              T_s,T_sim,PLOT)
if nargin==13,
   PLOT=0;
end

N_1_s=ceil(N_1/(2/pi*asin(kappa_0)));
[f1,c1,th1]=parameter_Jakes('es_j',N_1_s,sigma_0_2,f_max,'rand',0);
f1 =f1(1:N_1);
c1 =c1(1:N_1);
th1=th1(1:N_1);

[f3,c3,th3]=parameter_Gauss('es_g',N_3,1,f_max,f_c,'rand',0);
gaMma=(2*pi*f_c/sqrt(2*log(2)))^2;
f3(N_3)=sqrt(gaMma*N_3/(2*pi)^2-sum(f3(1:N_3-1).^2));

N=ceil(T_sim/T_s);
t=(0:N-1)*T_s;

xi_t=abs(Mu_i_t(c1,f1,th1,T_s,T_sim)+rho*cos(theta_rho)+...
         j*(Mu_i_t(c1,f1,th1-theta_0,T_s,T_sim)+...
            rho*sin(theta_rho) ) );

lambda_t=exp(Mu_i_t(c3,f3,th3,T_s,T_sim)*sigma_3+m_3);

eta_t=xi_t.*lambda_t;


if PLOT==1,
   plot(t,20*log10(eta_t),'b-')
   set(0,'DefaultAxesFontSize',16)
   xlabel('t (s)','Interpreter','latex','FontSize',24)
   ylabel('20 log $\eta$ (t)',,'Interpreter','latex','FontSize',24)
end

%-----------------------------------------------------------------
% det_mod_Loo.m ---------------------------------------------------
%
% Program for the simulation of modified Loo processes.
```

```
%
% Used m-files: parameter_Jakes.m, parameter_Gauss.m, Mu_i_t.m
%-------------------------------------------------------------------
% rho_t=det_mod_Loo(N_1,N_2,N_3,sigma_1_2,kappa_1,sigma_2_2,...
%                   kappa_2,f_max,sigma_3,m_3,f_rho,...
%                   theta_rho,f_c,T_s,T_sim,PLOT)
%-------------------------------------------------------------------
% Explanation of the input parameters:
%
% N_1, N_2, N_3: number of harmonic functions of the real determi-
%                nistic Gaussian processes nu_1(t), nu_2(t), and
%                nu_3(t), respectively
% sigma_1_2: average power of the real deterministic Gaussian
%            process nu_1(t)
% kappa_1: frequency ratio f_min/f_max (0<=kappa_0<=1) of nu_1(t)
% sigma_2_2: average power of the real deterministic Gaussian
%            process nu_2(t)
% kappa_2: frequency ratio f_min/f_max (0<=kappa_0<=1) of nu_2(t)
% f_max: maximum Doppler frequency
% sigma_3: square root of the average power of the real deterministic
%          Gaussian process nu_3(t)
% m_3: average value of the third real deterministic Gaussian
%      process mu_3(t)
% f_rho: Doppler frequency of the LOS component m(t)
% theta_rho: phase of the LOS component m(t)
% f_c: 3-dB-cut-off frequency
% T_s: sampling interval
% T_sim: duration of the simulation
% PLOT: plot of the time-domain signal rho(t), if PLOT==1

function rho_t=det_mod_Loo(N_1,N_2,N_3,sigma_1_2,kappa_1,...
               sigma_2_2,kappa_2,f_max,sigma_3,m_3,f_rho,...
               theta_rho,f_c,T_s,T_sim,PLOT)

if nargin==15,
   PLOT=0;
end

sigma_1=sqrt(sigma_1_2);
sigma_2=sqrt(sigma_2_2);

N_1_s=ceil(N_1/(2/pi*asin(kappa_1)));
[f1,c1,th1]=parameter_Jakes('es_j',N_1_s,sigma_1_2,f_max,'rand',0);
f1 =f1(1:N_1);
c1 =c1(1:N_1)/sqrt(2);
th1=th1(1:N_1);

N_2_s=ceil(N_2/(2/pi*asin(kappa_2)));
[f2,c2,th2]=parameter_Jakes('es_j',N_2_s,sigma_2_2,f_max,'rand',0);
f2 =f2(1:N_2);
c2 =c2(1:N_2)/sqrt(2);
th2=th2(1:N_2);

[f3,c3,th3]=parameter_Gauss('es_g',N_3,1,f_max,f_c,'rand',0);
gaMma=(2*pi*f_c/sqrt(2*log(2)))^2;
f3(N_3)=sqrt(gaMma*N_3/(2*pi)^2-sum(f3(1:N_3-1).^2));

N=ceil(T_sim/T_s);
t=(0:N-1)*T_s;
```

```
arg=2*pi*f_rho*t+theta_rho;

RHO_t=exp(Mu_i_t(c3,f3,th3,T_s,T_sim)*sigma_3+m_3);

rho_t=abs(Mu_i_t(c1,f1,th1,T_s,T_sim)+...
          Mu_i_t(c2,f2,th2,T_s,T_sim)+RHO_t.*cos(arg)+...
          j*(Mu_i_t(c1,f1,th1-pi/2,T_s,T_sim)-...
          Mu_i_t(c2,f2,th2-pi/2,T_s,T_sim)+RHO_t.*sin(arg)));

if PLOT==1,
   plot(t,20*log10(rho_t),'b-',t,20*log10(RHO_t),'y--')
   set(0,'DefaultAxesFontSize',16);
   xlabel('t (s)','FontName','Arial','FontSize',24,'Interpreter','latex')
   ylabel('20 log $\rho$
(t)','FontName','Arial','FontSize',24,'Interpreter','latex')
end

%-----------------------------------------------------------------------
% F_S_K.m ------------------------------------------------------------
%
% Program for the simulation of deterministic frequency-selective
% mobile radio channels.
%
%-----------------------------------------------------------------------
% [y_t,T,t_0]=F_S_K(x_t,f_max,m_s,T,t_0,q_l,...
%                        C1,F1,TH1,C2,F2,TH2,F01,F02,RHO,F_RHO,PLOT)
%-----------------------------------------------------------------------
% Explanation of the input parameters:
%
% x_t: time-domain input signal of the channel simulator (sampled
%      with T_s=0.2E-6 s)
% f_max: maximum Doppler frequency
% m_s: sampling rate ratio
% T: contents of the delay elements of the time variant FIR filter
% t_0: offset in time
% q_l: q_l=tau_l/T_s+1
%-----------------------------------------------------------------------
% The following matrices are generated in F_S_K_p.m:
% F1, F2: discrete Doppler frequencies
% C1, C2: Doppler coefficients
% TH1, TH2: Doppler phases
% F01, F02: frequency shift value of the Doppler PSD according to
%           Gauss I and Gauss II, respectively
% RHO: amplitude of the direct component
% F_RHO: Doppler frequency of the direct component
%-----------------------------------------------------------------------
% PLOT: plot of the output signal of the channel, if PLOT==1

function [y_t,T,t_0]=F_S_K(x_t,f_max,m_s,T,t_0,q_l,...
                      C1,F1,TH1,C2,F2,TH2,F01,F02,RHO,F_RHO,PLOT)

T_s=0.2E-6;

% Initialization:
mu_l=zeros(size(q_l));
y_t=zeros(size(x_t));

for n=0:length(x_t)-1,
    if rem(n/m_s,m_s)-fix(rem(n/m_s,m_s))==0,
       mu_l=sum((C1.*cos(2*pi*F1*f_max*(n*T_s+t_0)+TH1)).').*...
```

```
            exp(-j*2*pi*F01*f_max*(n*T_s+t_0))+j*...
            (sum((C2.*cos(2*pi*F2*f_max*(n*T_s+t_0)+TH2)).').*...
            exp(-j*2*pi*F02*f_max*(n*T_s+t_0)))+...
            RHO.*exp(j*2*pi*F_RHO*f_max*(n*T_s+t_0));
    end
    T(1)=x_t(n+1);
    y_t(n+1)=sum(mu_l.*T(q_l));
    T(2:length(T))=T(1:length(T)-1);
end

t_0=length(x_t)*T_s+t_0;

if PLOT==1,
   plot((0:length(y_t)-1)*T_s,20*log10(abs(y_t)),'g-')
end
```

MATLAB-PROGRAMS of the spatial shadowing simulator

```
%--------------------------------------------------------------------------
% shadowing_parameters.m ------------------------------------------------
%
% Program for the computation of the gains and spatial frequencies
% of a spatial shadowing simulation using the modified method of
% equal areas (MMEA).
%
% Used m-files: fun_butterworth.m,acf_mue.m
%--------------------------------------------------------------------------
% [c_n,alpha_n,sigma_L,mL,D]=shadowing_parameters(N,envir_type,model_type,
% PLOT)
%--------------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: number of sinusoids
% envir_type:
% |-----------------------------------------------|------------------|
% | Shadowing areas                               |      Input       |
% |-----------------------------------------------|------------------|
% |-----------------------------------------------|------------------|
% | Urban area                                    |     'Urban'      |
% |-----------------------------------------------|------------------|
% | Suburban area                                 |     'Surbn'      |
% |-----------------------------------------------|------------------|
% model_type:
% |-----------------------------------------------|------------------|
% | Correlation models                            |      Input       |
% |-----------------------------------------------|------------------|
% |-----------------------------------------------|------------------|
% | Gudmundson model                              |     'Gudm'       |
% |-----------------------------------------------|------------------|
% | Gaussian model                                |     'Gaus'       |
% |-----------------------------------------------|------------------|
% | Butterworth model                             |     'Butw'       |
% |-----------------------------------------------|------------------|
% PLOT: plot of the resulting autocrrelation function r_vv(x),
%       if PLOT==1
```

```
function
[c_n,alpha_n,sigma_L,mL,D]=shadowing_parameters(N,envir_type,model_type,PLO
T)
if nargin==3,
    PLOT=0;
end
x=(1:N).';
T_a=0.01;
if envir_type == 'Urban'
    D=8.3058;                   % decorrelation distance
    delta_max=40;               % deltax_max=x2-x1
    sigma_L=4.3;                % standard deviation
    mL=0;                       % area mean
% computation of the parameters of the structure of spatial shadowing
% according to the model's type
    switch model_type
        case 'Butw'
            alpha_n=zeros(N,1);
            for i=1:N
            alpha_n(i)=fzero(@(x) fun_butterworth(i,N,x ,D),1);
            end
            c_n=sqrt(2./N).*ones(N,1);
            % computation of the simulation of ACF
            delta = [0:T_a:delta_max];
            r_vv=acf_mue(alpha_n,c_n,delta);
            % computation of the acf according to Butterworth's model
            D2=1.2396./(sqrt(2)*pi*D);
            z=sqrt(2)*pi*D2.*delta;
            R_r = exp(-z).*(cos(z)+sin(z));
        case 'Gaus'
            c_n=sqrt(2./N).*ones(N,1);
            alpha_n=erfinv((x-0.5)./N)./(D*pi);
            % computation of the simulation of ACF
            delta = [0:T_a:delta_max];
            r_vv=acf_mue(alpha_n,c_n,delta);
            % computation of the acf according to Gaussian's model
            R_r = exp(-(delta/D).^2);
        case 'Gudm'
            alpha_n=1./(2.*pi.*D).*tan(pi.*(x.'-1./2)./(2*N));
            c_n=sqrt(2./N).*ones(N,1);
            % computation of the simulation of ACF
            delta = [0:T_a:delta_max];
            r_vv=acf_mue(alpha_n,c_n,delta);
            % computation of the acf according to Gudmundson's model
            R_r = exp(-abs(delta)/D);
        otherwise
            disp('Please enter the right model type(Butw,Gaus or Gudm)!!');
        return;
    end
elseif envir_type == 'Surbn'
    D=503.9;                    % decorrelation distance
    delta_max=2500;             % deltax_max=x2-x1
    sigma_L=7.5;                % standard deviation
    mL=0;                       % area mean
% computation of the parameters of the structure of spatial shadowing
% according to the model's type
    switch model_type
        case 'Butw'
            alpha_n=zeros(N,1);
            for i=1:N
            alpha_n(i)=fzero(@(x) fun_butterworth(i,N,x ,D),1);
```

```
            end
            c_n=sqrt(2./N).*ones(N,1);
            % computation of the simulation of ACF
            delta = [0:100*T_a:delta_max];
            r_vv=acf_mue(alpha_n,c_n,delta);
            % computation of the acf according to Butterworth's model
            D2=1.2396./(sqrt(2)*pi*D);
            z=sqrt(2)*pi*D2.*delta;
            R_r = exp(-z).*(cos(z)+sin(z));
        case 'Gaus'
            c_n=sqrt(2./N).*ones(N,1);
            alpha_n=erfinv((x-0.5)./N)./(D*pi);
            % computation of the simulation of ACF
            delta = [0:T_a:delta_max];
            r_vv=acf_mue(alpha_n,c_n,delta);
            % computation of the acf according to Gaussian's model
            R_r = exp(-(delta/D).^2);
        case 'Gudm'
            alpha_n=1./(2.*pi.*D).*tan(pi.*(x.'-1./2)./(2*N));
            c_n=sqrt(2./N).*ones(N,1);
            % computation of the simulation of ACF
            delta = [0:100*T_a:delta_max];
            r_vv=acf_mue(alpha_n,c_n,delta);
            % computation of the acf according to Gudmundson's model
            R_r = exp(-(delta/D).^2);
        otherwise
            disp('Please enter the right model type(Butw,Gaus or Gudm)!!');
        return;
    end
else
    disp('Please enter the right environment type(Urban or Surbn)!!')
    return;
end
if PLOT==1,
    plot(delta, R_r,'k-');
    hold on;
    it=10;
    L=length(delta);
    plot(delta(1:it:L),r_vv(1:it:L), 'ko', 'MarkerSize',4);
    set(0,'DefaultAxesFontSize',16);
    xlabel('Spatial separation, $\Delta x$ (m)',
'FontSize',24,'Interpreter','latex');
    ylabel('Spatial ACF, $\tilde{r}_{\nu\nu}$ $(\Delta x)$',
'FontSize',24,'Interpreter','latex');
    legend({['simulation model'],['simulation']},...

'FontName','Arial','Location','NorthEast','FontSize',22,'Interpreter','late
x');
    axis([0 delta_max -0.2 1.2]);
end

%--------------------------------------------------------------------------
% fun_butterworth.m -------------------------------------------------------
%
% Computation of the equation relatively to Butterworth model of order 2
% that should be resolved in order to find the spatial frequencies alpha_n
%
%--------------------------------------------------------------------------
% [y]=fun_butterworth(n,N,x,D)
%--------------------------------------------------------------------------
% Explanation of the input parameters:
```

```
%
% n: iteration index
% N: number of the sinusoids
% x: arbitrary vector which should be as longer as N
% D: decorrelation distance

function [y]=fun_butterworth(n,N,x,D)

D1=1.2396./(sqrt(2)*pi*D);
y1=linspace(0,x./D1,10000);
y=trapz(y1,1./(1+y1.^4))-(n-0.5)*pi./2./N./sqrt(2);



%-------------------------------------------------------------------------
% shadowing_processes.m ------------------------------------------------
%
% Program for the simulation of deterministic log-normal processes
%
% Used m-file: shadowing_parameters.m
%-------------------------------------------------------------------------
% [x,lamda_x]=shadowing_processes(N,T_s,N_s,N_processes,envir_type,
% model_type,PLOT)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: number of sinusoids
% T_s: sampling interval
% N_s: number of samples per process
% N_processes: number of samples functions
% envir_type:
% |-----------------------------------------|------------------|
% | Shadowing areas                         |      Input       |
% |-----------------------------------------|------------------|
% |-----------------------------------------|------------------|
% | Urban area                              |     'Urban'      |
% |-----------------------------------------|------------------|
% | Suburban area                           |     'Surbn'      |
% |-----------------------------------------|------------------|
% model_type:
% |-----------------------------------------|------------------|
% | Correlation models                      |      Input       |
% |-----------------------------------------|------------------|
% |-----------------------------------------|------------------|
% | Gudmundson model                        |     'Gudm'       |
% |-----------------------------------------|------------------|
% | Gaussian model                          |     'Gaus'       |
% |-----------------------------------------|------------------|
% | Butterworth model                       |     'Butw'       |
% |-----------------------------------------|------------------|
% % PLOT: plot of the resulting shadowing process xi_t,
%       if PLOT==1

function
[x,lamda_x]=shadowing_processes(N,T_s,N_s,N_processes,envir_type,...
            model_type,PLOT)
[c_n,alpha_n,sigma_L,mL,D]=shadowing_parameters(N,envir_type,model_type);
if nargin==6,
   PLOT=0;
end
T_sim=N_s*T_s;
```

```
x=0:T_s:T_sim;
lamda_x=[];
for i=1:N_processes,
   theta=(rand(N,1)*2-1).*pi;
   v_x=zeros(1,length(x));
   for i=1:N,
      v_x=c_n(i).*cos(2.*pi.*alpha_n(i).*x+theta(i))+v_x;
end
lamda_x=[lamda_x;10.^((sigma_L.*v_x+mL)./20)];
end
if PLOT==1,
    semilogy(x,lamda_x(1,:));
    xlabel({'Spatial distance, $x$'},'FontName','Arial','FontSize',...
      24,'Interpreter','latex');
    ylabel({'Spatial Shadowing model, $V(x)$'},'FontName',...
      'Arial','FontSize',24,'Interpreter','latex');
end
```

MATLAB-PROGRAMS of MIMO one-ring model simulator

```
%-------------------------------------------------------------------------
% MIMO_one_ring_parameters.m-----------------------------------------------
%
% Program for the computation of angle of arrivals phi of MIMO one-ring
%  model using LPNM method and Modified method of equal areas
%
% Used files: data_one_ring.mat, errorfunc_one_ring.m
%-------------------------------------------------------------------------
% [phi]=MIMO_one_ring_parameters(N,alphaBS,alphaMS,phimax,kappa,f_max,phi_0
% ,alphav,p,method,PLOT)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: number of scatterers
% alphaBS: BS antenna tilt angle in degre
% alphaMS: MS antenna tilt angle in degre
% phimax: one half of the maximum angle of departure in degre
% kappa: parameter used in Mises density
% f_max: maximum doppler frequency
% phi_0: the angle of rotation in degre
% alphav: the angle of motion in degre
% p: the power of LPNM method
% method: parameters computation method that can be:
%       1) Modified MEA:    'MMEA'
%       2) Lp-norm method:  'LPNM'
% PLOT: if PLOT==1, plot of the Time ACF and 2D-space CCF of the reference
% model as well as the simulation model

function [phi]=MIMO_one_ring_parameters(N,alphaBS,alphaMS,phimax,...
            kappa,f_max,phi_0,alphav,p,method,PLOT)
if nargin<11
    PLOT=0;
end
% converting parameters from degre to radium
alphaBS=2*pi*alphaBS/360;
alphaMS=2*pi*alphaMS/360;
phimax=2*pi*phimax/360;
alphav=2*pi*alphav/360;
```

```
phi_0=2*pi*phi_0/360;
save data_one_ring.mat N alphaBS alphaMS phimax kappa f_max phi_0 alphav p;
% computation of the parameters phi using MMEA
if all(upper(method)=='MMEA')
    phi=zeros(N,1);
    for n=1:N
        phi(n)=fzero(@(x) fun_phi(n,N,kappa,x,phi_0),pi);
    end
elseif all(upper(method)=='LPNM')
    % computation of initial values for the parameter phi using MMEA
    phi=zeros(N,1);
    for n=1:N
        phi(n)=fzero(@(x) fun_phi(n,N,kappa,x,phi_0),pi);
    end
    % determination of the phases phi through LPNM method
    options = optimset('Display','iter','MaxIter',100,'TypicalX',phi);
    x=fminsearch('errorfunc_one_ring',phi,options);
    phi=x;
else
    disp('enter a valid input method belonging to{MMEA,LPNM}');
end

% testing the correctness of the simulation model by determining the time
% ACF for the simulation model as well as the reference model
tau=linspace(0,N/(2*f_max),8*N);
phiMS=linspace(-pi,pi,4*N);
for k=1:length(tau)
    ACF(k)=trapz(phiMS,exp(-sqrt(-1)*2*pi*f_max*...
      cos(phiMS-alphav)*tau(k)).*Mises(phiMS,kappa,phi_0));
    ACF_teld(k)=sum(exp(-sqrt(-1).*2.*pi.*f_max.*...
      cos(phi-alphav).*tau(k))./length(phi);
end

% testing the correctness of the simulation model by determining the 2D-
space
% CCF for the simulation model as well as the reference model
deltaBS=linspace(0,N/4,4*N);
deltaMS=linspace(0,N/4,4*N);
phiMS=linspace(-pi,pi,4*N);
CCF_teld=zeros(length(deltaBS),length(deltaMS));
CCF=zeros(length(deltaBS),length(deltaMS));
for q=1:length(deltaBS)
    for g=1:length(deltaMS)
        a=exp(sqrt(-1).*pi.*deltaBS(q).*(cos(alphaBS)+...
            phimax.*sin(alphaBS).*sin(phiMS)));
        b=exp(sqrt(-1).*pi.*deltaMS(g).*cos(phiMS-alphaMS));
        CCF(q,g)=trapz(phiMS,a.^2.*b.^2.*Mises(phiMS,kappa,phi_0));
        aa=exp(sqrt(-1).*pi.*deltaBS(q).*(cos(alphaBS)...
            +phimax.*sin(alphaBS).*sin(phi)));
        bb=exp(sqrt(-1).*pi.*deltaMS(g).*cos(phi-alphaMS));
        CCF_teld(q,g)=sum(aa.^2.*bb.^2)/length(phi);
    end
end
Err=CCF-CCF_teld;
if PLOT==1
    subplot(2,2,1);
    grid on;
    plot(tau*f_max,real(ACF));
    hold on;
    plot(tau*f_max,real(ACF_teld),'r*')
    xlim([0 N/2])
```

```
    ylim([-1 1])
    set(0,'DefaultAxesFontSize',16);
    legend({['Reference model'],['Simulation model ...
            with N = '  ,num2str(N)]},...
            'FontName','Arial','Location','NorthEast',...
            'FontSize',16,'Interpreter','latex');
    xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    ylabel({'Time ACF'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    subplot(2,2,2);
    surf(deltaBS,deltaMS,real(CCF));
    zlim([-1 1])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
    title({'Reference model'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    zlabel({'2D space CCF'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    xlabel({'$\delta_{BS}$ / $\lambda$'},'FontName','Arial','FontSize',...
            20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{MS}$ / $\lambda$'},'FontName','Arial','FontSize',...
            20,'Rotation',-20,'Interpreter','latex');
    subplot(2,2,3);
    surf(deltaBS,deltaMS,real(CCF_teld));
    zlim([-1 1])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
    title({'Simulation model'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    zlabel({'2D space CCF'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    xlabel({'$\delta_{BS}$ / $\lambda$'},'FontName','Arial','FontSize',...
            20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{MS}$ / $\lambda$'},'FontName','Arial','FontSize',...
            20,'Rotation',-20,'Interpreter','latex');
    subplot(2,2,4);
    surf(deltaBS,deltaMS,real(Err));
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
    zlabel({'Error function'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    xlabel({'$\delta_{BS}$ / $\lambda$'},'FontName','Arial','FontSize',...
            20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{MS}$ / $\lambda$'},'FontName','Arial','FontSize',...
            20,'Rotation',-20,'Interpreter','latex');
end

%-------------------------------------------------------------------------
% errorfunc_one_ring.m----------------------------------------------------
%
% Program for the computation of the error function of the reference
% model and the simulation model that will be used in LPNM method
%
% used files: data_one_ring.mat
%-------------------------------------------------------------------------
% [E]=errorfunc_one_ring(phi)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% phi: angle of arrival phi
```

```
function [E]=errorfunc_one_ring(phi)
load data_one_ring.mat;

% The error function between time ACF reference and time ACF simulation
tau=linspace(0,N/(4*f_max),12*N);
phiMS=linspace(-pi,pi,6*N);
for k=1:length(tau)
    ACF(k)=trapz(phiMS,exp(-j*2*pi*f_max*cos(phiMS-
alphav)*tau(k)).*Mises(phiMS,kappa,phi_0));
    ACF_teld(k)=sum(exp(-j*2*pi*f_max*cos(phi-alphav)*tau(k)))/length(phi);
end
% The error function between space CCF reference and space CCF simulation
deltaBS=linspace(0,N/4,N);
deltaMS=linspace(0,N/4,N);
phiMS=linspace(-pi,pi,4*N);
for q=1:length(deltaBS)
    for g=1:length(deltaMS)
        a=exp(sqrt(-1).*pi.*deltaBS(q).*(cos(alphaBS)+phimax.*...
            sin(alphaBS).*sin(phiMS)));
        b=exp(sqrt(-1).*pi.*deltaMS(g).*cos(phiMS-alphaMS));
        CCF(q,g)=trapz(phiMS,a.^2.*b.^2.*Mises(phiMS,kappa,phi_0));
        aa=exp(sqrt(-1).*pi.*deltaBS(q).*(cos(alphaBS)+phimax.*...
            sin(alphaBS).*sin(phi)));
        bb=exp(sqrt(-1).*pi.*deltaMS(g).*cos(phi-alphaMS));
        CCF_teld(q,g)=sum(aa.^2.*bb.^2)/length(phi);
    end
end
deltaBS_max=deltaBS(length(deltaBS));
deltaMS_max=deltaMS(length(deltaMS));
if kappa==0
    E=(1/(deltaBS_max*deltaMS_max)*trapz(deltaBS,...
      trapz(deltaMS,abs(CCF-CCF_teld).^p)))^(1/p);
else
    E=norm(ACF-ACF_teld,p);
end

%-------------------------------------------------------------------------
% MIMO_one_ring_gains.m---------------------------------------------------
%
% Program for the determination of the MIMO one-ring gains
%
% Used file: MIMO_one_ring_parameters.m
%-------------------------------------------------------------------------
%[H]=MIMO_one_ring_gains(Mbs,Mms,N,alphaBS,alphaMS,phimax,kappa,f_max,phi_0
%,alphav)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% Mbs: number of antennas element in base station
% Mms: number of antennas element in mobile station
% N: number of scatterers
% alphaBS: BS antenna tilt angle in degre
% alphaMS: MS antenna tilt angle in degre
% phimax: one half of the maximum angle of departure in degre
% kappa: parameter used in Mises density
% f_max: maximum doppler frequency
% phi_0: the angle of rotation in degre
% alphav: the angle of motion in degree
```

```
function [H]=MIMO_one_ring_gains(Mbs,Mms,N,alphaBS,alphaMS,phimax,...
            kappa,f_max,phi_0,alphav)
nbr=10000;
p=2;
[phi]=MIMO_one_ring_parameters(N,alphaBS,alphaMS,phimax,...
      kappa,f_max,phi_0,alphav,p,'mmea');
lamda=0.15;
teta=2*pi*rand(nbr,N);
deltaBS=lamda/2;
deltaMS=lamda/2;
t=linspace(0,10,nbr);
H=zeros(Mbs,Mms,nbr);
h=zeros(1,nbr);
for q=1:Mbs
    for p=1:Mms
        for i=1:nbr
            sum1=0;
            for n=1:length(phi)
                a(q)=exp(sqrt(-1)*pi*(Mbs-2*q+1)*(deltaBS/lamda)*...
                    (cos(alphaBS)+phimax*sin(alphaBS)*sin(phi(n))));
                b(p)=exp(sqrt(-1)*pi*(Mms-2*p+1)*(deltaMS/lamda)*...
                        cos(phi(n)-alphaMS));
                f=f_max.*cos(phi(n)-alphav);
                sum1=sum1+a(q)*b(p)*exp(sqrt(-1)*(2*pi*f.*t(i)+teta(i,n)));
            end
            H(q,p,i)=sum1/sqrt(length(phi));
        end
    end
end

%-------------------------------------------------------------------------
% MIMO_one_ring_Im_response.m-------------------------------------------
%
% Program for the determination of the impulse response of MIMO one-ring
%
%-------------------------------------------------------------------------
%[H_teld]=MIMO_one_ring_Im_response(H,tau_teld_n,a_l_teld)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% H: the channels gains matrix
% tau_teld_n: channel delays
% a_l_teld: channel gains

function [H_teld]=MIMO_one_ring_Im_response(H,tau_teld_n,a_l_teld)
S=size(H);
H_teld=zeros(S(1),S(2),S(3));
for q=1:S(1)
    for p=1:S(2)
        for i=1:S(3)
            sum1=0;
            for n=1:length(a_l_teld)
                sum1=sum1+a_l_teld(n)/length(a_l_teld).*H(q,p,i);
            end
            H_teld(q,p,i)=sum1;
        end
    end
end
```

MATLAB-PROGRAMS of MIMO two-ring model simulator

```
%-------------------------------------------------------------------------
% gen_phiR_MIMO_tow_ring.m-----------------------------------------------
%
% Program for the computation of the angle of arrivals phi of
% MIMO two-ring model using LPNM method and Modified MEA
%
% Used files: data_two_ring.mat, errorfunc2ring.m
%-------------------------------------------------------------------------
% [phiR]=gen_phiR_MIMO_tow_ring(M,alphaR,fR_max,betaR,,kappa,p,method,PLOT)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: number of scatterers
% alphaR: MS antenna tilt angle in degre
% fR_max: Doppler frequency at the receiver side
% betaR: receive antenna tilt angle
% p: the power of LPNM method
% method: parameters computation method that can be:
%       1) Modified MEA:    'MMEA'
%       2) Lp-norm method:  'LPNM'
% PLOT: if PLOT==1, plot of the Time ACF and 2D-space CCF of the reference
% model as well as the simulation model

function
[phiR]=gen_phiR_MIMO_tow_ring(N,alphaR,fR_max,betaR,kappa,p,method,PLOT)
if nargin<8
    PLOT=0;
end
% converting parameters from degre to radium
alphaR=2*pi*alphaR/360;
betaR=2*pi*betaR/360;
save data_tow_ring.mat N alphaR fR_max kappa betaR p;
% computation of the parameters phi using Extended MEDS
if all(upper(method)=='MMEA')
    phiR=zeros(N,1);
    for n=1:N
        phiR(n)=fzero(@(x) fun_phi(n,N,kappa,x,betaR),betaR);
    end
elseif all(upper(method)=='LPNM')
    % computation of initial values for the parameter phi using EMEDS
    phiR=zeros(N,1);
    for n=1:N
        phiR(n)=fzero(@(x) fun_phi(n,N,kappa,x,betaR),betaR);
    end
    % determination of the phases phi through LPNM method
    options =
optimset('Display','iter','MaxIter',1000,'TypicalX',phiR,'TolFun',1e-12);
    x=fminsearch('errorfunc2ring',phiR,options);
    phiR=x;
else
    disp('enter a valid input method belonging to {MMEA,LPNM}');
end

tau=linspace(0,N/(2*fR_max),8*N);
phiMS=linspace(-pi,pi,4*N);
for k=1:length(tau)
```

```
    f=fR_max*cos(phiMS-alphaR);
    ACF(k)=trapz(phiMS,exp(-2*...
            sqrt(-1)*pi*f*tau(k)).*Mises(phiMS,kappa,betaR));
    ff=fR_max*cos(phiR-alphaR);
    ACF_teld(k)=sum(exp(-2*sqrt(-1)*pi*ff*tau(k)))/length(phiR);
end

% testing the correctness of the simulation model by determining the 2D-
space
% CCF for the simulation model as well as the reference model
deltaR=linspace(0,N/4,2*N);
tau1=linspace(0,N/(4*fR_max),2*N);
phiMS=linspace(-pi,pi,4*N);
for q=1:length(deltaR)
    for g=1:length(tau1)
        a=exp(sqrt(-1)*pi*deltaR(q).*cos(phiMS-betaR));
        f=fR_max*cos(phiMS-alphaR);
        ro12(q,g)=trapz(phiMS,a.^2.*...
            exp(-2*sqrt(-1)*pi*f*tau1(g)).*Mises(phiMS,kappa,betaR));
        aa=exp(sqrt(-1)*pi*deltaR(q).*cos(phiR-betaR));
        ff=fR_max*cos(phiR-alphaR);
        ro12_teld(q,g)=sum(aa.^2.*...
            exp(-2*sqrt(-1)*pi*ff*tau1(g)))/length(phiR);
    end
end
Err=ro12-ro12_teld;
if PLOT==1
    subplot(2,2,1);
    plot(tau*fR_max,real(ACF));
    hold on;
    plot(tau*fR_max,real(ACF_teld),'r*')
    xlim([0 N/2])
    ylim([-1 1])
    set(0,'DefaultAxesFontSize',16);
    legend({['Reference model'],['Simulation model with N =',...
      num2str(N)]},'FontName','Arial','Location','NorthEast','FontSize',...
      16,'Interpreter','latex');
    xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',...
      20,'Interpreter','latex');
    ylabel({'Time ACF'},'FontName','Arial','FontSize',...
      20,'Interpreter','latex');
    subplot(2,2,2);
    surf(tau1*fR_max,deltaR,real(ro12));
    zlim([-1 1])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
    title({'Reference model'},'FontName','Arial','FontSize',...
      20,'Interpreter','latex');
    zlabel({'CCF'},'FontName','Arial','FontSize',20,'Interpreter','latex');
    xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',20,...
    'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_R$ / $\lambda$'},'FontName','Arial','FontSize',20,...
        'Rotation',-20,'Interpreter','latex');
    subplot(2,2,3);
    surf(tau1*fR_max,deltaR,real(ro12_teld));
    zlim([-1 1])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
    title({'Simulation model'},'FontName','Arial','FontSize',...
      20,'Interpreter','latex');
    zlabel({'CCF'},'FontName','Arial','FontSize',20,'Interpreter','latex');
```

```
    xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',20,...
        'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_R$ / $\lambda$'},'FontName','Arial','FontSize',20,...
        'Rotation',-20,'Interpreter','latex');
    subplot(2,2,4);
    surf(tau1*fR_max,deltaR,real(Err));
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
    zlabel({'Error function'},'FontName','Arial','FontSize',...
      20,'Interpreter','latex');
    xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',20,...
        'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_R$ / $\lambda$'},'FontName','Arial','FontSize',20,...
        'Rotation',-20,'Interpreter','latex');
end

%-------------------------------------------------------------------------
% gen_phiT_MIMO_tow_ring.m-----------------------------------------------
%
% Program for the computation of the angle of departure phi of
% MIMO two-ring model using LPNM method and Modified MEA
%
% Used files: data_two_ring.mat, errorfunc2ring.m
%-------------------------------------------------------------------------
% [phiT]=gen_phiT_MIMO_tow_ring(M,alphaT,fT_max,betaT,,kappa,p,method,PLOT)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: number of scatterers
% alphaT: BS antenna tilt angle in degre
% fT_max: Doppler frequency at the tranmitter side
% betaT: transmit antenna tilt angle
% p: the power of LPNM method
% method: parameters computation method that can be:
%       1) Modified MEA:    'MMEA'
%       2) Lp-norm method:  'LPNM'
% PLOT: if PLOT==1, plot of the Time ACF and 2D-space CCF of the reference
% model as well as the simulation model

function
[phiT]=gen_phiT_MIMO_tow_ring(N,alphaT,fT_max,betaT,kappa,p,method,PLOT)
if nargin<8
    PLOT=0;
end
% converting parameters from degre to radium
alphaT=2*pi*alphaT/360;
betaT=2*pi*betaT/360;
save data_tow_ring.mat N alphaT fT_max kappa betaT p;
% computation of the parameters phi using Extended MEDS
if all(upper(method)=='MMEA')
    phiT=zeros(N,1);
    for n=1:N
        phiT(n)=fzero(@(x) fun_phi(n,N,kappa,x,betaT),betaT);
    end
elseif all(upper(method)=='LPNM')
    % computation of initial values for the parameter phi using EMEDS
    phiT=zeros(N,1);
    for n=1:N
        phiT(n)=fzero(@(x) fun_phi(n,N,kappa,x,betaT),betaT);
    end
    % determination of the phases phi through LPNM method
```

```
    options = optimset('Display','iter','MaxIter',1000,...
      'TypicalX',phiT,'TolFun',1e-12);
    x=fminsearch('errorfunc2ring',phiT,options);
    phiT=x;
else
    disp('enter a valid input method belonging to {MMEA,LPNM}');
end

tau=linspace(0,N/(2*fT_max),8*N);
phiBS=linspace(-pi,pi,4*N);
for k=1:length(tau)
    f=fT_max*cos(phiBS-alphaT);
    ACF(k)=trapz(phiBS,exp(-2*...
      sqrt(-1)*pi*f*tau(k)).*Mises(phiBS,kappa,betaT));
    ff=fT_max*cos(phiT-alphaT);
    ACF_teld(k)=sum(exp(-2*sqrt(-1)*pi*ff*tau(k)))/length(phiT);
end

% testing the correctness of the simulation model by determining the 2D-
space
% CCF for the simulation model as well as the reference model
deltaT=linspace(0,N/4,2*N);
tau1=linspace(0,N/(4*fT_max),2*N);
phiBS=linspace(-pi,pi,4*N);
for q=1:length(deltaT)
    for g=1:length(tau1)
        a=exp(sqrt(-1)*pi*deltaT(q).*cos(phiBS-betaT));
        f=fT_max*cos(phiBS-alphaT);
        ro12(q,g)=trapz(phiBS,a.^2.*...
            exp(-2*sqrt(-1)*pi*f*tau1(g)).*Mises(phiBS,kappa,betaT));
        aa=exp(sqrt(-1)*pi*deltaT(q).*cos(phiT-betaT));
        ff=fT_max*cos(phiT-alphaT);
        ro12_teld(q,g)=sum(aa.^2.*...
            exp(-2*sqrt(-1)*pi*ff*tau1(g)))/length(phiT);
    end
end
Err=ro12-ro12_teld;
if PLOT==1
    subplot(2,2,1);
    plot(tau*fT_max,real(ACF));
    hold on;
    plot(tau*fT_max,real(ACF_teld),'r*')
    xlim([0 N/2])
    ylim([-1 1])
    set(0,'DefaultAxesFontSize',16);
    legend({['Reference model'],...
      ['Simulation model with N = ',num2str(N)]},...
      'FontName','Arial','Location','NorthEast',...
            'FontSize',16,'Interpreter','latex');
    xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',...
      20,'Interpreter','latex');
    ylabel({'Time ACF'},'FontName','Arial','FontSize',...
      20,'Interpreter','latex');
    subplot(2,2,2);
    surf(tau1*fT_max,deltaT,real(ro12));
    zlim([-1 1])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
    title({'Reference model'},'FontName','Arial','FontSize',...
      20,'Interpreter','latex');
    zlabel({'CCF'},'FontName','Arial','FontSize',20,'Interpreter','latex');
```

```
    xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',20,...
            'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_T$ / $\lambda$'},'FontName','Arial','FontSize',20,...
        'Rotation',-20,'Interpreter','latex');
    subplot(2,2,3);
    surf(tau1*fT_max,deltaT,real(ro12_teld));
    zlim([-1 1])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
    title({'Simulation model'},'FontName','Arial','FontSize',...
      20,'Interpreter','latex');
    zlabel({'CCF'},'FontName','Arial','FontSize',20,'Interpreter','latex');
    xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',20,...
        'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_T$ / $\lambda$'},'FontName','Arial','FontSize',20,...
        'Rotation',-20,'Interpreter','latex');
    subplot(2,2,4);
    surf(tau1*fT_max,deltaT,real(Err));
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
    zlabel({'Error function'},'FontName','Arial','FontSize',...
      20,'Interpreter','latex');
    xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',20,...
        'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_T$ / $\lambda$'},'FontName','Arial','FontSize',20,...
        'Rotation',-20,'Interpreter','latex');
end

%-------------------------------------------------------------------------
% errorfunc2ring.m--------------------------------------------------------
%
% Program for the computation of the joint error function of the reference
% model and the simulation model that will be used in LPNM method
%
% used files: data_two_ring.mat
%-------------------------------------------------------------------------
% [E]=errorfunc2ring(phi)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% phi: angles of arrival/departure

function [E]=errorfunc2ringR(phi)
load data_tow_ring.mat;

% The error function between time ACF reference and time ACF simulation
tau=linspace(0,N/(2*f_max),8*N);
phiRT=linspace(-pi,pi,4*N);
for k=1:length(tau)
    f=f_max*cos(phiRT-alphaRT);
    ACF(k)=trapz(phiRT,exp(-2*...
      sqrt(-1)*pi*f*tau(k)).*Mises(phiRT,kappa,betaRT));
    ff=f_max*cos(phi-alphaRT);
    ACF_teld(k)=sum(exp(-2*sqrt(-1)*pi*ff*tau(k)))/length(phi);
end

% The error function between space CCF reference and space CCF simulation
deltaRT=linspace(0,N/4,N);
tau=linspace(0,N/(4*f_max),N);
phiRT=linspace(-pi,pi,6*N);
```

```
for q=1:length(deltaRT)
    for g=1:length(tau)
        a=exp(sqrt(-1)*pi*deltaRT(q).*cos(phiRT-betaRT));
        f=f_max*cos(phiRT-alphaRT);
        ro12(q,g)=trapz(phiRT,a.^2.*...
            exp(-2*sqrt(-1)*pi*f*tau(g)).*Mises(phiRT,kappa,pi));
        aa=exp(sqrt(-1)*pi*deltaRT(q).*cos(phi-betaRT));
        ff=f_max*cos(phi-alphaRT);
        ro12_teld(q,g)=sum(aa.^2.*...
            exp(-2*sqrt(-1)*pi*ff*tau(g)))/length(phi);
    end
end
deltaRT_max=deltaRT(length(deltaRT));
tau_max=tau(length(tau));
if kappa==0
    E=(1/(deltaRT_max*tau_max)*...
      trapz(deltaRT,trapz(tau,abs(ro12-ro12_teld).^p)))^(1/p);
else
    E=norm(ACF-ACF_teld,p);
end

%--------------------------------------------------------------------------
% MIMO_tow_ring_gains.m-----------------------------------------------------
%
% Program for the determination of the MIMO one-ring gains
%
% Used file: gen_phiR_MIMO_tow_ring.m, gen_phiR_MIMO_tow_ring.m
%--------------------------------------------------------------------------
%[H]=MIMO_tow_ring_gains(N,M,Rt,Rr,Mt,Mr,kappa,alphaT,fT_max,betaT,alphaR,
% fR_max,betaR)
%--------------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: number of scatterers arround the BS
% M: number of scatterers arround the MS
% Rt: radii of transmitter ring
% Rr: radii of receiver ring
% Mt: number of antennas element in base station
% Mr: number of antennas element in mobile station
% kappa
% alphaT: BS antenna tilt angle in degre
% fT_max: Doppler frequency at the tranmitter side
% betaT: transmit antenna tilt angle
% alphaR: MS antenna tilt angle in degre
% fR_max: Doppler frequency at the receiver side
% betaR: receive antenna tilt angle

function [H]=MIMO_tow_ring_gains(N,M,Rt,Rr,Mt,Mr,kappa,alphaT,fT_max,...
      betaT,alphaR,fR_max,betaR);

%MIMO one-ring parameters
alphaT=2*pi*alphaT/360;
betaT=2*pi*betaT/360;
alphaR=2*pi*alphaR/360;
betaR=2*pi*betaR/360;
nbr=10000;
p=2;
[phiR]=gen_phiR_MIMO_tow_ring(M,alphaR,fR_max,betaR,kappa,p,'mmea')
[phiT]=gen_phiT_MIMO_tow_ring(N,alphaT,fT_max,betaT,kappa,p,'mmea')
lamda=0.15;
deltaT=lamda/2;
```

```
deltaR=lamda/2;
tetan=2*pi*rand(1,length(phiR));
tetam=2*pi*rand(1,length(phiT));
tetamn=[];
for a=1:length(phiR)
    tetamn=[tetamn;mod(tetam+tetan(a),2*pi)];
end
fT=fT_max*cos(phiT-alphaT);
fR=fR_max*cos(phiR-alphaR);
t=linspace(0,10,nbr);
H=zeros(Mt,Mr,nbr);
for q=1:Mt
    for p=1:Mr
        for i=1:nbr
        sum2=0;
        for m=1:length(phiT)
            sum1=0;
            for n=1:length(phiR)
                an=exp((Mt-2*q+1).*...
                  sqrt(-1)*pi*deltaT*(cos(phiT(m)-betaT)));
                bn=exp((Mr-2*p+1).*sqrt(-1)*pi*deltaR*cos(phiR(n)-betaR));
                cmn=exp(sqrt(-1)*2*pi*...
                  (Rt*cos(phiT(m))-Rr*cos(phiR(n)))/lamda);
                sum1=sum1+an*bn*cmn*...
                  exp(sqrt(-1)*2*pi*(fT(m)+fR(n)).*t(i)+tetamn(n,m));
            end
            sum2=sum2+sum1;
        end
        H(q,p,i)=sum2/(sqrt(length(phiR))*sqrt(length(phiT)));
        end
    end
end
```

MATLAB-PROGRAMS of MIMO elliptical model simulator

```
%-------------------------------------------------------------------------
% MIMO_elliptical_parameters.m-------------------------------------------
%
% Program for the computation of the angle of arrivals phi of MIMO
% elliptical model using LPNM method and
%
% Used files: data_elliptical.mat, errorfunc_elliptical.m
%-------------------------------------------------------------------------
% [phi]=MIMO_elliptical_parameters(N,alphaBS,alphaMS,phimax,kappa,f_max,
%       phi_0,p,PLOT)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: number of scatterers
% alphaT: BS antenna tilt angle in degre
% alphaR: MS antenna tilt angle in degre
% kappa: parameter used in Mises density
% f_max: maximum doppler frequency
% phi_0: the angle of rotation in degre
% alphav: the angle of motion in degre
% p: the power of LPNM method
% method: parameters computation method that can be:
```

```
%       1) Modified MEA:    'MMEA'
%       2) Lp-norm method:  'LPNM'
% PLOT: if PLOT==1, plot of the Time ACF and 2D-space CCF of the reference
% model as well as the simulation model

function [phi]=MIMO_elliptical_parameters(N,alphaT,alphaR,kappa,...
           f_max,phi_0,alphav,p,method,PLOT)
if nargin<10
    PLOT=0;
end
% converting parameters from degre to radium
alphaT=2*pi*alphaT/360;
alphaR=2*pi*alphaR/360;
alphav=2*pi*alphav/360;
phi_0=2*pi*phi_0/360;
save data_elliptical.mat N alphaT alphaR kappa f_max p phi_0 alphav;

% computation of an initial values for the parameter phi using MMEA
if all(upper(method)=='MMEA')
    phi=zeros(N,1);
    for n=1:N
        phi(n)=fzero(@(x) fun_phi(n,N,kappa,x,phi_0),phi_0);
    end
elseif all(upper(method)=='LPNM')
    phi=zeros(N,1);
    for n=1:N
        phi(n)=fzero(@(x) fun_phi(n,N,kappa,x,phi_0),phi_0);
    end
    % determination of the phases phi through LPNM method
    options = optimset('Display','iter','TolFun',...
      1*10^(-12),'MaxIter',1000,'MaxFunEvals',100000000,'TypicalX',...
      phi,'TolX',1*10^(-12));
    x=fminsearch('errorfunc_elliptical',phi,options);
    phi=x;
else
    disp('enter a valid input method belonging to{MMEA,LPNM}');
end
% testing the correctness of the simulation model by determining the time
% ACF for the simulation model as well as the reference model
tau=linspace(0,N/(2*f_max),12*N);
phiMS=linspace(-pi,pi,4*N);
for k=1:length(tau)
    ACF(k)=trapz(phiMS,exp(-sqrt(-1)*2*pi*...
      f_max*cos(phiMS-alphav)*tau(k)).*Mises(phiMS,kappa,phi_0));
    ACF_teld(k)=sum(exp(-sqrt(-1).*2.*pi.*...
      f_max.*cos(phi-alphav).*tau(k))./length(phi);
end

% testing the correctness of the simulation model by determining the 2D-
% space CCF for the simulation model as well as the reference model
deltaT=linspace(0,N/4,2*N);
deltaR=linspace(0,N/4,2*N);
phiR=linspace(-pi,pi,4*N);
CCF_teld=zeros(length(deltaT),length(deltaR));
CCF=zeros(length(deltaT),length(deltaR));
for q=1:length(deltaT)
    for g=1:length(deltaR)
        cn=exp(-sqrt(-1).*2.*pi.*deltaT(q).*cos(gen_phiT(phiR)-alphaT));
        dn=exp(-sqrt(-1).*2*pi.*deltaR(g).*cos(phiR-alphaR));
        CCF(q,g)=trapz(phiR,cn.*dn.*Mises(phiR,kappa,phi_0));
        cn_sim=exp(-sqrt(-1).*2.*pi.*deltaT(q).*...
```

```
            cos(gen_phiT(phi)'-alphaT));
        dn_sim=exp(-sqrt(-1).*2.*pi.*deltaR(g).*cos(phi-alphaR));
        CCF_teld(q,g)=sum(cn_sim.*dn_sim)/length(phi);
    end
end
Err=CCF-CCF_teld;
if PLOT==1
    subplot(2,2,1);
    grid on;
    plot(tau*f_max,real(ACF));
    hold on;
    plot(tau*f_max,real(ACF_teld),'r*')
    xlim([0 N/2])
    ylim([-1 1])
    set(0,'DefaultAxesFontSize',16);
    legend({['Reference model'],['Simulation model with N = ',...
      num2str(N)]},'FontName','Arial','Location','NorthEast','FontSize',...
            16,'Interpreter','latex');
    xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    ylabel({'Time ACF'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    subplot(2,2,2);
    surf(deltaR,deltaT,real(CCF));
    zlim([-1 1])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
    title({'Reference model'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    zlabel({'2D space CCF'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    xlabel({'$\delta_{R}$ / \lambda$'},'FontName','Arial','FontSize',20,...
        'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{T}$ /$\lambda$'},'FontName','Arial','FontSize',20,...
        'Rotation',-20,'Interpreter','latex');
    subplot(2,2,3);
    surf(deltaR,deltaT,real(CCF_teld));
    zlim([-1 1])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
    title({'Simulation model'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    zlabel({'2D space CCF'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    xlabel({'$\delta_{R}$ /$\lambda$'},'FontName','Arial','FontSize',20,...
        'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{T}$ /$\lambda$'},'FontName','Arial','FontSize',20,...
        'Rotation',-20,'Interpreter','latex');
    subplot(2,2,4);
    surf(deltaR,deltaT,real(Err));
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
    zlabel({'Error function'},'FontName','Arial','FontSize',...
            20,'Interpreter','latex');
    xlabel({'$\delta_{R}$ /$\lambda$'},'FontName','Arial','FontSize',20,...
        'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{T}$ /$\lambda$'},'FontName','Arial','FontSize',20,...
        'Rotation',-20,'Interpreter','latex');
end
```

```
%------------------------------------------------------------------------
% gen_phiT.m-------------------------------------------------------------
%
% Program for the computation of angles of departure according to given
% angles of arrivals
%
%------------------------------------------------------------------------
% [phiT]=gen_phiT(phiR)
%------------------------------------------------------------------------
% Explanation of the input parameters:
%
% phiR: angles of arrivals

function [phiT]=gen_phiT(phiR)
v=1.5;
phiT=zeros(1,length(phiR));
phiv=atan((v^2-1)/(2*v));
for i=1:length(phiR)
  if (phiR(i)>-pi)&(phiR(i)<=-phiv)
             phiT(i)=atan(((v^2-1)*...
                  sin(phiR(i)))/((2*v)-(v^2+1)*cos(phiR(i))));
  elseif (phiR(i)>-phiv)&(phiR(i)<=0)
             phiT(i)=atan(((v^2-1)*...
                  sin(phiR(i)))/((2*v)-(v^2+1)*cos(phiR(i))))-pi;
  elseif (phiR(i)>0)&(phiR<=phiv)
             phiT(i)=atan(((v^2-1)*...
                  sin(phiR(i)))/((2*v)-(v^2+1)*cos(phiR(i))))+pi;
  elseif (phiR(i)>phiv)&(phiR<=pi)
             phiT(i)=atan(((v^2-1)*...
                  sin(phiR(i)))/((2*v)-(v^2+1)*cos(phiR(i))));
  end
end

%------------------------------------------------------------------------
% errorfunc_elliptical.m-------------------------------------------------
----------
%
% Program for the computation of the joint error function of the reference
% model and the simulation model that will be used in LPNM method
%
% used files: data_elliptical.mat
%------------------------------------------------------------------------
% [E]=errorfunc(phi)
%------------------------------------------------------------------------
% Explanation of the input parameters:
%
% phiR: angle of arrival phi

function [E]=errorfunc_elliptical(phi)
load data_elliptical.mat;

% The error function between time ACF reference and time ACF simulation
tau=linspace(0,N/(4*f_max),6*N);
phiMS=linspace(-pi,pi,6*N);
for k=1:length(tau)
    ACF(k)=trapz(phiMS,exp(-sqrt(-1)*2*pi*f_max*cos(phiMS-
alphav)*tau(k)).*Mises(phiMS,kappa,phi_0));
    ACF_teld(k)=sum(exp(-sqrt(-1)*2*pi*f_max*cos(phi-
alphav)*tau(k)))/length(phi);
end
```

```
% The error function between space CCF reference and space CCF simulation
deltaT=linspace(0,N/4,N);
deltaR=linspace(0,N/4,N);
phiR=linspace(-pi,pi,4*N);
for q=1:length(deltaT)
    for g=1:length(deltaR)
        cn=exp(-sqrt(-1).*2.*pi.*deltaT(q).*cos(gen_phiT(phiR)-alphaT));
        dn=exp(-sqrt(-1).*2*pi.*deltaR(g).*cos(phiR-alphaR));
        CCF(q,g)=trapz(phiR,cn.*dn.*Mises(phiR,kappa,phi_0));
        cn_sim=exp(-sqrt(-1).*2.*pi.*deltaT(q).*cos(gen_phiT(phi)-alphaT));
        dn_sim=exp(-sqrt(-1).*2.*pi.*deltaR(g).*cos(phi'-alphaR));
        CCF_teld(q,g)=sum(cn_sim.*dn_sim)/length(phi);
    end
end
deltaT_max=deltaT(length(deltaT));
deltaR_max=deltaR(length(deltaR));

if kappa==0
    E=norm(ACF-ACF_teld,p);
else
    E=(1/(deltaT_max*deltaR_max)*...
      trapz(deltaT,trapz(deltaR,(abs(CCF-CCF_teld)).^p)))^(1/p);
end

%-------------------------------------------------------------------------
% MIMO_elliptical_gains.m-----------------------------------------------
%
% Program for the determination of the MIMO one-ring gains
%
% Used file: data_one_ring.mat
%-------------------------------------------------------------------------
%[H]=MIMO_elliptical_gains(Mt,Mr,N,alphaT,alphaR,kappa,f_max,phi_0,alphav)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% Mt: number of antennas element in base station
% Mr: number of antennas element in mobile station
% alphaT: BS antenna tilt angle in degre
% alphaR: MS antenna tilt angle in degre
% kappa: parameter used in Mises density
% f_max: maximum doppler frequency
% phi_0: the angle of rotation in degre
% alphav: the angle of motion in degre

function
[H]=MIMO_elliptical_gains(Mt,Mr,N,alphaT,alphaR,kappa,f_max,phi_0,alphav)

%MIMO elliptical parameters
nbr=10000;
teta=2*pi*rand(nbr,N);
lamda=0.15;
deltaT=lamda/2;
deltaR=lamda/2;
t=linspace(0,10,nbr);
H=zeros(Mt,Mr,nbr);
p=2;
for l=1:Mt
    for k=1:Mr
```

```
[phi]=MIMO_elliptical_parameters(N,alphaT,alphaR,kappa,f_max,phi_0,alphav,p
,'lpnm');
        f=f_max.*cos(phi-alphav);
        for i=1:nbr
            sum1=0;
            for n=1:length(phi)
                a=exp(sqrt(-1)*pi*...
                  (Mt-2*l+1)*(deltaT/lamda)*cos(gen_phiT(phi(n)-alphaT)));
                b=exp(sqrt(-1)*pi*(Mr-2*k+1)*...
                  (deltaR/lamda)*cos(phi(n)-alphaR));
                sum1=sum1+a*b*exp(sqrt(-1)*(2*pi*f(n).*t(i)+teta(i,n)));
            end
            H(l,k,i)=sum1/sqrt(length(phi));
        end
        p=p+1;
    end
end

%-------------------------------------------------------------------------
% MIMO_elliptical_Im_response.m-------------------------------------------
%
% Program for the determination of the impulse response of MIMO elliptical
%
%-------------------------------------------------------------------------
%[H_teld]=MIMO_elliptical_Im_response(H,tau_teld_n,a_l_teld)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% Mt: number of antennas element in base station
% Mr: number of antennas element in mobile station
% N: number of scatterers
% alphaT: BS antenna tilt angle in degre
% alphaR: MS antenna tilt angle in degre
% f_max: maximum doppler frequency
% phi_0: the angle of rotation in degre
% alphav: the angle of motion in degre
% tau_teld_n: channel delays
% a_l_teld: channel gains

function [H_teld]=MIMO_elliptical_Im_response(Mt,Mr,N,alphaT,alphaR,...
            f_max,phi_0,alphav,tau_teld_n,a_l_teld)
kappa=0;
nbr=10000;
Z=zeros(Mt,Mr,nbr);
for c=1:4
    [H]=MIMO_elliptical_gains(Mt,Mr,N,alphaT,alphaR,kappa,...
            f_max,phi_0,alphav);
    Z=Z+1/2*H;
    kappa=kappa+2;
end
S=size(Z);
H_teld=zeros(S(1),S(2),S(3));
for q=1:S(1)
    for p=1:S(2)
        for i=1:S(3)
            sum1=0;
            for n=1:length(a_l_teld)
                sum1=sum1+a_l_teld(n).*Z(q,p,i);
            end
            H_teld(q,p,i)=sum1;
```

```
        end
    end
end
```

Common functions for MIMO channels simulators

```
% ------------------------------------------------------------------------
% Mises.m-----------------------------------------------------------------
%
% Program for the computation of von Mises PDF
%
%-------------------------------------------------------------------------
% [v]=Mises(phi,kappa,phi_0)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% phi: angle of arrival phi
% kappa: parameter used in Mises density
% phi_0: the angle of rotation

function [v]=Mises(phi,kappa,phi_0)

v=exp(kappa*cos(phi-phi_0))/(2*pi*besseli(0,kappa));

%-------------------------------------------------------------------------
% fun_phi.m ---------------------------------------------------------------
%
% Computation of the equation relatively to MMEA method that should be
% resolved in order to find the AoA phi_n
%
%-------------------------------------------------------------------------
% [y]=fun_phi(n,N,kappa,x,phi_0)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% n: iteration index
% N: number of the sinusoids
% x: arbitrary vector which should be as longer as N
% D: decorrelation distance

function [y]=fun_phi(n,N,kappa,x,phi_0)

phi_n=linspace(-pi,x,10000);
y=trapz(phi_n,Mises(phi_n,kappa,phi_0))-((1/N)*(n-1/4));

%-------------------------------------------------------------------------
% MIMO_freq_selec_parameters.m--------------------------------------------
%
% Program for the determination of the delays and gains for MIMO frequency
% selective channel
%
% Used file: hiperLAN_Parameters.mat
%-------------------------------------------------------------------------
%[tau_teld_n,a_l_teld]=MIMO_freq_selec_parameters(model)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
```

```
% model: the model defined by HIPERLAN standard which belong to:
%       1)  HIPERLAN model A: 'HIPLA'
%       2)  HIPERLAN model B: 'HIPLB'
%       3)  HIPERLAN model C: 'HIPLC'
%       4)  HIPERLAN model D: 'HIPLD'
%       5)  HIPERLAN model E: 'HIPLE'

function [tau_teld_n,a_l_teld]=MIMO_freq_selec_parameters(model)
load hiperLAN_Parameters.mat;
if all(upper(model)=='HIPLA')
    tau_teld_n=delaysA;
    a_l_teld=gainsA;
elseif all(upper(model)=='HIPLB')
    tau_teld_n=delaysB;
    a_l_teld=gainsB;
elseif all(upper(model)=='HIPLC')
    tau_teld_n=delaysC;
    a_l_teld=gainsC;
elseif all(upper(model)=='HIPLD')
    tau_teld_n=delaysD;
    a_l_teld=gainsD;
elseif all(upper(model)=='HIPLE')
    tau_teld_n=delaysE;
    a_l_teld=gainsE;
else
    disp('Please enter a right model containing in
{HipLA,HipLB,HipLC,HipLD,HipLE}');
end

%-------------------------------------------------------------------------
% MIMO_capacity.m----------------------------------------------------------
%
% Program for the determination and the plotting of the MIMO capacity
%
%-------------------------------------------------------------------------
%[C]=MIMO_elliptical_capacity(H,SNR,Ptotal,PLOT)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
% H: channel gains Hij
% SNR: signal to noise ration in dB
% Ptotal: total transmitted power in dBw
% PLOT: if PLOT==1, plot of the capacity of the channel bits/sec/Hz over
the variable time
function [C]=MIMO_capacity(H,SNR,Ptotal,PLOT)
if nargin<4
    PLOT=0;
end
nbr=10000;
SNR=10^(SNR/10);
Ptotal=10^(Ptotal/10);
S=size(H);
I=eye(S(1));
% computaion of the capacity C of the model
for i=1:nbr
    C(i)=real(log2(det(I+Ptotal/(S(1)*SNR).*H(:,:,i)*H(:,:,i)')));
end
Cmean=mean(C)*ones(1,length(C));
if PLOT==1
    plot(C);
    hold on
```

```
    plot(Cmean,'r--');
    set(0,'DefaultAxesFontSize',16);
    legend({'Capacity C','mean of C'},...
        'FontName','Arial','Location','NorthEast','FontSize',...
            24,'Interpreter','latex');
    ylabel({'Simulated channel capacity'},'FontName','Arial','FontSize',...
            24,'Interpreter','latex');
    xlabel({'Time, t (s)'},'FontName','Arial','FontSize',...
            24,'Interpreter','latex');
end
```

APPENDIX 3 - MATLAB PROGRAMS FOR PERFORMANCE TESTS

```
%-------------------------------------------------------------------
% acf_mue.m ---------------------------------------------------------
%
% Computation of the ACF of deterministic Gaussian processes mu_i(t)
%
%-------------------------------------------------------------------
% r_mm=acf_mue(f,c,tau)
%-------------------------------------------------------------------
% Explanation of the input parameters:
%
% f: discrete Doppler frequencies
% c: Doppler coefficients
% tau: time separation variable

function r_mm=acf_mue(f,c,tau)

r_mm=0;
for n=1:length(c),
    r_mm=r_mm+0.5*c(n)^2*cos(2*pi*f(n)*tau);
end

%-------------------------------------------------------------------
% acf_mue_avg.m -----------------------------------------------------
%
% Computation of the sample ACF of set of deterministic
% Gaussian processes mu_i(t)
%
%-------------------------------------------------------------------
% r_mm=acf_mue(f,c,tau)
%-------------------------------------------------------------------
% Explanation of the input parameters:
%
% f: discrete Doppler frequencies
% c: Doppler coefficients
% tau: time separation variable
% N_i: number of harmonic functions
% K: number of sets (partitions)
function r_mm_avg=acf_mue_avg(f,c,N_i,K,tau)
r_mm_avg=0;
for i=1:K,
    r_mm=0;
    for n=1:N_i,
        r_mm=r_mm+0.5*c(n)^2*cos(2*pi*f(i,n)*tau);
    end
    r_mm_avg=r_mm_avg + r_mm;
end
r_mm_avg=r_mm_avg/K;

%-------------------------------------------------------------------
% adf_sim.m ---------------------------------------------------------
%
% Program for the computation of the average duration of fades T_(r).
%
% Used m-files: cdf_sim.m, lcr_sim.m
%-------------------------------------------------------------------
% adf=adf_sim(xi_t,r,T_sim,PLOT)
%-------------------------------------------------------------------
% Explanation of the input parameters:
%
% xi_t: deterministic process or time-domain signal to be analysed
%       with respect to the average duration of fades T_(r)
```

```
% r: equidistant level vector
% T_sim: duration of the simulation
% PLOT: plot of the resulting average duration of fades T_(r),
%       if PLOT==1

function adf=adf_sim(xi_t,r,T_sim,PLOT)

cdf=cdf_sim(xi_t,r);
lcr=lcr_sim(xi_t,r,T_sim);

adf=cdf./lcr;

if PLOT==1,
   plot(r,adf,'rx')
   set(0,'DefaultAxesFontSize',16);
   xlabel({'$r$'},'FontName','Arial','FontSize',24,'Interpreter','latex')
   ylabel({'$T_-(r)$'},'FontName','Arial','FontSize',...
      24,'Interpreter','latex')
end

%-----------------------------------------------------------------------
% cdf_sim.m ------------------------------------------------------------
%
% Program for the computation of cumulative distribution
% functions F(r).
%
%-----------------------------------------------------------------------
% F_r=cdf_sim(xi_t,r,PLOT)
%-----------------------------------------------------------------------
% Explanation of the input parameters:
%
% xi_t: deterministic process or time-domain signal to be analysed
%       with respect to the cumulative distribution function F(r).
% r: level vector
% PLOT: plot of the resulting cumulative distribution function F(r),
%       if PLOT==1

function F_r=cdf_sim(xi_t,r,PLOT)

if nargin==2,
   PLOT=0;
end

F_r=zeros(size(r));

for l=1:length(r),
    F_r(l)=length(find(xi_t<=r(l)));
end

F_r=F_r/length(xi_t);

if PLOT==1,
   plot(r,F_r,'rx')
   xlabel({'$r$'},'FontName','Arial','FontSize',24,'Interpreter','latex')
   ylabel({'$F(r)$'},'FontName','Arial','FontSize',...
      24,'Interpreter','latex')
end

%-----------------------------------------------------------------------
```

```
% pdf_sim.m ------------------------------------------------------------
%
% Program for the computation of probability density functions p(z).
%
%-----------------------------------------------------------------------
% p_z=pdf_sim(xi_t,z,PLOT)
%-----------------------------------------------------------------------
% Explanation of the input parameters:
%
% xi_t: deterministic process or time-domain signal to be analysed
%       with respect to the probability density function p(z).
% z: equidistant level vector
% PLOT: plot of the resulting probability density function p(z),
%       if PLOT==1

function p_z=pdf_sim(xi_t,z,PLOT)

if nargin==2,
   PLOT=0;
end

p_z=hist(xi_t,z)/length(xi_t)/abs(z(2)-z(1));

if PLOT==1,
   plot(z,p_z,'mx')
   set(0,'DefaultAxesFontSize',16)
   xlabel('$z$','Interpreter','latex','FontSize',24)
   ylabel('$p(z)$','Interpreter','latex','FontSize',24)
end

%-----------------------------------------------------------------------
% lcr_sim.m ------------------------------------------------------------
%
% Program for the computation of the level-crossing rate N(r).
%
%-----------------------------------------------------------------------
% N_r=lcr_sim(xi_t,r,T_sim,PLOT)
%-----------------------------------------------------------------------
% Explanation of the input parameters:
%
% xi_t: deterministic process or time-domain signal to be analysed
%       with respect to the level-crossing rate N(r)
% r: level vector
% T_sim: duration of the simulation
% PLOT: plot of the resulting level-crossing rate N(r), if PLOT==1

function N_r=lcr_sim(xi_t,r,T_sim,PLOT)

if nargin==3,
   PLOT=0;
end

N_r=zeros(size(r));

for k=1:length(r),
    N_r(k)=sum(xi_t(2:length(xi_t)) < r(k) & ...
               xi_t(1:length(xi_t)-1) >= r(k) );
end
```

```
N_r=N_r/T_sim;

if PLOT==1,
   plot(r,N_r,'rx')
   set(0,'DefaultAxesFontSize',16)
   xlabel({'$r$'},'FontName','Arial','FontSize',24,'Interpreter','latex')
   ylabel({'$N(r)$'},'FontName','Arial','FontSize',...
      24,'Interpreter','latex')
end
```