UNIVERSITY OF AGDER

Development of a MATLAB Toolbox for Mobile Radio Channel Simulators

Dadallage Samansiri De Silva
Hassan Uthman Aboklosa

Supervisor
Prof. Dr.-Ing. habil. Matthias Uwe Pätzold

University of Agder, 2011
Faculty of Engineering and Science
Department of ICT

Abstract

A profound knowledge of mobile radio channels is required for the development, evaluation, and also assessment at practical conditions of present and future mobile radio communication systems. The modelling, analysis, and simulation of mobile radio channels are important sub area since the initiation of mobile communications. In addition to that knowledge of channel behaviour 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 and embed them into MATLB software as a toolbox.

Implemented channel simulators were combined into a user-friendly Matlab toolbox from which users can easily select well-known channel models to test and to study the performance of mobile communication systems. The help file was developed based on HTML. It gives better support for the new users to work on the developed channel simulators, run the test procedures as well as parameter computation. The help file consistent with other supplementary programs like computation of PDF and CDF for different distributions, Rice simulation model, extended Suzuki process type I and II simulator etc. In addition to that each program consists with guidelines embedded with the source code. The help file web interfaces are listed in Appendix-1.The toolbox can be integrated into the new release of Matlab software.

The toolbox contains channel simulators for simulating non-stationary land mobile satellite channel, spatial shadowing processes, MIMO channels, multiple uncorrelated Rayleigh fading channels, mobile to mobile channel, frequency hopping channels etc. We developed 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. Proposed new algorithms 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, “how can a channel simulator be tested?” have been address in the thesis as a research question. It was based on the comparison of simulation results with the measured model or the reference model under different scenarios. In addition to that selection of the simulation time duration, sampling rate and size of the samples were considered. Developed test procedures were helped to assess the implemented channel simulators.

Keywords: Matlab toolbox, parameter computation methods, Mobile fading channel simulators, Performance tests, Deterministic channel modelling.

Preface

This thesis was based on the Master thesis work carried out between January and May 2011 at Faculty of Science and Engineering, University of Agder, Grimstad, Norway. It was based on the implementation of the mobile radio channel simulators which were based on one of the literature M. Pätzold: Mobile Radio Channels, 2nd edition, Chichester: John Wiley & Sons, 2011. Similarly, the previous works serve as support for our thesis and stand as a starting point for the required tasks.

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

The thesis work was carried out as a group work and the content was divided as follows. Thesis writing, parameters computation, implementation of the non-stationary land mobile satellite channel, the spatial channel simulator for shadowing fading, implementation of the web based Matlab help file and development of the supplementary Matlab programs were done by Dadallage Samansiri De Silva. Implementation and thesis writing of the MIMO channel models, uncorrelated Rayleigh channels, and frequency hopping mobile radio channels were done by Hassan Utham Aboklosa.

We would like to send our special thanks to thesis supervisor Prof. Dr.-Ing. habil. Matthias Uwe Pätzold. His kind advices and guidance helped us to fulfil the master's thesis work. Without his help this work couldn't be achieved within the project timeframe. Finally, we are grateful to the Master's Studies Coordinator-Associate Prof. Ole Christopher, Head of Master's studies-Ms. Sissel Andreassen and International Coordinator-Mr. Tor Erik Kristiansen.

Dadallage Samansiri De Silva,
Hassan Utham Aboklosa,
Grimstad,
Norway.
25.05.2011.

Table of Contents

Page

Abstract.......... i
Preface.......... ii
Table of Content.......... iii
List of Figures.......... vi
List of Tables.......... x
CHAPTER 1 – INTRODUCTION.......... 1
1.1 BASIC CONCEPTS.......... 2
1.2 REQUIREMENTS.......... 6
1.2.1 Presentation of Matlab Environment.......... 6
1.3 THESIS OUTLINES.......... 7
CHAPTER 2 – PARAMETERS COMPUTATION.......... 9
2.1 FREQUENCY NON-SELECTIVE CHANNELS.......... 10
2.1.1 Parameterization of Sum-of-Sinusoids Channel Model.......... 10
2.1.1.1 Method of Equal Distances (MED).......... 10
2.1.1.2 Mean-Square-Error Method (MSEM).......... 11
2.1.1.3 Method of Equal Areas (MEA).......... 12
2.1.1.4 Monte Carlo Method (MCM).......... 13
2.1.1.5 L_p -norm Method (LPNM).......... 14
2.1.1.6 Method of Exact Doppler Spread (MEDS).......... 14
2.1.1.7 Jakes Method (JM).......... 15
2.1.1.8 Randomized MEDS (RMEDS).......... 15
2.1.1.9 MEDS with Set Partitioning (MEDS-SP).......... 16
2.1.1.10 Implementation.......... 17
2.1.1.11 Results.......... 18
2.1.1.12 Comparison of Parameter Computation Methods.......... 24
2.1.2 Parameterization of Sum-of-cisoids Channel Model.......... 26
2.1.2.1 Extended method of Exact Doppler Spread (EMEDS).......... 26
2.1.2.2 Lp-Norm Method (LPNM).......... 27
2.1.2.3 Generalized Method of Equal Area (GMEA).......... 27
2.1.2.4 Implementation.......... 28
2.1.2.5 Results.......... 29
2.1 FREQUENCY SELECTIVE CHANNELS.......... 31
2.2.1 Basic Concept.......... 31
2.2.1.1 Reference Model.......... 32
2.2.1.2 Simulation Model.......... 32
2.2.2 Parameter Computation Method.......... 32
2.2.2.1 Method of Equal Distance (MED).......... 33
2.2.2.2 Mean-Square-Error Method (MSEM).......... 34
2.2.2.3 Method of Equal Areas (MEA).......... 35

2.2.2.4 Monte-Carlo Method (MCM).. 36
2.2.2.5 Lp-norm Method (LPNM).. 37
2.2.3 Implimentation.. 38
2.2.4 Results.. 38
2.3 CONCLUSION OF THE PARAMETER COMPUTATION................................ 43
CHAPTER 3 – MOBILE FADING CHANNEL SIMULATORS.................................. 45
3.1 NON-STATIONARY LAND MOBILE SATELLITE CHANNEL..................... 45
3.1.1. Reference Model.. 46
3.1.2. Simulation Model... 47
3.1.3. Implementation. 48
3.1.4 Results and Interpretation.. 49
3.2 SPATIAL CHANNEL SIMULATOR FOR SHADOWING FADING................ 52
3.2.1 Reference Model.. 52
3.2.1.1 The PDF and CDF of Spatial Shadowing Process................................ 52
3.2.1.2 Spatial Correlation.. 53
3.2.1.3 Level-Crossing Rate and Average Duration of Fades........................... 53
3.2.3 Simulation Model... 54
3.2.4 Implementation... 55
3.2.5 Results and Interpretation.. 56
3.3 MIMO FADING CHANNEL SIMULATORS.. 60
3.3.1 MIMO Channel Simulator Based On One-Ring Model................................ 61
3.3.1.1 Geometrical One-Ring Scattering Model.. 61
3.3.1.2 Reference Model.. 62
3.3.1.3 Simulation Model... 64
3.3.1.4 An Extension to the Impulse Response of Frequency-Selective MIMO Channels.. 65
3.3.1.6 Results and interpretation.. 67
3.3.2 The Two-Ring MIMO Mobile-to-Mobile Channel Model............................ 69
3.3.2.1 Reference Model.. 69
3.3.2.2 Simulation Model... 71
3.3.2.3 Implementation... 74
3.3.2.4 Results and Interpretation.. 75
3.3.3 MIMO Channel Simulator Based on Elliptical Model……………………… 77
3.3.3.1 Reference Model.. 78
3.3.3.2 Simulation Model... 79
3.3.3.3 Implementation... 80
3.3.3.4 Results and Interpretation.. 81
3.4 UNCORRELATED RAYLEIGH CHANNELS.. 84
3.4.1 Problem Description.. 84
3.4.2 Generalized Method of Exact Doppler Spread (GMEDS$_q$)............................ 85
3.4.3 Implementation.. 87
3.4.4 Results and Interpretation.. 87
3.5 FREQUENCY HOPPING MOBILE RADIO CHANNELS................................. 90
3.5.1 Reference Model.. 90
3.5.2 Simulation Model... 90
3.5.3 Implementation.. 92

3.5.3 Results and Interpretation... 93
3.6 Conclusion of the Channel Models.. 95
CHAPTER 4 – CONCLUSIONS AND RECOMMENDATIONS................................ 96
REFERENCES……………………………………………………………………….. 99
ACRONYMS ………………………………………………………………………… 101
SYMBOLS …………………………………………………………………………… 103
APPENDIX 1- USER INTERFACES OF THE HELP FILE ………………………….. 107
Main Interface………………………………………………………………………… 108
Sub Interfaces………………………………………………………………………… 109
APPENDIX 2 - MATLAB PROGRAMS FOR PARAMETERS COMPUTATION METHODS…………………………………………………………………………… 126
MATLAB-PROGRAMS of Parameters Computation Methods for Frequency-Non-Selective Channels……………………………………………………………………. 126
Parameter Computation for SOS Channel Model……………………………………. 126
Parameter Computation for SOC Channel Model……………………………………. 138
MATLAB-PROGRAMS of Parameters Computation Methods for Frequency-Selective Channels……………………………………………………………………………. 145
APPENDIX 3 - MATLAB PROGRAMS FOR MOBILE FADING CHANNEL SIMULATORS ……………………………………………………………………. 157
MATLAB -PROGRAMS of the Fundamental Channel Simulators……………………. 157
MATLAB-PROGRAMS of Non-Stationary Land Mobile Satellite Channel Simulator... 166
MATLAB-PROGRAMS of the Spatial Shadowing Channel Simulator………………... 175
MATLAB-PROGRAMS of MIMO One-Ring Model Simulator……………………….. 179
MATLAB-PROGRAMS of MIMO Two-Ring Model Simulator………………………. 185
MATLAB-PROGRAMS of MIMO Elliptical Model Simulator………………………... 194
MATLAB-PROGRAMS of Uncorrelated Rayleigh Channel Simulator……………….. 200
MATLAB-PROGRAMS of Frequency Hopping Mobile Radio Channel Simulator…..... 205
APPENDIX 4 - MATLAB PROGRAMS FOR PERFORMANCE TESTS…………… 211
APPENDIX 5 - SUPPLEMENTARY MATLAB PROGRAMS OF THE TOOLBOX... 215
MATLAB-PROGRAMS for Computation of PDF and CDF for Given Distributions..... 215
MATLAB-PROGRAMS for Study of Elementary Properties of Rayleigh and Rice Channel (ACF and PSD)……………………………………………………………… 234

List of Figures

Page

Figure 1.1: Typical mobile radio propagation scenario…………………………………….. 2
Figure 1.2: Relationship between the physical channel, the stochastic reference model, the deterministic simulation model and measurement or specification …………… 4
Figure 1.3: An example of sum-of-sinusoids principle…………………………………….. 4
Figure 1.4: The steps of deterministic channel modelling.. 5
Figure 2.4: Set-partitioning principle…………………………………………………… 15
Figure 2.2: The implementation diagram of the parameters computation methods for the frequency-non-selective channel……………………………………………..... 16
Figure 2.5: (a) PSD and (b) ACF of Jakes model using MED method …………………… 17
Figure 2.6: (a) PSD and (b) ACF of Jakes model using MSEM method………………….. 18
Figure 2.7: (a) PSD and (b) ACF of Jakes model using MEA method…………………… 18
Figure 2.8: (a) Power density function and (b) Autocorrelation function using MCM method for Jakes model…………………………………………………….. 18
Figure 2.9: (a) PSD and (b) ACF of Jakes model using LPNM method………………….. 19
Figure 2.10: (a) PSD and (b) ACF of Jakes model using MEDS method………………….. 19
Figure 2.11: PSD and ACF of Jakes model using JM method…………………………... 20
Figure 2.12: (a) PSD and (b) ACF of Jakes model using RMEDS method………………... 20
Figure 2.13: (a) PSD and (b) ACF of Jakes model using MEDS-SP method……………… 21
Figure 2.14: PSD and ACF of Gauss model using MED method…………………………. 21
Figure 2.15: (a) PSD and (b) ACF of Gauss model using MSEM method……………....... 22
Figure 2.16: (a) PSD and (b) ACF of Gauss model using MEA method…………………... 22
Figure 2.17: a) PSD and (b) ACF of Gauss model using MCM method………………. … 22
Figure 2.18: (a) PSD and (b) ACF of Gauss model using MEDS method…………………. 23
Figure 2.19: (a) PSD and (b) ACF of Gauss model using LPNM method…………………. 23
Figure 2.20: Autocorrelation function for (a) N_i = 7 and (b) N_i = 21 (RMEDS, Jakes PSD). 24
Figure 2.19: (a) PSD and (b) ACF of Jakes model using MEDS-SP method $N=7$………. 24
Figure 2.20: Sample mean autocorrelation function by using (a) the MEDS-SP and (b) the RMEDS with N_i =10 and various values of K.. 25
Figure 2.21: The implementation diagram of the parameters computation methods for SOC channel…………….……………………………………………… 28
Figure 2.22: ACF of Jakes model using EMEDS method………………………………… 28
Figure 2.23: ACF of Jakes model using GMEA method…………………………………... 28
Figure 2.24: Comparison between the envelop distribution p_{ς}(z) of reference model and the envelop distribution $p_{\hat{\varsigma}}$ (z) of simulation model by applying the GMEA to the von Mises distribution of the angle of arrival α……………………… 29
Figure 2.25: Deterministic simulation model for frequency-selective mobile radio channel in the equivalent complex baseband………………………………………….. 30
Figure 2.26: The implementation diagram of the parameters computation methods for

the frequency-selective channel……………………………………………. 36
Figure 2.27: Absolute value of the FCF behaviour shown with referred to reference model for the 12 path, TU and under parameter computation methods of (a) MED, (b) MSEM, (c) MEA, (d) MCM and (f) LPNM……………………… 38
Figure 2.28: Absolute value of the frequency correlation function (FCF) behaviour shown (with referred to reference model for the 12 path, RA) under parameter computation methods of (a) MED, (b) MSEM, (c) MEA, (d) MCM and (f) LPNM……………………………………………………………………… 39
Figure 2.29: Absolute value of the frequency correlation function (FCF) behaviour shown (with referred to reference model for the 12 path, BU) under parameter computation methods of (a) MED, (b) MSEM,(c) MEA, (d) MCM and (f) LPNM……………………………………………………………………… 40
Figure 2.30: Absolute value of the frequency correlation function (FCF) behaviour shown (with referred to reference model for the 12 path, HT) under parameter computation methods of (a) MED, (b) MSEM, (c) MEA, (d) MCM and (f) LPNM……………………………………………………………………. 41
Figure 3.1: Dynamic model with M states for modelling of non-stationary LMS channel... 45
Figure 3.2: State transition of the two-state Markov model……………………………... 45
Figure 3.4: Block diagram for implementation of the 2 states non-stationary LMS simulation channel……………………………………………………….. 48
Figure 3.5: Simulated fading envelop under suburban area (15° elevation angle) obtained by using the dynamic (2+2)state model with embedded extended Suzuki process of Type I ………………………………………………………… 49
Figure 3.6: Complementary cumulative distribution function for (a) urban area (EA 25°) (b) suburban area (EA 15°) obtained by embedding the extended Suzuki process of Type I in dynamic (2+2)-state model……………………………. 49
Figure 3.7: Level-crossing rate for (a) urban area (EA 25°) (b) suburban area (EA 15°) obtained by embedding the extended Suzuki process of Type I in dynamic (2+2)-state model…………………………………………………….......... 49
Figure 3.8: Average duration of fades for (a) urban area (EA 25°) (b) suburban area (EA 15°) obtained by embedding the extended Suzuki process of Type I in dynamic (2+2)-state model……………………………………………………….. 50
Figure 3.9: Structure of the stochastic spatial shadowing simulator with constant gain c_n, constant spatial frequencies s_n, and random phases θ_n……………………… 54
Figure 3.10: The diagram for the implementation of the spatial shadowing process………. 54
Figure 3.11: The lognormal distribution of the reference model comparison with the simulation model for suburban and urban area ($c_n = \sqrt{2/N}$ and N = 25)...... 56
Figure 3.12: Spatial autocorrelation functions of reference model and simulation for (a) Suburban environment and (b) Urban environment under Gudmundson computation method, for MEA with N = 25………………………………….. 56
Figure 3.13: Spatial autocorrelation functions of reference model and simulation for (a) Suburban environment and (b) Urban environment under Gaussian computation method, for MEA with N = 25………………………………….. 57
Figure 3.14: Spatial autocorrelation functions of reference model and simulation for (a) Suburban environment and (b) Urban environment under Butterworth computation method, for MEA with N = 25………………………………….. 57

Figure 3.15: Spatial autocorrelation of the measured channel and the simulation model for (a) suburban areas (b) urban areas (Measurement-based correlation model, LPNM with $N = 25$)………………………………………………….. 57
Figure 3.16: Comparison of the exact and approximated solutions for level crossing rate of the simulation model for shadow fading in urban and suburban areas (Measurement-based correlation model, LPNM with $N = 25$)………………. 58
Figure 3.17example of MIMO system based on smart antennas………………………….. 60
Figure 3.18: Geometrical model (one-ring model)………………………………………. 63
Figure 3.19: Structure of the one-ring 2x2 MIMO channel simulator…………………….. 65
Figure 3.20: The diagram for the implementation of a MIMO channel simulator based on one-ring geometrical model………………………………………………….. 66
Figure 3.21: Time ACF reference model and simulation model for isotropic scattering ($\kappa = 0$)……………………………………………………………………… 67
Figure 3.22: The 2D space CCF $\rho(\delta_T, \delta_R)$ of the reference model for isotropic scattering. ($\kappa = 0, \beta_R = \beta_T = 90, \quad \alpha_{max}^T = 2^0$)…………………………………………. 68
Figure 3.23: The 2D space CCF $\tilde{\rho}(\delta_T, \delta_R)$ of the simulation model for isotropic scattering ($\kappa = 0, \beta_R = \beta_T = 90, \alpha_{max}^T = 2^0$, EMEDS, $N = 25$)…………………………… 68
Figure 3.24: The 2D space CCF $\rho(\delta_T, \delta_R)$ of the reference model for non-isotropic scattering ($\kappa = 0, \beta_R = \beta_T = 90, \alpha_{max}^T = 2^0$…………………………………… 68
Figure 3.25: The 2D space CCF $\tilde{\rho}(\delta_T, \delta_R)$ of the simulation model for non-isotropic scattering ($\kappa = 0, \beta_R = \beta_T = 90, \alpha_{max}^T = 2^0$, LPNM, $N = 25$)…………………. 68
Figure 3.26: the geometrical two-ring model for a 2x2 MIMO channel with local scatters around a mobile transmitter and mobile receiver…………………………….. 69
Figure3.27: Structure of the two-ring 2x2 MIMO channel simulator for mobile-to-mobile channels…………………………………………………………………….. 72
Figure 3.28: The diagram for the implementation of a MIMO channel simulator based on two-ring geometrical model…………………………………………………… 74
Figure 3.29: The transmitter CF $\rho_T\left(\delta_T, \tau\right)$of the 2x2 MIMO mobile-to-mobile reference model (isotropic scattering)... 75
Figure 3.30: The transmitter CF $\rho_T\left(\delta_T, \tau\right)$of the 2x2 MIMO mobile-to-mobile channel simulator based on EMEDS with $M = 40$ (isotropic scattering)...................... 75
Figure 3.31: Absolute error $| e_T(\delta_T,\tau) | = | \rho_T(\delta_T,\tau) - \tilde{\rho}_T(\delta_T,\tau) |$ By using the EMEDS with M=40 (isotropic scattering).. 75
Figure 3.32: Absolute value of the transmitter CF $| \rho_T(\delta_T,\tau) |$of the 2x2 MIMO mobile to mobile reference channel model under non-isotropic scattering condition (Von Mises density with m_α^T=60 and κ_c= 40)……………………………... 76
Figure 3.33: Absolute value of the transmitter CF $| \tilde{\rho}_T(\delta_T,\tau) |$of the 2x2 MIMO mobile to mobile channel simulator design with $m_\alpha^T = 60$ and $\kappa_c = 40$)……………… 76
Figure: 3.34: Geometrical elliptical scattering model for an M_T x M_R MIMO channel with local scatterers S_n placed on an ellipse…………………………………….. 77
Figure 3.35: Block diagram for the implementation of a MIMO channel simulator based on elliptical geometrical model……………………………………………… 80
Figure 3.36: The 2D space CCF (a) $\rho_{11,22}(\delta_T,\delta_R)$ of the reference model (isotropic scattering) (b) $\tilde{\rho}_{11,22}(\delta_T,\delta_R)$ of the simulation model(isotropic) for $\kappa = 0$, $N = 25$…………………………………………………………………….. 81

Figure 3.37: Absolute error of $|e_T(\delta_T,\tau)| = |\rho_{11,22}(\delta_T,\delta_R) - \tilde{\rho}_{11,22}(\delta_T,\delta_R)|$ By using the LPNM with (isotropic scattering)... 82
Figure 3.38: The time ACF $r_{h_{kl}}(\tau)$ of the reference model (isotropic scattering) for $\kappa = 0$, $N = 25$…………………………………………………………… 82
Figure 3.39: Absolute value of the 2D space CCF (non-isotropic scattering, von Mises density with parameter $\kappa = 10$ and $m_\alpha = 0$) (a) $|\rho_{11,22}(\delta_T,\delta_R)|$ of the reference model (b) $|\tilde{\rho}_{11,22}(\delta_T,\delta_R)|$ of the simulation model for N=25…….. 82
Figure 3.40: Absolute error of $|e_T(\delta_T,\tau)| = |\rho_{11,22}(\delta_T,\delta_R) - \tilde{\rho}_{11,22}(\delta_T,\delta_R)|$ By using the LPNM (non- isotropic scattering).. 83
Figure 3.41: Absolute value of the ACF $|r_{h_{kl}}(\tau)|$ of the reference model and $|\tilde{r}_{h_{kl}}(\tau)|$ simulation model (non-isotropic scattering, Von Mises density with parameter $\kappa = 10$ and $m_\alpha = 0$)……………………………………….. 83
Figure: 3.42: Structure of the SOS channel simulator generating Rayleigh fading waveforms (k = 1, 2,…, K)…………………………………………………… 84
Figure 3.43: The error function $E_2 = E_2\left(\alpha_{i,0}^{(k)}\right)$ in terms of the angle-of rotation $\alpha_{i,0}^{(k)}$ by using the GMEDS$_1$ for various values of N_i……………………………………… 88
Figure 3.44: The autocorrelation function $r_{\mu_i\mu_i}(\tau)$ of the reference model in comparison to the autocorrelation function $\tilde{r}_{\mu_i\mu_i}(\tau)$ of the simulation model by using the MEDS with $\alpha_{i,0}^{(k)} = 0$ and the GMEDS$_1$ with $\alpha_{i,0}^{(k)} = \pm\pi/120$ (N_i=20, k=K=4) 88
Figure3.45:The autocorrelation function $r_{\mu_i\mu_i}(\tau)$ of the complex Gaussian process of the reference model in comparison with the corresponding autocorrelation function $\tilde{r}_{\mu_i\mu_i}(\tau)$ of the simulation model designed by using the MEDS with $\alpha_{i,0}^{(k)} = \pm\pi/160$ and $\alpha_{i,0}^{(k)} = \pm\pi/480$ ($N_1 = N_2$,= 20, K = 4)………………… 89
Figure 3.46: Simulated uncorrelated Rayleigh fading waveform by using GMEDS$_1$ (f_{max}= 91 Hz, $N_1 = N_2 = 20$, K = 4)………………………………………………… 89
Figure 3.47: Frequency hopping Rayleigh fading channel simulator……………………… 91
Figure 3.48: The diagram for the implementation of a Frequency hopping channel simulator……………………………………………………………………… 92
Figure 3.49: Simulation of the received envelops $\tilde{\varsigma}(t)$ for the rural area environment: (a) Without frequency hopping (b) with frequency hopping………………….. 93
Figure 3.50: Autocorrelation function $\gamma_{\mu_i\mu_i}(\tau)$ of the reference model in comparison with the autocorrelation function $\tilde{\gamma}_{\mu_i\mu_i}(\tau)$ of the simulation model (MEDS with N_i = 20, f_{max} = 91 Hz and σ_0 = 1)…………………………………………… 93
Figure 3.51 : Cross correlation function obtained for COST207 rural area profile (a = 0.1086µs, f_{max} = 91 Hz, σ_0^2=1) (a) Reference model (b) Simulation model for N_i = 20……………………………………………………….... 94
Figure 3.52: Cross-correlation function of reference model comparison with the simulated model (N_i = 20) for COS207 Rural area profile (a = 0.1086µs, σ_0^2=1)………. 94
Figure 3.53: Cross-correlation function of reference model for COST 207 Rural area profile (a = 0.1086µs, f_{max} = 91 Hz, σ_0^2=1)………………………………….. 94

List of Tables

Page

Table 2.1: Parameter value under four environmental conditions specified by COST 207. 38
Table 3.1: Optimized component of the parameter vector $\mathbf{\Omega}^{(m)}$ (m = 1, 2) of the reference model for an equivalent satellite channel in a suburban and urban area…………………….. 48
Table 3.2: Parameters of the dynamic (2+2)-state model for an equivalent satellite channel in suburban and urban area…………………………………………………………. 48

CHAPTER 1

INTRODUCTION

Mobile radio communication is one of the highly contested fields that draw continuing interests among researches and engineers in the world. Currently, mobile radio communication field is developing grammatically and most of the researchers continuously trying to improve quality of the systems with high data rates economically. Most of the time prototype and mathematical models are used to improve the new system features than it implement in hardware means. These simulation models assist vastly in implementation of the systems as well as help for the research work based on mobile radio communication. Recently, an important emergence of mobile systems has been given opportunities to researchers' new ways and means to achieve good performance and high communication quality.

With the number of users of such systems is rapidly growing, a compromise between quality and capacity should be taken into consideration in developing the new systems. The initiation 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 and techniques of transmission. So far the capacity of the system regarding the number of users is limited and this leads to dissatisfy all demands. Therefore, a new generation of mobile radio systems were begun. It was 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 make challenges to invent new systems. As a result, third generation mobile radio systems were developed in early 2000. The prime objective of 3G mobile system is to achieve a fully integrated digital mobile terrestrial communication network that offers voice, data, and multimedia service at any time everywhere in the world with seamless global roaming. In the future, a universal new system, called the fourth generation mobile radio systems, will be of interests to scientists. The goal of this generation is to consolidate the broadband mobile services. 4G system will offer enhanced peak data rate of 100Mbit/s for high mobility device and 1Gbit/s for low mobility devices [3]. Principle technologies used in 4G systems include multiple-input multiple-output (MIMO) technique, turbo coding techniques, adaptive modulation and errorcorrecting coding schemes, orthogonal frequency division multiple access (OFDM) techniques and fixed relaying and cooperative relaying networks.

Currently, numbers of standardization bodies supported by industry and research institutions are trying to establish new system standards for future high speed wireless local area network and 4th generation wireless system employing new techniques. Day by day the system grows with high level of techniques in the field and majority deal with quality and cost reduction. Mobile radio communication was investigated theoretically and experimentally in order to build up these systems and concepts.

The term mobile radio channel is the physical medium that is used to send the signal from the transmitter to receiver. The channel modelling is an important part of mobile radio communication. These developed channel simulators help us to study how real world mobile communication systems behave in effective and efficient manner. The task of channel modelling helps us to identify, analyse and model the main characteristics properties of the channel and to create thus a basis for the development, optimization, and examine the digital transmission system.

1.1 BASIC CONCEPTS

Channel simulators are important depend on the parameter optimization, and the performance analysis of mobile communication systems. Obviously, an acceptable model avoids unexpected degradations whenever it initiate to implementation. In addition, especially in mobile radio communication, knowledge of the channels is very important when implement the new system and solutions. The diagram below depicts a typical mobile radio propagation scenario.

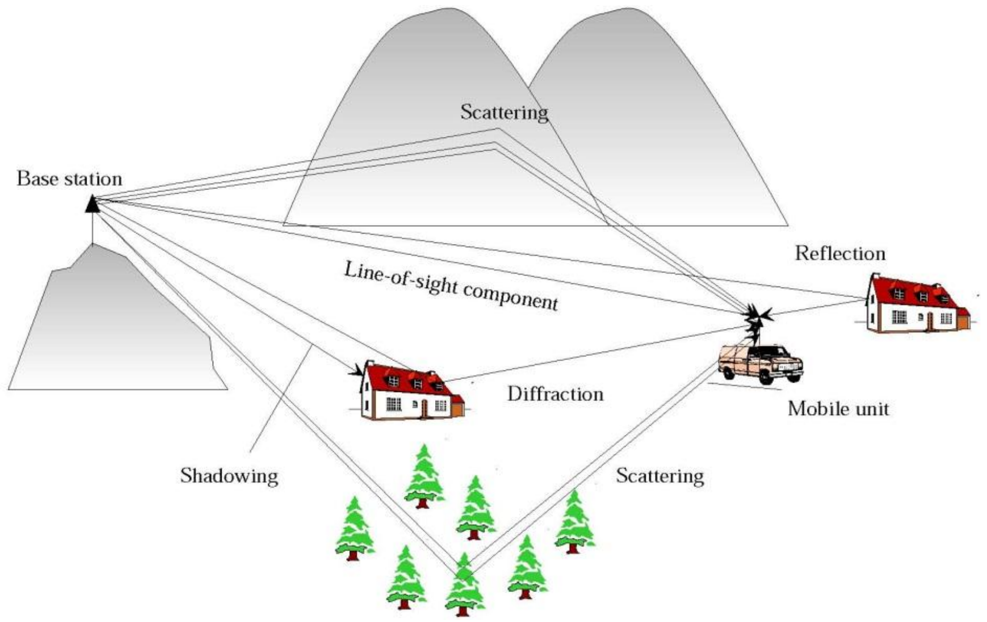


Figure 1.1: Typical mobile radio propagation scenario

Signal travelling in wireless link may be subjected to reflection, diffraction, shadowing and scattering as well as its power alter due to channel response. If the power of signals fluctuates considerably (drop down) under this scenario, channel is said to be faded channel.
Fading can be defined as;

1. Modulated signal subjected to deviation of attenuation while propagated through wireless channel due to response of channel.
2. Unpredictable, irregular and random change in magnitude and phase.
3. Fading is result of different reasons, Multi path fading, interference, shadowing, path loss etc.

Classical methods of modelling the fading behaviour of mobile radio channel are characterized by modelling of the transmission link between a base station and a mobile station. In the early stages of channel modelling, the aim was to characterize the statistical properties of real-world channel mainly with respect to the probability density function of the channel's envelope. The time characteristics, and later the frequency characteristics of the mobile radio channel, have been included in the design procedure only to a limited degree. Modern methods of channel modelling pursue to characterize the envelop fading regarding the first order statistics and the second order statistics, which include the level crossing rate and average duration of fade.

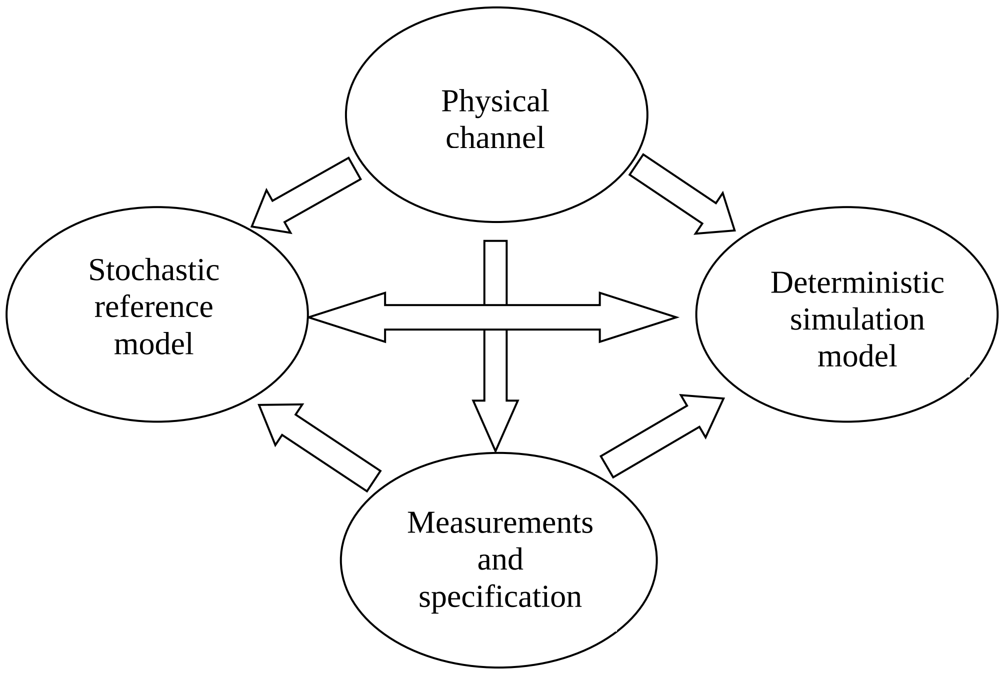

Physical channel
Stochastic reference model
Deterministic simulation model
Measurements and specification

Figure 1.2: Relationship between the physical channel, the stochastic reference model, the deterministic simulation model and measurement or specification [2]

During the process of implementation of the mobile radio channel simulator, some of the following scientifically approved steps are required to get a fitted model. First of all, a concrete phenomenon or case must be presented. The simulation model is usually derived from the underlying reference model or directly from measurements of physical channel. Then, measurements should be taken in order to specify the physical aspects. The usefulness and importance of the reference model and corresponding simulation model is ultimately judge on how well their statistical properties can be matched to the statistical properties of specified or measured channels. Following primary goals are illustrated in Figure 1.2. It shows relationship between the physical channel, the stochastic reference model, and the derivable simulation model. These relationships will be employed in proposed channel simulator implementation.

The simulation model is usually derived from the underlying reference model or directly from physical channel as mention in above paragraph in order to get a simulations model that emulates accurately the behaviour of the fading channel. Finally, we compare the simulation model with the reference model. Thus, we try to minimize the error modelling as much as possible to achieve the best match. These relationships will accompany for all channel models.

After observing the natural phenomena and developed model, analyse are done to seek which can be used to solve the problems. For instance, mathematical and physical backgrounds were found to be important and the modelling step requires the reference model, exact model and simulation model in order to examine 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 which controls the performance of the entire system.

Mainly, most of the channel model simulators are implemented based on the sum-of-sinusoids principle which is first introduced by Rice and then 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. Here, the Gaussian process is structured as follow:

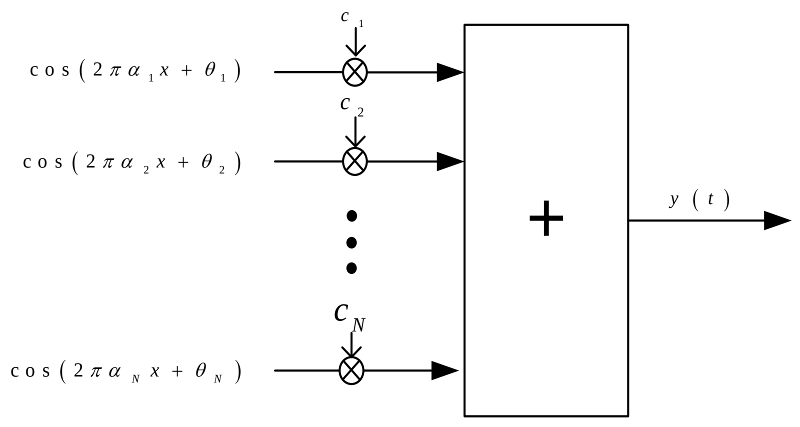

c_1
$\cos(2\pi\alpha_1 x+\theta_1)$
c_2
$\cos(2\pi\alpha_2 x+\theta_2)$
$+$
$y(t)$
c_N
$\cos(2\pi\alpha_N x+\theta_N)$

Figure 1.3: An example of sum-of-sinusoids principle

As it's well aware, when using the Rice method, we assume that numbers of sinusoids are 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 in 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 such as the exigency of ideal filter.

In our thesis; the simulation models are based on the principle of deterministic channel modelling. The principle of deterministic channel modelling consists of the following steps to proceed

Step 1: Starting point is the reference model; based on one or several Gaussian processes, each with prescribe autocorrelation function.

Step 2: Derive a stochastic simulation model from the reference model by replacing the Gaussian process by sum-of sinusoid with fixed gain, fixed frequency, and random phases.

Step 3: Determine a deterministic simulation model by fixing all model parameters of the stochastic simulation model including the phases.

Step 4: Compute the model parameters of the simulation model by fitting the relevant statistical properties of the deterministic (or stochastic) simulation model to those of reference model.

Step 5: Perform the simulation of one (or some few) sample functions (Deterministic processes).

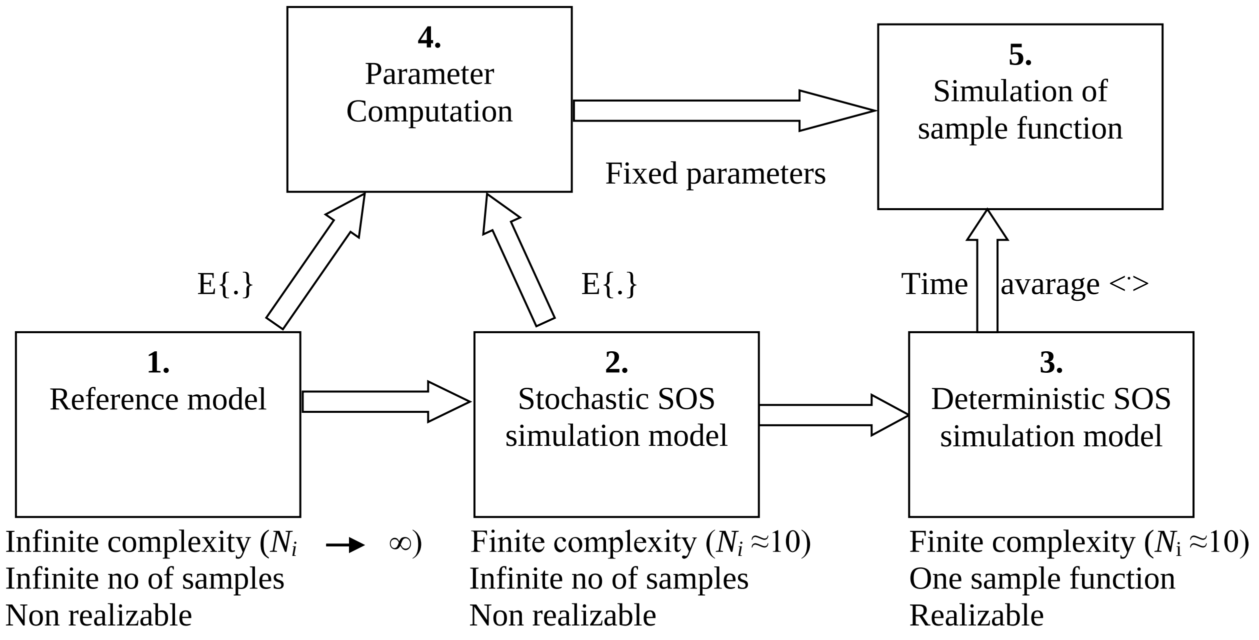

4.
Parameter Computation
5.
Simulation of sample function
Fixed parameters
E{.}
E{.}
Time avarage <·>
1.
Reference model
2.
Stochastic SOS simulation model
3.
Deterministic SOS simulation model
Infinite complexity ($N_i \rightarrow \infty$)
Infinite no of samples
Non realizable
Finite complexity ($N_i \approx 10$)
Infinite no of samples
Non realizable
Finite complexity ($N_i \approx 10$)
One sample function
Realizable

Figure 1.4: The steps of deterministic channel modelling

The principle of deterministic channel modelling can be used to model following type of models;

1. Multiple uncorrelated Rayleigh fading channels.
2. Multiple cross correlated Rayleigh fading channels.
3. Frequency non selective channels (Rayleigh, Rice, Suzuki, etc.)
4. Frequency selective channels (COST 207).
5. Frequency hopping fading channels.
6. Develop of MIMO channels etc.

Further explanation of these concepts is noticed in [2] chapter 4.

1.2 REQUIREMENTS

This thesis is based on the implementation of the channel models which based on reference [1, 2], papers and publications done over the last few years as listed under the references. In order to implement the toolbox fitted for mobile channels modelling, an exact solution is to 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 [4].”

MATLAB features a family of add-on application-specific solutions called toolboxes. The MATLAB system consists of four main parts:

1. Development Environment. This is the set of tools and facilities that help users 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.

2. 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.

3. 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.

4. MATLAB Applications Program Interface (API). This is a library that allows user 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.[4]

1.3 THESIS OUTLINES

Chapter one dedicates to introduce about the master thesis work as well as to demonstrate the methods which we used to implement the channel simulators. In the last part of the first chapter we conclude brief summary of some useful terms that is used in this thesis.

Chapter two emphasise onto all methods and procedures for calculating the parameters of the simulation models. The first part concerns the frequencies' non-selective channels. Under this section we worked on two sub division based on parameterization for the sum-of-sinusoids channel model and parameterization for the sum-of-cisoids channel model. The second part focused on the frequencies' selective channels. Here we describe how some of the statistical properties such as ACF and PSD behave according to the different model and theories.

Chapter three is deals with implementation of mobile radio fading channel simulators under different scenarios. The first section concerns with implementation of the real world land mobile satellite (LMS) channel model simulator. Then the second section focuses on implementation of the spatial shadowing channel simulator. There, briefly description of fundamental concept as well as some characteristics is given. In addition to that describe how we implement the generation of the process and functions such as autocorrelation function (ACF), level crossing rate (LCR), etc. for testing the fitting of the simulation model against the reference model. After that, in third section we introduced multiple input multiple output (MIMO) channels. At first, we implemented the MIMO one-ring model simulator. Then we extend to the 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, cross correlation function (CCF), etc. After modelling of the MIMO channel models; we describe how to implement the multiple uncorrelated Rayleigh channel (MURC) simulator. Some of the second order statistical properties are described. Finally we describe modelling of the hopping mobile radio channel simulator. All above mentioned channel simulation model behaviours were tested and compared with the reference model to ensure the correctness.

Implementation of the real world land mobile satellite channel model simulator and the spatial shadowing channel simulator were done by D. S. De Silva. Implementation of the Multiple input multiple output channels, multiple uncorrelated Rayleigh channel and hopping mobile radio channel simulators were done by H.U. Aboklosa.

The last chapter is entirely dedicated to the conclusions and recommendations based on 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 recommendations.

CHAPTER 2

PARAMETERS COMPUTATION METHODS

In this chapter, most of the parameters computation methods founded in literature and issued from many articles and publications are presented. Each channel is characterized by a set of parameters that should be determined.

Parameter computations were done for two channels types; non-selective-frequency and selective-frequency. Firstly 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; frequency-non-selective channels and frequency-selective channels.

In the first section of the parameters computation is done under two categories. First section; Parameter computation is done for the Sum-of-sinusoids channel model. Under that, the intention is to compute the parameters of each channel type using many methods such as MEA, MED, MSEM, MCM, LPNM, MEDS, RMEDS, MEDS-SP [2]. Secondly the parameter computation done for the Sum-of-cisoids channel model. Three methods namely extended method of exact Doppler spread (EMEDS), Lp-norm method (LPNM) and Generalized method of equal area (GMEA) were considered for the parameterization of the sum-of-cisoids channel model.

2.1 FREQUENCY NON-SELECTIVE CHANNELS

If the duration of a modulated symbol is much greater than the time spread of the propagation path delay, it is defined as frequency non selective fading and such channel called frequency-non-selective channel. Generally, in analysing the frequency-non-selective channels, emphasis were put onto the factors that are related to Doppler effects, such as Doppler frequencies and Doppler coefficients. As a result, stated in below, different methods to compute these parameters were considered. First, the principles of these methods were explained briefly.

Afterwards, we depict our implementation approach and finally we show some explicit results for both Sum-of-sinusoid channel model and Sum-of-cisoids channel model.

2.1.1 Parameterization of Sum-of-sinusoid Channel Model

In this section, the principle of each method and the way it was implemented is presented. Further details can be founded in [2] chapter 5. Parameter computations were done under methods called MEA, MED, MSEM, MCM, LPNM, MEDS, RMEDS and MEDS-SP.

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 same distances [5]. The property is achieved by defining the discrete Doppler frequencies $f_{i,n}$ as

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

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:

$$c_{i,n} = 2\sqrt{\int_{f \in I_{i,n}} S_{\mu_i \mu_i}(f)\,df} \tag{2.3}$$

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

Jakes power spectral density:

The frequency range of the Jakes power spectral density is limited to the range $|f| \leq f_{\max}$. Using Jakes's approach [2], 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 [2], chapter 5, section 5.1.1.

2.1.1.2 Mean-Square-Error Method (MSEM)

The mean-square-error method (MSEM) is based on the minimization of the mean-square error. Model parameters of $c_{i,n}$ and f_{in} are computed minimizing the $E_{r_{\mu_i\mu_i}}$;

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

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 [2, 5]. Then, we obtain:

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

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;

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

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 [2], chapter 5, section 5.1.2.

2.1.1.3 Method of Equal Areas (MEA)

The method of equal areas (MEA) is to obtain that the obtained Doppler frequencies should fulfil the condition where the area under the Doppler power spectral density $S_{\mu_i\mu_i}\left(f\right)$ must be equal to $\sigma_0^2/2N_i$ within $f \in \left[f_{i,n-1}, f_{i,n}\right]$. Then, it was found the Doppler frequencies depending on the number of scatters 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 scatters.[1,2,5] 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; Doppler frequencies of $f_{i,n}$;

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

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.

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

As with the Jakes examples, the Doppler coefficients $c_{i,n}$ are given by (2.18). More explanation about this method is presented in [2], 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 $p_{\mu_i}(f) = \frac{1}{\sigma_0^2} S_{\mu_i\mu_i}(f)$. As result, the Doppler frequencies are obtained from:

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

Where

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

u_n is a random variable uniformly distributed over $(0, 1]$. $c_{i,n}$ 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, the frequencies 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) also follows 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 $f_{i,n}$.

Initially $f_{i,n}$ is consider to obtain 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 it 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. In this research two different approaches for the definition of this method are presented.

Jakes power spectral density:

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

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

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

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

2.1.1.7 Jakes Method (JM)

The jakes method has been developed exclusively for Jake power spectral density. More explanation about this method is presented in [2], chapter 5, section 5.1.7.

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

The Doppler frequencies are shown here

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

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 Method of Exact Doppler Spread (RMEDS)

The Randomized method of exact Doppler spread RMEDS is a combination of the Monte Carlo method and the MEDS in order to get higher performance for a small number of sinusoids. In [6], 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

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

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, a progressive step has been taken to divide the aimed constellation of locations of scatterers into sub-constellation as shown in the figure below.

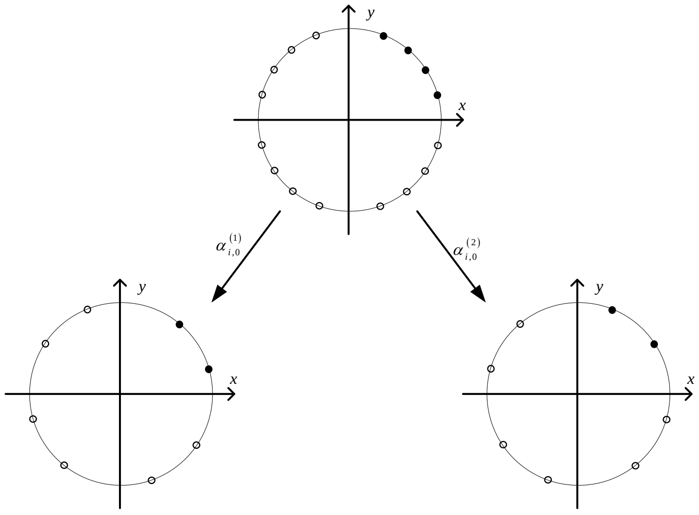


Figure 2.1: Set-partitioning principle

The Figures 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 [7], 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

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

2.1.1.10 Implementation

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

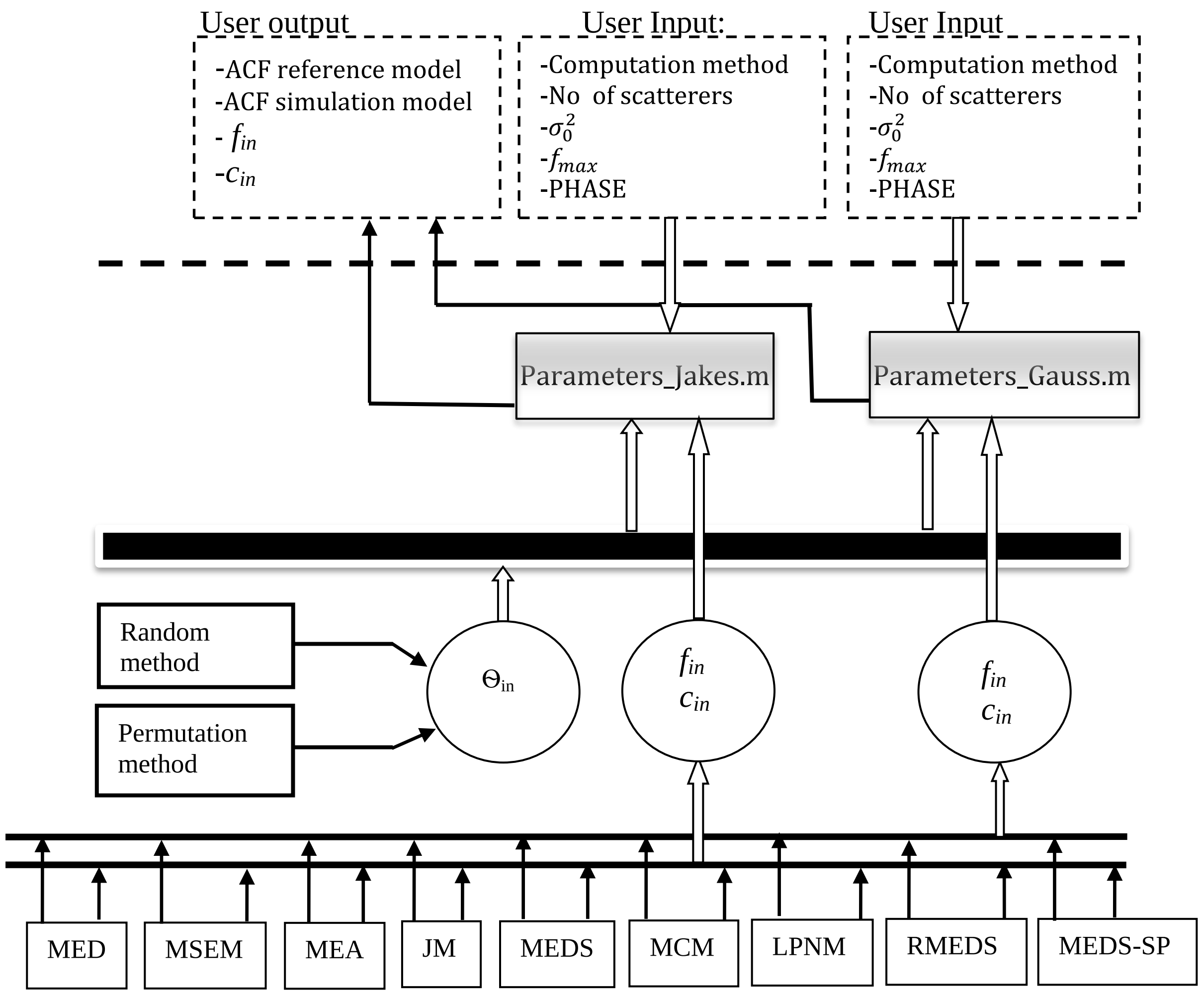


Figure 2.2: The implementation diagram of the parameters computation methods for the frequency-non-selective channel

The Figure 2.2 presents the flow of implementation proposed 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, user has to arrange the number of scatterers, the variance σ_0^2, and the frequency Doppler shift. In the last phase, user is allowed 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, RMEDS, 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.

2.1.1.11 Results

Some results are presented herein to show the performance of each parameter in computation method. Further evaluations are carried out each method by comparison with others. The plots were taken for N = 25, $\sigma_0^2 = 1$, f_{max} = 91 Hz 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.

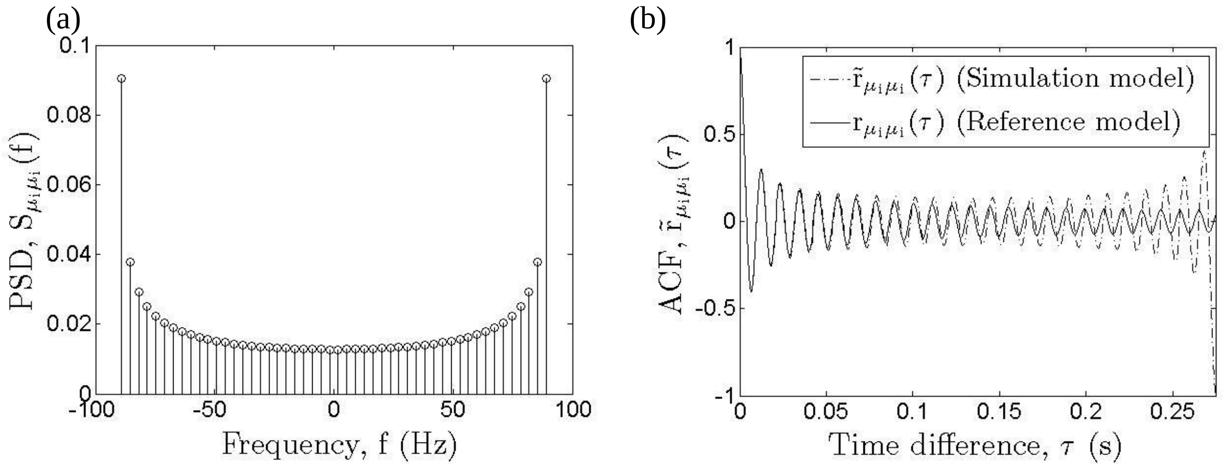


Figure 2.3: (a) PSD and (b) ACF of Jakes model using MED method (N = 25, $\sigma_0^2 = 1$, f_{max} = 91 Hz)

It has been revealed that the method of equal distances doesn't fulfil the constraint of a good match between the simulation model and reference model as shown in the Figure 2.3 (b). Simulated ACF is deviate from the behaviour of the reference model.

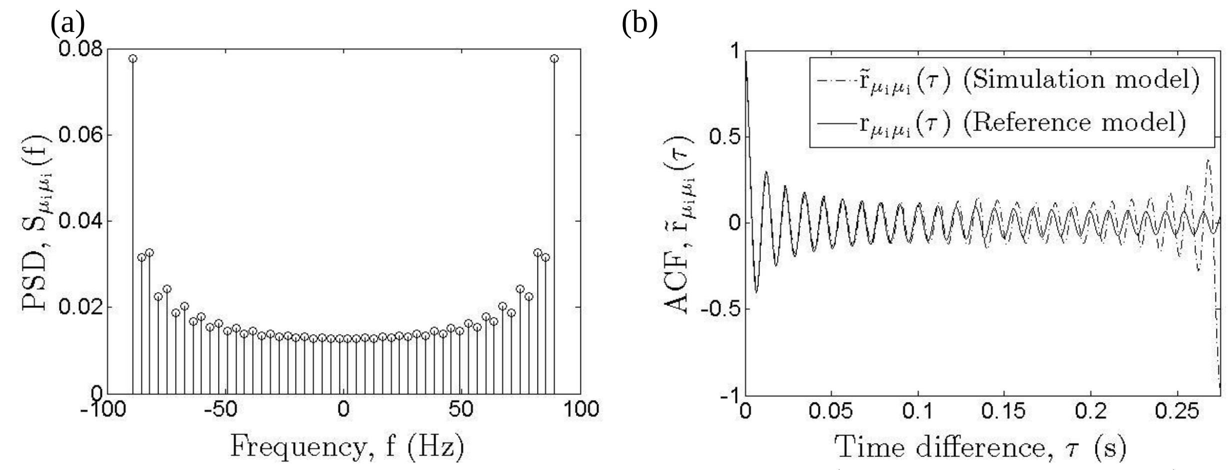


Figure 2.4: (a) PSD and (b) ACF of Jakes model using MSEM method $(N = 25, \sigma_0^2 = 1, f_{max} = 91\ Hz)$

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

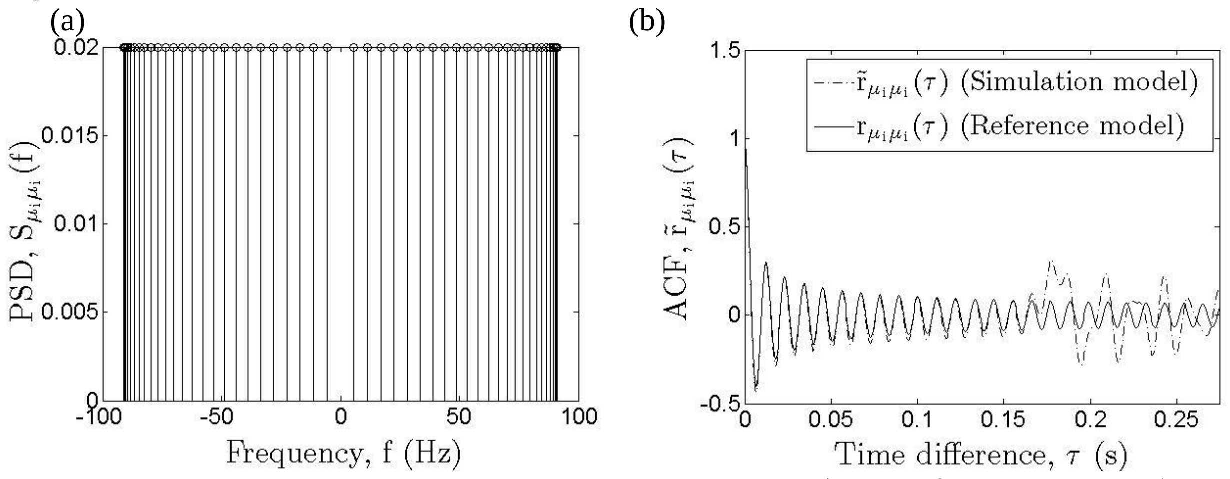


Figure 2.5: (a) PSD and (b) ACF of Jakes model using MEA method $(N = 25, \sigma_0^2 = 1, f_{max} = 91\ Hz)$

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

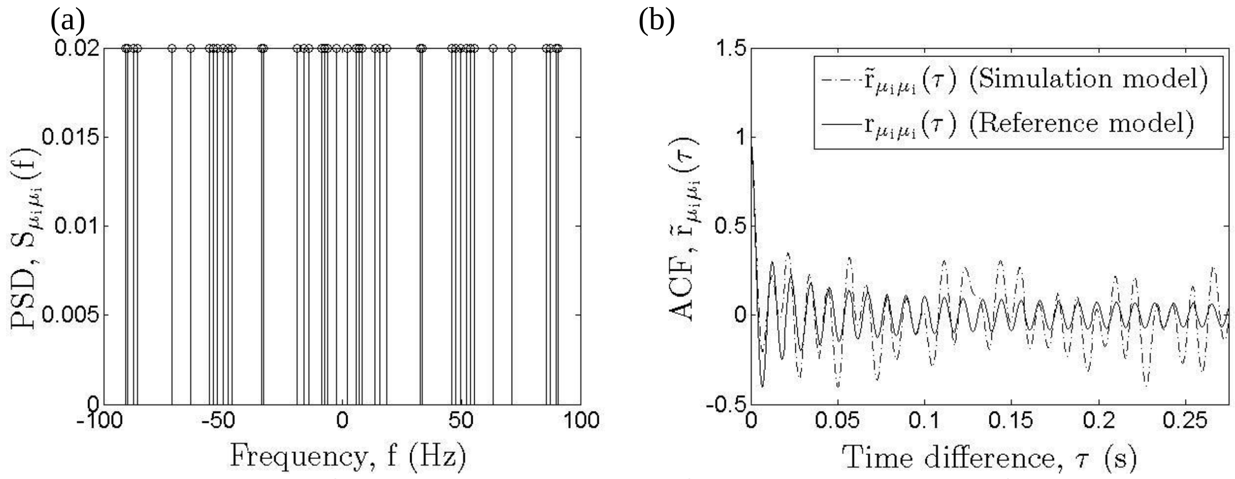


Figure 2.6: (a) Power density function and (b) Autocorrelation function using MCM method for Jakes model $(N = 25, \sigma_0^2 = 1, f_{max} = 91\ Hz)$

The Figure 2.6 is given by just one realization of the MCM method. It seems that more deviation of the simulated model form the reference model of the ACF as well as PSD. However, this method requires several realizations so that we approach the desired fitting.

(a) (b)

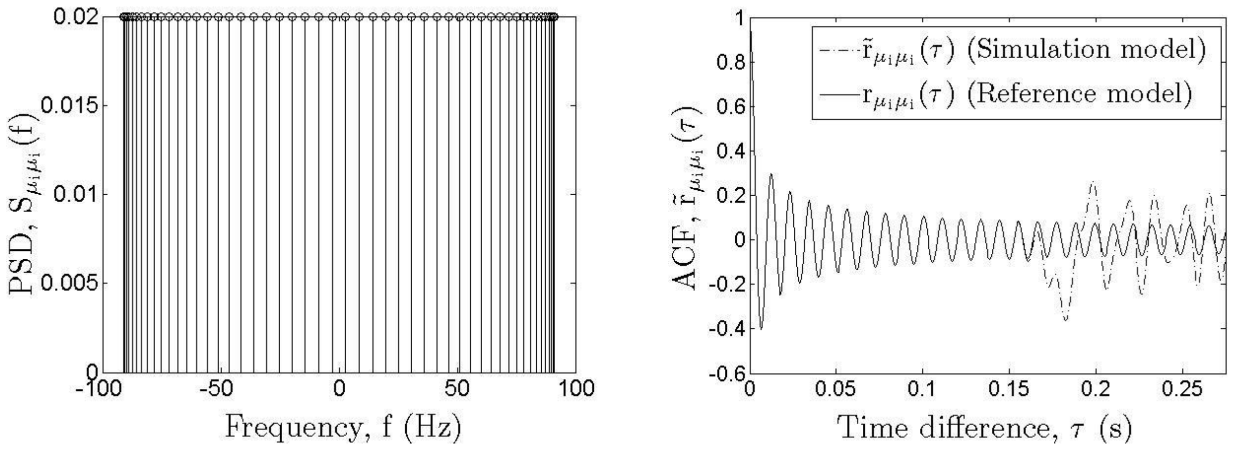


Figure 2.7: **(a)** PSD and (b) ACF of Jakes model using LPNM method $(N = 25, \sigma_0^2 = 1 , f_{max} = 91\ Hz)$

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

(a) (b)

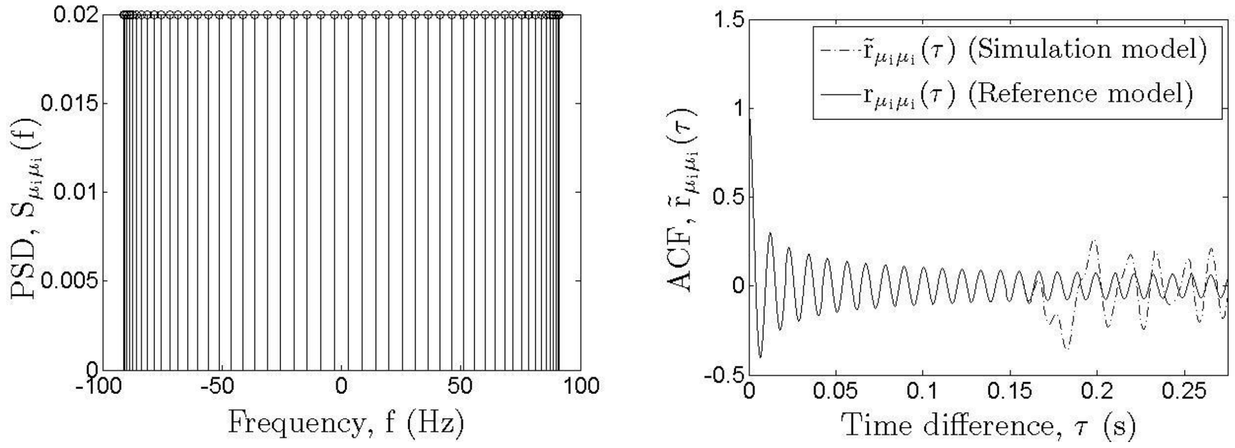


Figure 2.8: (a) PSD and (b) ACF of Jakes model using MEDS method $(N = 25, \sigma_0^2 = 1 , f_{max} = 91\ Hz)$

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

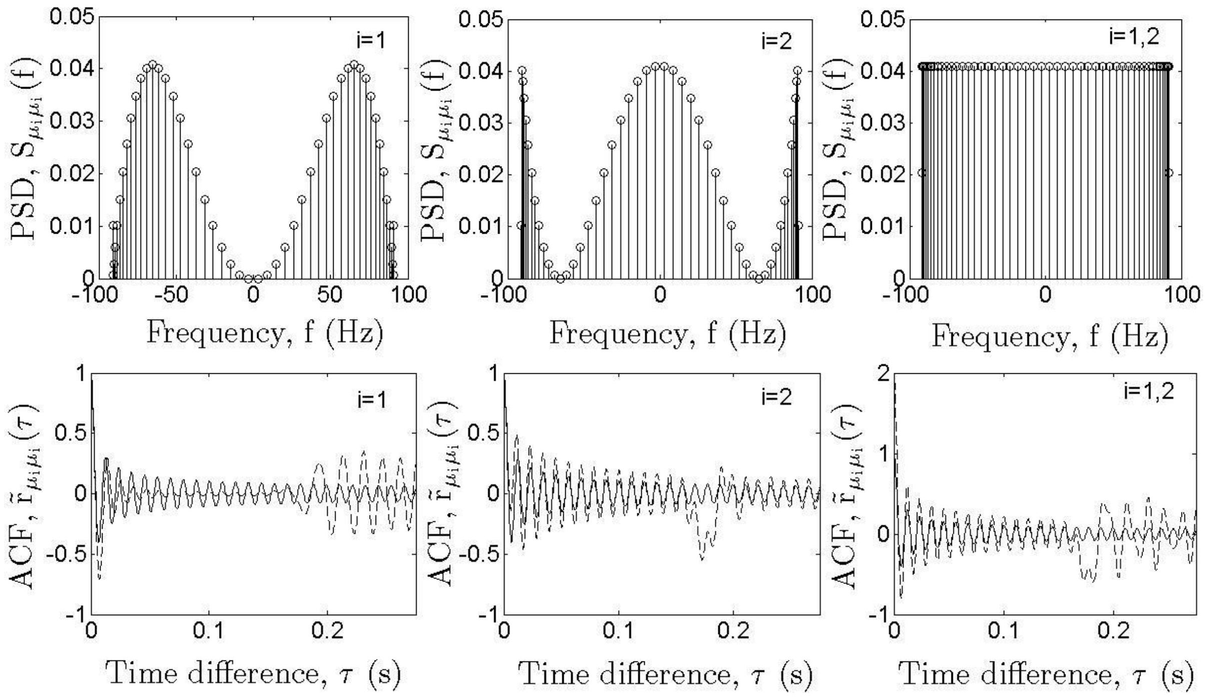


Figure 2.9: PSD and ACF of Jakes model using JM method $(N = 25, \sigma_0^2 = 1, f_{max} = 91\ Hz, f_c = 76\ Hz)$

The JM method is supposed to have much iteration in order to compensate the error caused by one realization. It is understood that sum-up two realizations as depicted in Figure 2.9 leads for good performance.

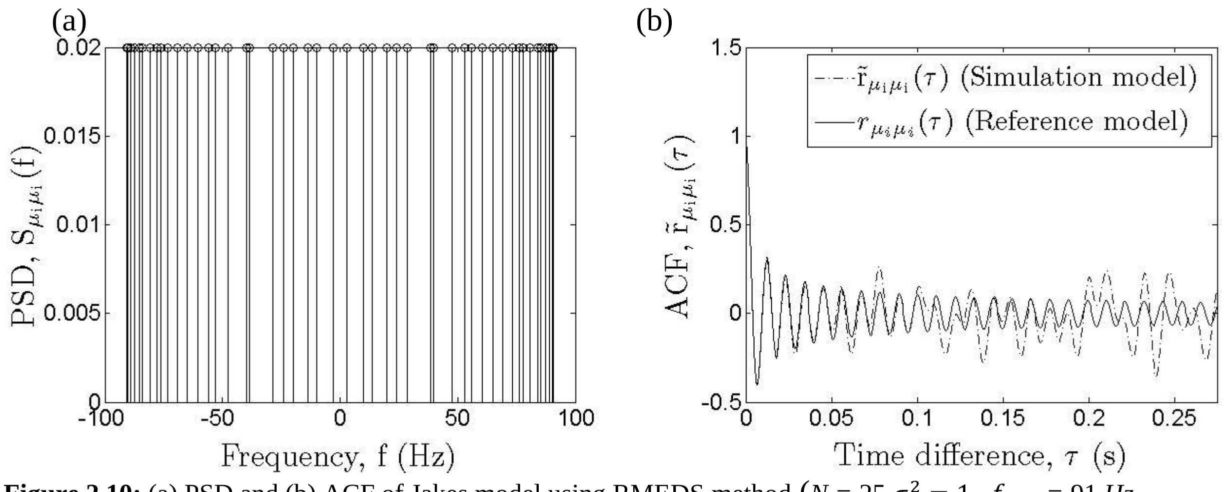


Figure 2.10: (a) PSD and (b) ACF of Jakes model using RMEDS method $(N = 25, \sigma_0^2 = 1, f_{max} = 91\ Hz,.$ $f_c = 76)$

In this particular research, the RMEDS 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 some behaviour as the simulation model until around τ_{max} /8.

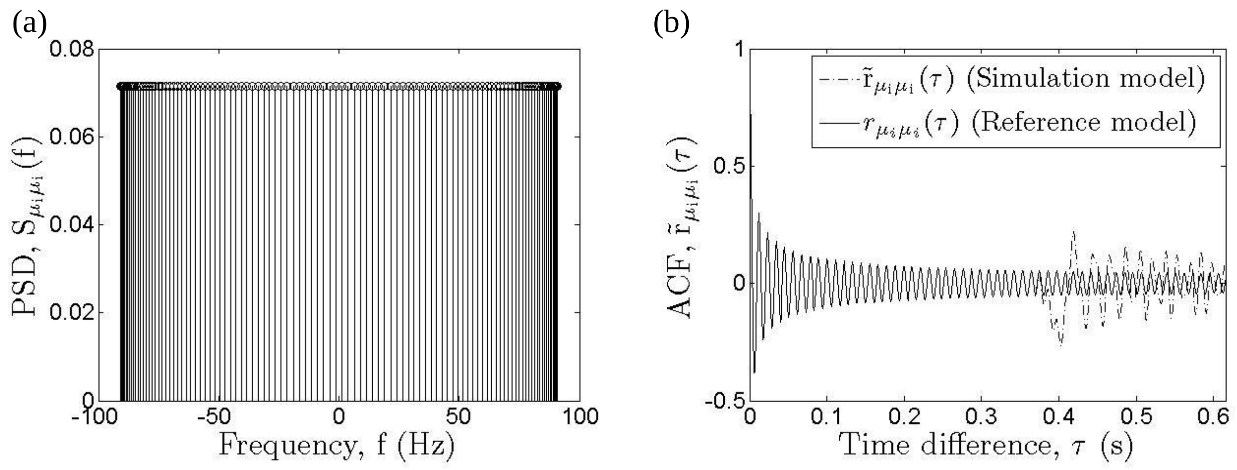


Figure 2.11: (a) PSD and (b) ACF of Jakes model using MEDS-SP method (N = 25,$\sigma_0^2 = 1$, f_{max} = 91Hz, number of sub-constellation = 8)

The MEDS-SP is considered to be the best method altogether until $N/2$. The main advantage of this method is the time saving and the efficiency with the reduction of number of iteration for the computation parameters.

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=76 Hz and for different methods.

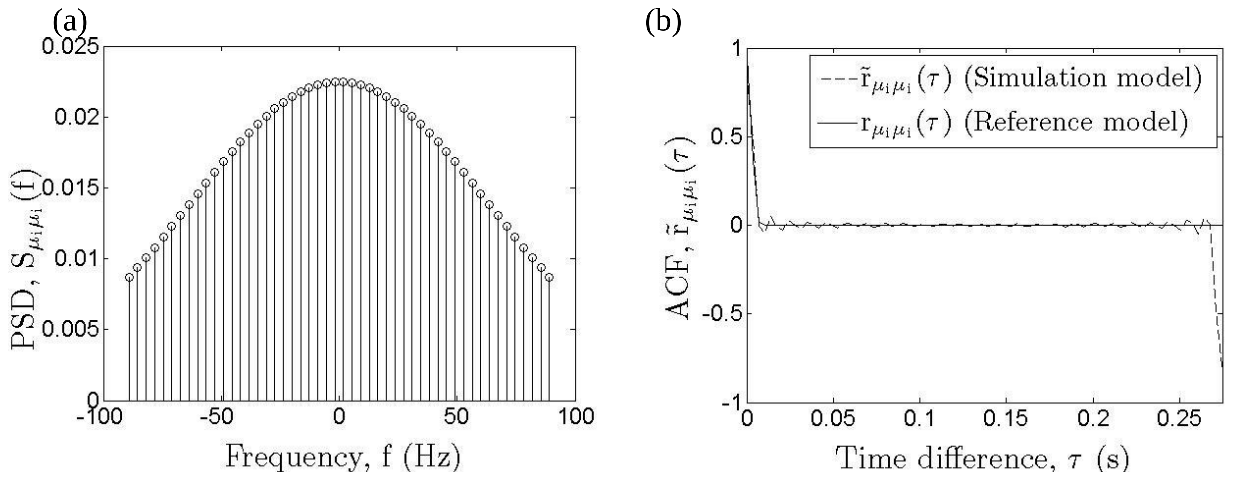


Figure 2.12: PSD and ACF of Gauss model using MED method (N = 25,$\sigma_0^2 = 1$, f_{max} = 91 Hz, .f_c = 76 Hz)

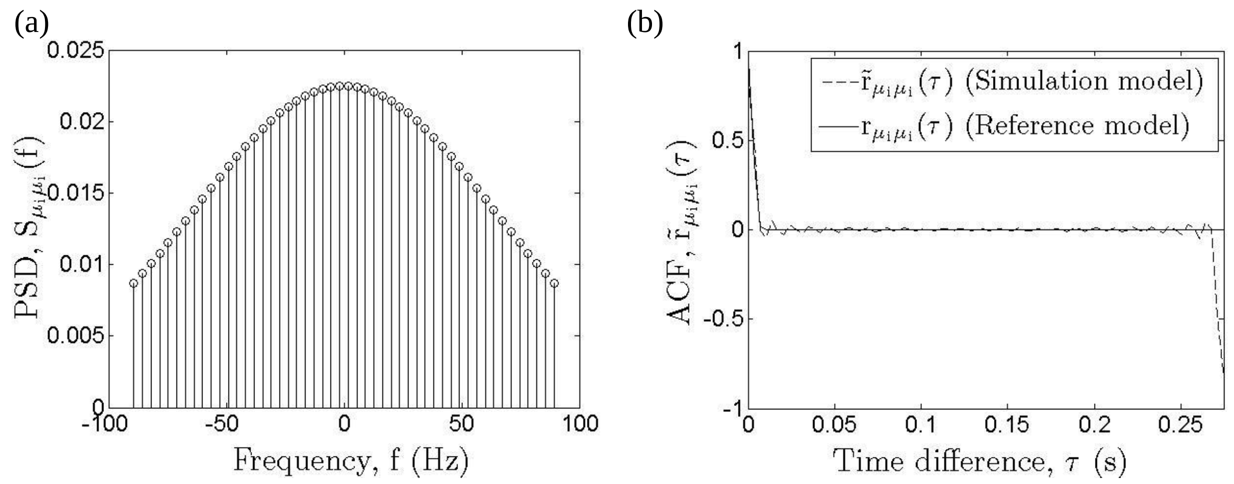


Figure 2.13: (a) PSD and (b) ACF of Gauss model using MSEM method $(N = 25, \sigma_0^2 = 1$, f_{max} = 91 *Hz*, f_c = 76 *Hz)*

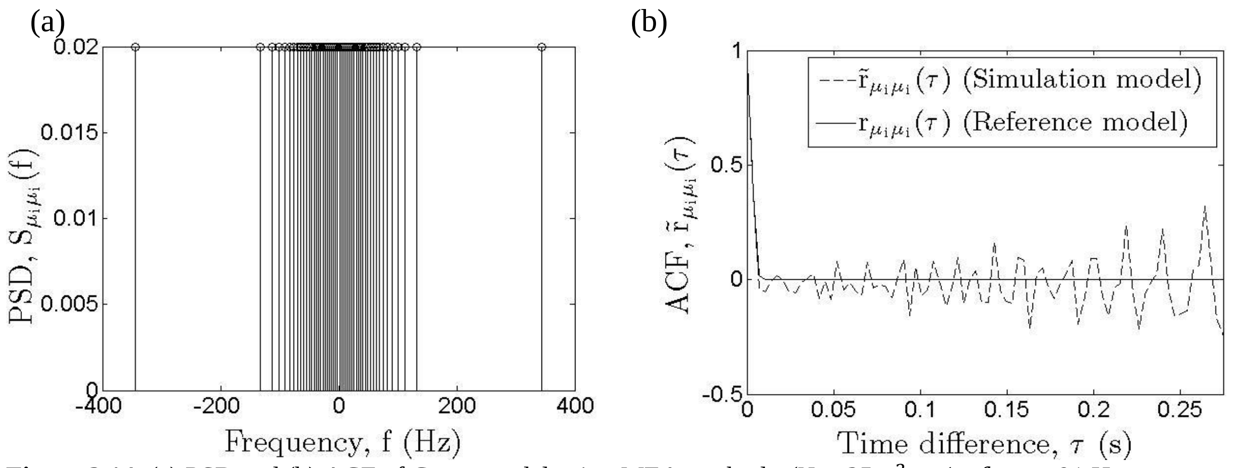


Figure 2.14: (a) PSD and (b) ACF of Gauss model using MEA method *(N = 25,*$\sigma_0^2 = 1$, f_{max} *= 91 Hz,* f_c = 76*Hz)*

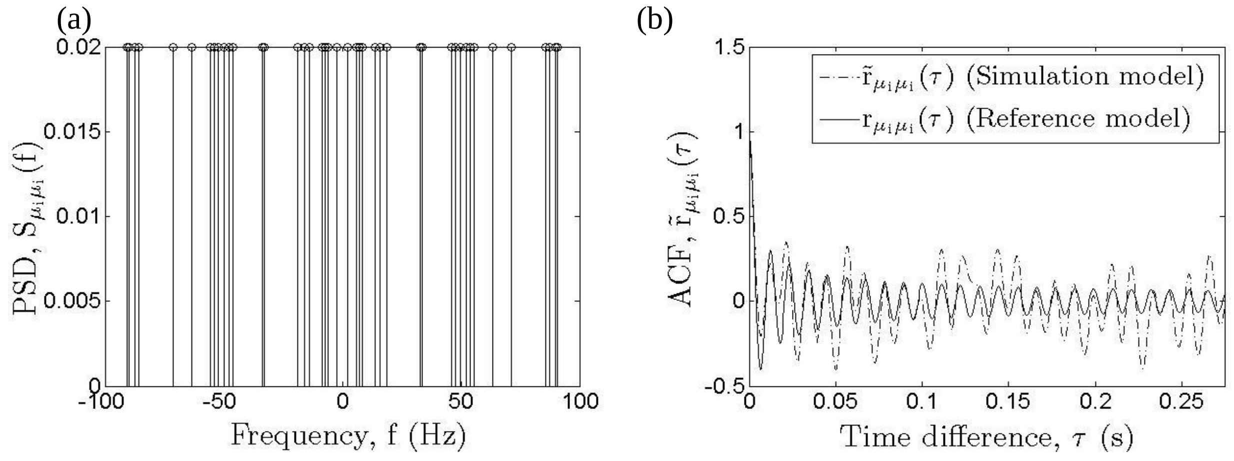


Figure 2.15: a) PSD and (b) ACF of Gauss model using MCM $(N$ = 25, $\sigma_0^2 = 1$, f_{max} = 91 *Hz*, f_c = 76 *Hz*)

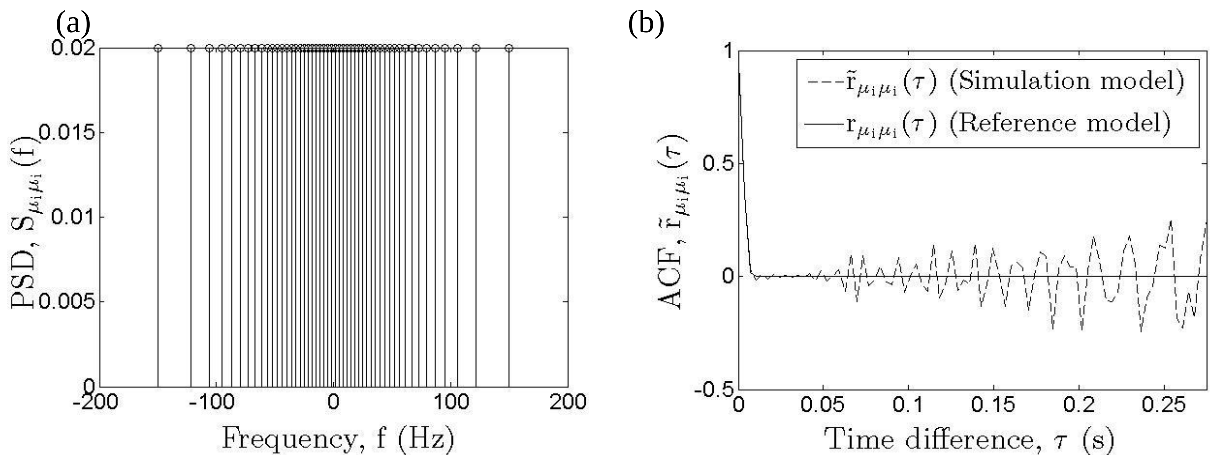


Figure 2.16: (a) PSD and (b) ACF of Gauss model using MEDS method (N = 25, $\sigma_0^2 = 1$, f_{max} = 91 Hz, f_c = 76 Hz)

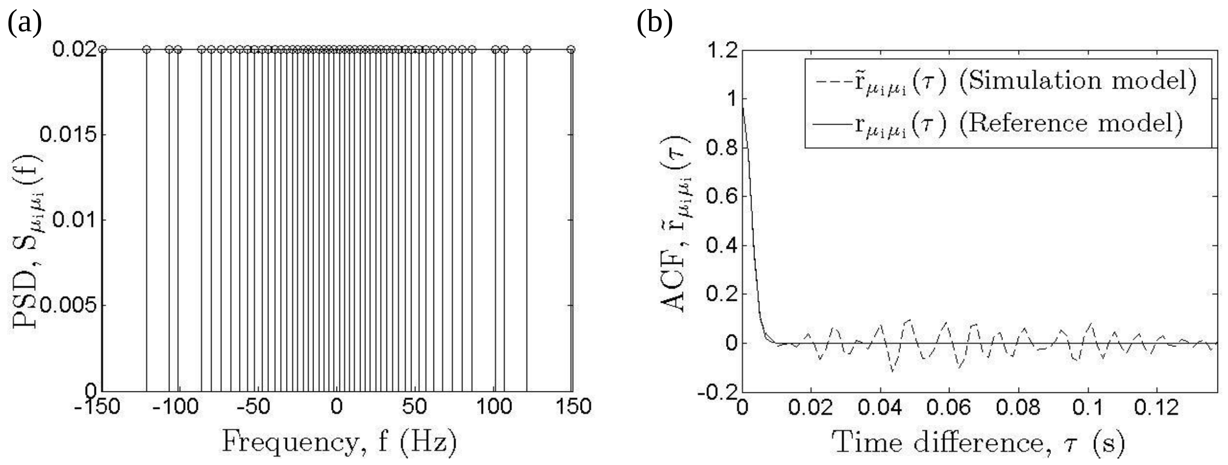
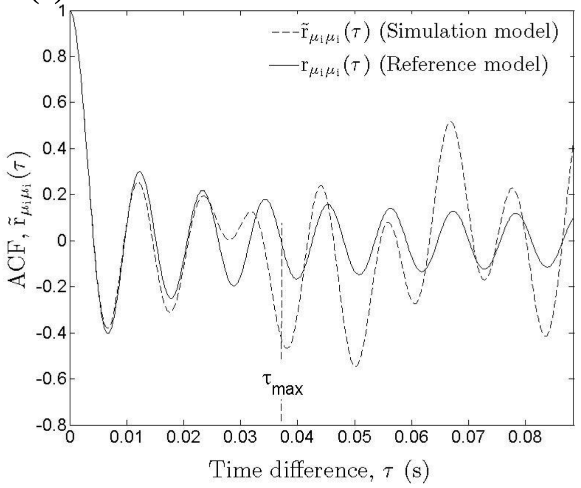


Figure 2.17: (a) PSD and (b) ACF of Gauss model using LPNM (N = 25, $\sigma_0^2 = 1$, f_{max} = 91 Hz, f_c = 76 Hz)

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

2.1.1.12 Comparison of Parameter Computation Methods

We compare the parameters computation methods (MED, MSEM, MEA, MCM, LPNM, MEDS, JM, RMEDS and 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 shows the best performance. Nevertheless, the MCM method for one realization and RMEDS 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 need to equip to address the drawbacks which occurred in the previous methods described above.

(a)


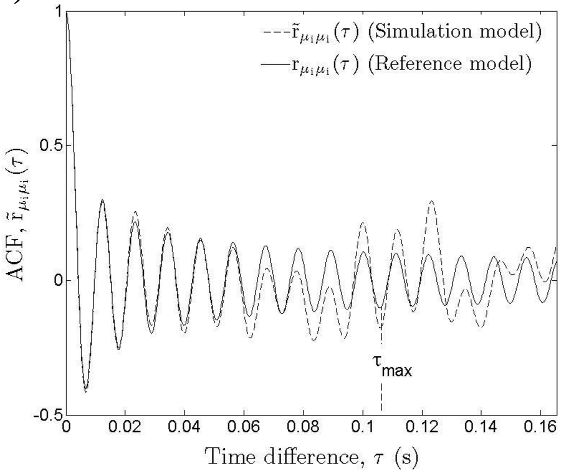
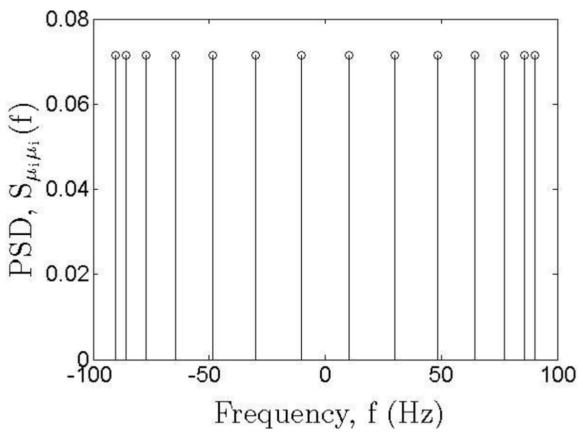
(b)


Figure 2.18: Autocorrelation function for (a) $N_i = 7$ and (b) $N_i = 21$ (RMEDS, Jakes PSD, f_{max} = 91 *Hz*, $\sigma_0^2 = 1$)

Figure 2.18 (a) and (b) shows that how ACF behave against change of the number of harmonic functions. When the number of harmonic function increases; it gives better performance in ACF compared with $N_i = 7$.

(a)


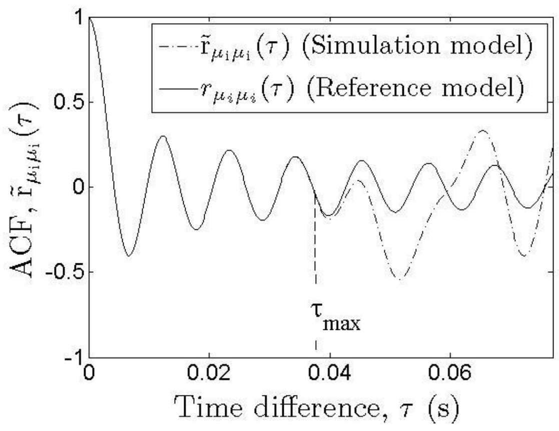
(b)


Figure 2.19: (a) PSD and (b) ACF of Jakes model using MEDS-SP method N = 7, f_{max} = 91 *Hz*, $\sigma_0^2 = 1$, *number of sub- constellation = 1*).

When we consider Figure 2.11 and Figure 2.19; Figure 2.11 shows better performance with 8 constellations (with MEDS-SP). Comparing the results obtained for the MEDS method with $N_i = 25$ and we merely reach the same performance with MEDS-SP for N = 7 and number of sub-constellation = 4. Therefore, the MEDS-SP considerably reduces the complexity of the system when it comes to realization. Figure 2.20 shows that how performance increase with K = 2 and 4. Here performance will increase with the number of constellation point for both methods of MEDS-SP and RMEDS.

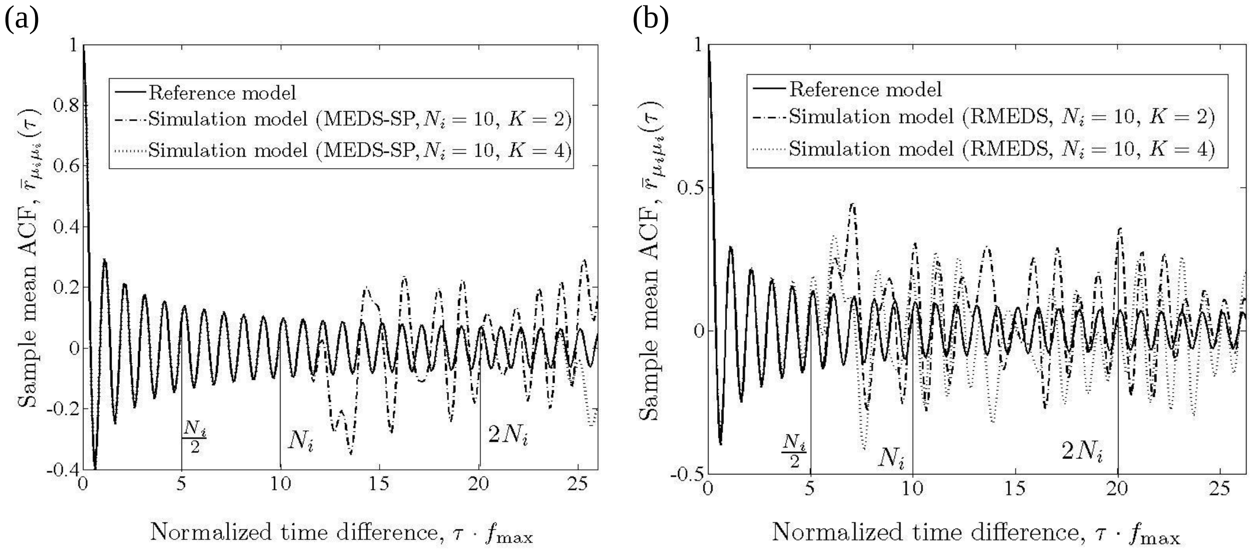


Figure 2.20: Sample mean autocorrelation function by using (a) the MEDS-SP and (b) the .RMEDS with N_i = 10 and various values of K

2.1.2 Parameterization of Sum-of-cisoids Channel Model

This section were focused on the parameter computation for sum-of-cisoids channel model and sum-of cisoids process generally represent as form of

$$\hat{\mu}(t) = \sum_{n=1}^{N} c_n \, e^{j(2\pi f_n t + \Theta_n)}, \tag{2.37}$$

Where c_n , f_n and Θ_n are the path gains, Doppler frequencies, and phase of the nth propagation path. Here we assume that Θ_n is i.i.d random variable which uniformly distributed over (0, 2π). The gain c_n and f_n are determined by using three methods namely the extended method of exact Doppler spread (EMEDS) the L_p-norm (LPNM), and the generalized method of equal area (GMEA).

2.1.2.1 Extended Method of Exact Doppler Spread (EMEDS)

The EMEDS was introduced as an extension of MEDS for computing the model parameters of channel simulator [2].

Doppler frequency f_n and gain c_n are specified as follows;

$$f_n = f_{max}\cos[\frac{2\pi}{N}\ (n\ -\ \frac{1}{4})\ ]\ , \tag{2.38}$$

$$c_n = \sigma_0\sqrt{\frac{2}{N}} \qquad \text{Where } n = 1, 2, 3,\ldots, N. \tag{2.39}$$

2.1.2.2 L_p-Norm Method (LPNM)

The L_p -norm method (LPNM) considers minimizing the error function presented below in order to get optimized Doppler frequencies and Doppler coefficients [2]. In terms of experimental point of view, when it sets $c_{i,n}$ to $\sigma_0\sqrt{2/N_i}$ and look for an optimized solution of

$$E_{p_{\mu_i}}^{(p)} = \left\{ \frac{1}{\tau_{\max}} \int_0^{\tau_{\max}} \left| r_{\mu_i\mu_i}(\tau) - \tilde{r}_{\mu_i\mu_i}(\tau) \right|^p dx \right\}^{1/p} , \quad p = 1, 2, \ldots \tag{2.40}$$

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

In first variant both Doppler frequency f_n and gain c_n are computed such that cost function

$$E^{(p)} = W_1 E_{r_{\mu\mu}}^{(p)} + W_2 E_{p\varsigma}^{(p)} \tag{2.41}$$

Is (2.41) minimized, where W_1 and W_2 are waighting factor and

$$E_{p\varsigma}^{(p)} = \{ \int_0^{\infty} \left| p_\varsigma(z) - \hat{p}_\varsigma(z) \right|^p dz \}^{1/p} \tag{2.42}$$

The second variant minimize $E^{(p)}$ and It considers only N-1 pairs of parameters f_n , c_n , where n= 1, 2, 3,…, N.

Parameter f_N and c_N obtained,

$$f_N = \frac{1}{c_N} \{ \frac{-\ddot{r}_{\mu\mu(0)}}{4\pi^2} - \sum_{n=1}^{N-1} (c_n f_n)^2 \}^{1/2} , \tag{2.43}$$

$$c_N = \{ 2\sigma_0^2 - \sum_{n=1}^{N-1} c_n^2 \} , \tag{2.44}$$

By fulfilling the boundary conditions of the $\ddot{\hat{r}}_{\mu\mu}(0) = \ddot{r}_{\mu\mu(0)}$ and $\hat{r}_{\mu\mu(0)} = r_{\mu\mu(0)}$.

2.1.2.3 Generalized Method of Equal Areas (GMEA)

Generalized method is well suited for the design of sum-of-cisoids simulators for Rayleigh fading channel characterized by any specific type of Doppler power spectral density [2]. GMEA requires comparatively large number of cisoids to properly emulate the channel's correlation properties.

Autocorrelation function can be express as;

$$r_{\mu\mu}(\tau) = 4\sigma_0^2 \int_0^{\pi} g_\alpha(\alpha) e^{j2\pi f_{max}\cos(\alpha)\tau} d\alpha \tag{2.45}$$

Where g_α is the even part of the PDF $p_\alpha(\alpha)$ of α.

From (2.43) we can obtain

$$S_{\mu\mu}(f) = 4\sigma_0^2 \frac{g_\alpha(\arccos(\frac{f}{f_{max}}))}{f_{max}\sqrt{1-(\frac{f}{f_{max}})^2}} \quad \text{for } |f| < f_{max} \tag{2.46}$$

We computer the Doppler frequency of $\hat{\mu}(t)$ Such that deterministic angle-of-arrival α_n satisfy the equation

$$\int_{\alpha_{n-1}}^{\alpha_n} g_\alpha(\alpha)d\alpha = \frac{1}{2N}, n = 2,3,\ldots,N \tag{2.47}$$

Here in phase and quadrature components of $\hat{\mu}(t)$ are mutually uncorrelated if and only if

$$\int_0^{\alpha_1} g_\alpha(\alpha)d\alpha = \frac{1}{4N} \ . \tag{2.48}$$

By using (2.45) and (2.46) we can compute angle of arrival $\alpha_n$by employing numerical root-finding technique to solve

$$\int_0^{\alpha_n} g_\alpha(\alpha)d\alpha = \frac{1}{2N}\left(n - \frac{1}{2}\right) \qquad for\, n = 1,2,\ldots,N \ . \tag{2.49}$$

Here corresponding CDF is given by $F_\alpha(\alpha) \coloneqq \int_{-\infty}^{\infty} g_\alpha(\alpha)dx$. Angle of arrival can be computed by evaluating

$$\alpha_n = F_\alpha^{-1}\left(\frac{1}{N}\left[n + N - \frac{1}{2}\right]\right) \text{ for } n = 1,2,\ldots,N. \tag{2.50}$$

Once the α_n are known, the Doppler frequencies f_n can be easily can be obtaind by using f_n = $f_{max}\cos(\alpha_n)$ considering equation (2.46) we can write

$$\int_{-f_{max}}^{f_n} S_{\mu\mu}(f)df = \frac{2\sigma_0^2}{N}\left(n + N - \frac{1}{2}\right) \quad \text{for } n = 1,2,\ldots,N \tag{2.51}$$

Where $f_n \in (-f_{max}, f_{max})$.
For the special case where the invers $P_s^{-1}(.) of\ P_s(.)$ exist, the Doppler frequencies of f_n can be computed by evaluating

$$f_n = P_s^{-1}(\frac{1}{N}\left[n + N - \frac{1}{2}\right]) \qquad \text{for } n = 1, 2, \ldots, N \tag{2.52}$$

If $P_s(.)$ does not exist f_n have to be computed by solving (2.49).

2.1.2.4 Implementation

The Figure 2.21 presents the flow to implement our toolbox for the computation of channel parameters for sum-of-cisoids channel models. The user has to first pick the appropriate computation method for the Doppler frequencies, and Doppler coefficients that can be, for instance GMEA, LPNM or EMEDS. Secondly, he sets the number of scatters, the variance σ_0^2, and the frequency Doppler shift.

Conditions on the computation methods: EMEDS, LPNM and GMEA. The LPNM method needs two functions RIM_VM_PDF_parmeters.m and Rice_pdf_SOCv6.m for the parameter computation.

The block diagram depicts flow of interrelated program.

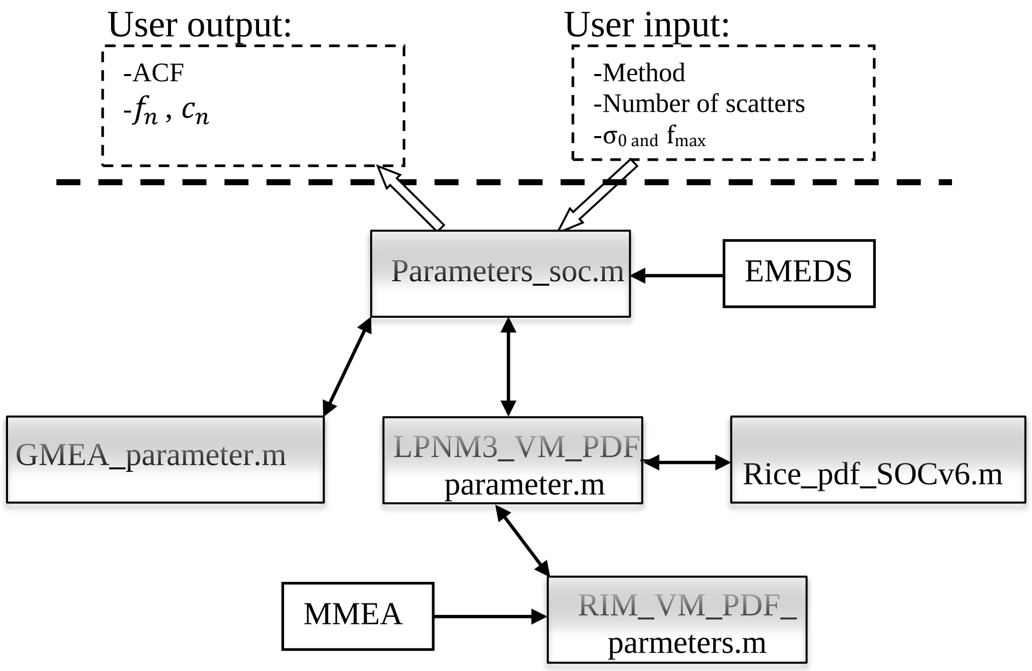


Figure 2.21: The implementation diagram of the parameters computation methods for SOC channel models

2.1.2.5 Results

Figure 2.22 and 2.23 compare ACF against the time difference for the simulation and the reference model under both methods of EMEDS and GMEA respectively.

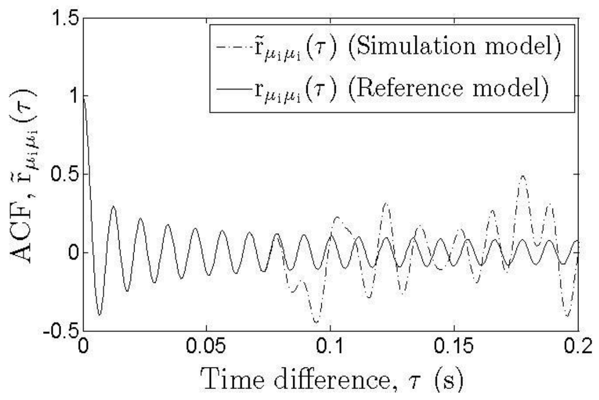


Figure 2.22: ACF of Jakes model using EMEDS method (N = 25, $\sigma_0^2 = 1$, f_{max} = 91 Hz)

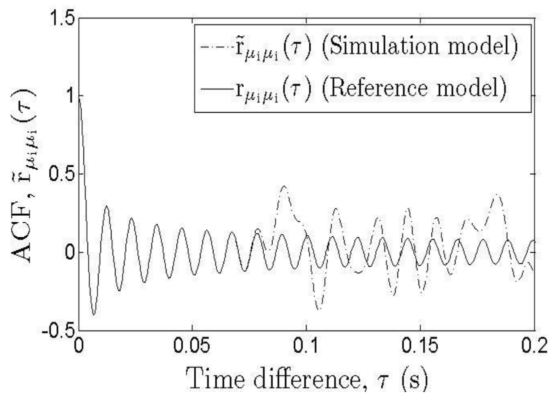


Figure 2.23: ACF of Jakes model using GMEA method (N = 25, $\sigma_0^2 = 1$, f_{max} = 91 Hz)

Figure 2.24 shows comparison between the envelop distribution $p_{\varsigma}(z)$ of the reference model and envelop distribution $\hat{p}_{\varsigma}(z)$ of $\hat{\mu}(t)$by applying the GMEA with N = 10. It is shown that the reference model, simulation theory model and simulation results are found to be over lapping each other closely.

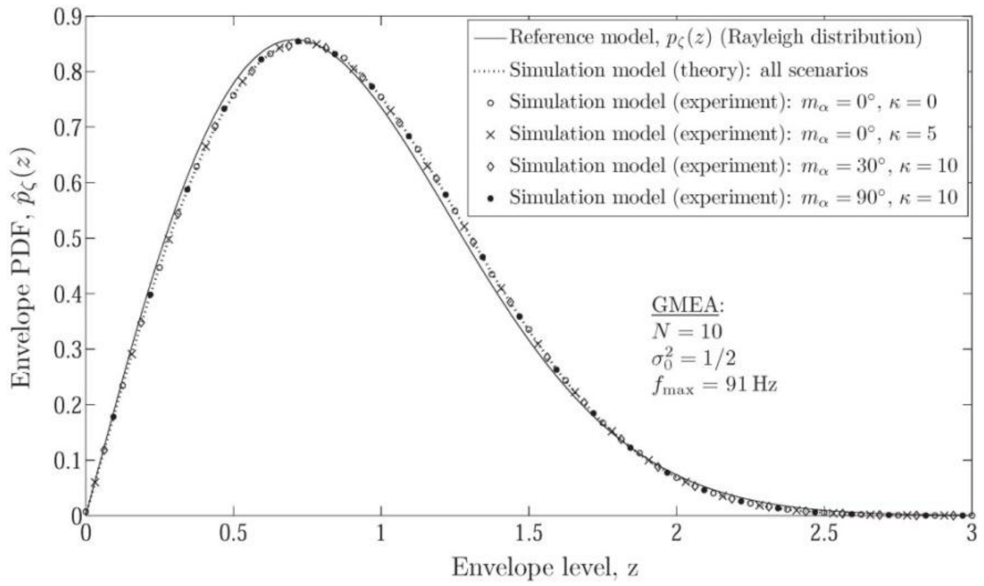


Figure 2.24: Comparison between the envelop distribution p_{ς}(z) of reference model and the envelop distribution $p_{\hat{\varsigma}}$ (z) of simulation model by applying the GMEA to the von Mises distribution of the angle of arrival α

2.2 FREQUENCY SELECTIVE CHANNELS

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

2.2.1 Basic Concepts

The principle of deterministic sum of sinusoid channel model was used to derivation and analysis of the channel model. Our approach is based on the Tapped-delay line model for a frequency-selective mobile fading channel which is drawn in the following figure.

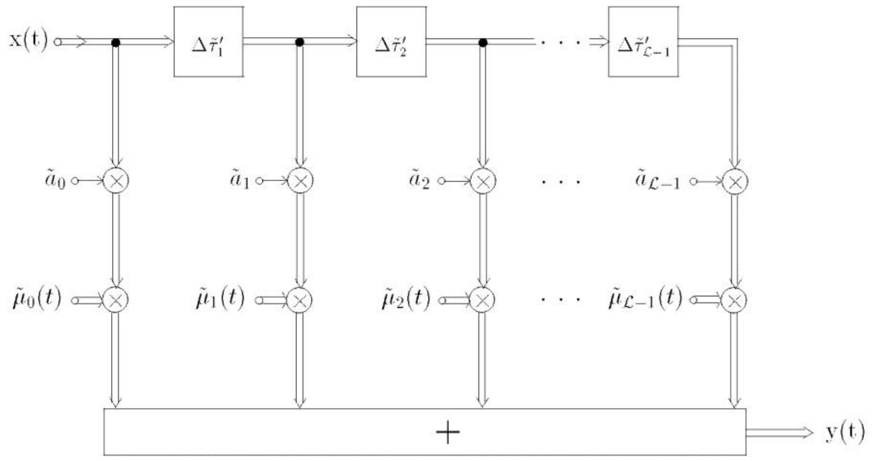


Figure 2.25: Deterministic simulation model for frequency-selective mobile radio channel in the equivalent complex baseband.

According to Figure 2.25, the Deterministic simulation model for frequency-selective mobile radio channel is featured by the time-variant impulse response which resumes the above structure and system function given in (2.53).

$$\tilde{h}\left(\tau',t\right)=\sum_{\ell=0}^{L-1}\tilde{a}_\ell\tilde{\mu}_\ell\left(t\right)\delta\left(\tau'-\tilde{\tau}'_\ell\right) \tag{2.53}$$

Output signal y(t) of the channel simulator can be express as superposition of L delayed replicas of the input signal $x(t-\tilde{\tau}'_l)$.

$$y(t) = \sum_{l=0}^{L-1}\hat{a}_l\,\hat{\mu}_l(t)x(t-\tilde{\tau}'_l) \tag{2.54}$$

We assume that the propagation delays and gains are statistically uncorrelated. These two important parameters can be obtained from the multipath power delay profile $\tilde{S}_{\tau'\tau'}\left(\tau'\right)$. Next, 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 will be shown.

2.2.1.1 Reference Model

The multipath delay power delay spectral density $\tilde{S}_{\tau'\tau'}(\tau')$ is given by (2.55) after considering boundary conditions.

$$\tilde{S}_{\tau'\tau'}(\tau') = \sum_{\ell=0}^{L-1} \tilde{a}_\ell^2 \delta(\tau' - \tilde{\tau}'_\ell) \tag{2.55}$$

Taking to boundary condition into consideration the frequency correlation function FCF can be expressed by the following equation

$$\tilde{r}_{\tau'\tau'}(v') = \sum_{\ell=0}^{L-1} \tilde{a}_\ell^2 e^{-j2\pi v' \tilde{\tau}'_\ell} \tag{2.56}$$

2.2.1.2 Simulation Model

The simulation model, in contrast with the reference model, takes into account the environment type. Therefore 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

$$S_{\tau'\tau'}(\tau') = \begin{cases} c_1 \cdot e^{-b_1 \cdot \tau'}, & 0 \le \tau' < \tau'_{\max} \\ 0, & otherwise \end{cases} \tag{2.57}$$

The frequency correlation function is obtained from

$$r_{\tau'\tau'}(v') = c_1 \cdot \frac{1 - e^{-\tau'_{\max}(b_1 + j2\pi v')}}{b_1 + j2\pi v'} \tag{2.58}$$

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

$$S_{\tau'\tau'}(\tau') = \begin{cases} c_1 \cdot e^{-b_1 \cdot \tau'}, & 0 \le \tau' < \tau'_1 \\ c_2 \cdot e^{b_2 - b_3 \cdot \tau'}, & \tau'_2 \le \tau' < \tau'_{\max} \\ 0, & otherwise \end{cases} \tag{2.59}$$

$$r_{\tau'\tau'}(v') = c_1 \cdot \frac{1 - e^{-\tau'_{\max}(b_1 + j2\pi v')}}{b_1 + j2\pi v'} + c_2 \cdot e^{b_2} \cdot \frac{e^{-\tau'_2(b_3 + j2\pi v')} - e^{\tau'_{\max}(b_3 + j2\pi v')}}{b_3 + j2\pi v'} \tag{2.60}$$

2.2.2 Parameters Computation Methods

Similar to section 2.1.1, different computation methods that are mostly used to compute the discrete delays and discrete power delay profile are studied.

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 emphasis is given to 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

$$\tilde{\tau}'_\ell = \ell \cdot \Delta\tilde{\tau}'_\ell, \qquad \ell = 0,1,\ldots,L-1 \tag{2.61}$$

Where

$$\Delta\tilde{\tau}'_\ell = \frac{\tau'_{\max}}{L-1} \tag{2.62}$$

For HT environment, they are expressed by

$$\tilde{\tau}'_\ell = \begin{cases} \ell \cdot \Delta\tilde{\tau}'_{\ell HT}, & \ell = 0,1,\ldots,L_1 \\ \left(\ell + L_2 - L_1 - 1\right) \cdot \Delta\tilde{\tau}'_{\ell HT}, & \ell = L_1+1, L_1+2,\ldots,L-1 \end{cases} \tag{2.63}$$

Where

$$\Delta\tilde{\tau}'_{\ell HT} = \frac{\tau'_{\max} - \tau'_2 + \tau'_1}{L-1} \tag{2.64}$$

$$L_1 = \left\lfloor \frac{\tau'_1}{\Delta\tilde{\tau}'_{\ell HT}} \right\rfloor \tag{2.65}$$

$$L_2 = round\left\{ \frac{\tau'_2}{\Delta\tilde{\tau}'_{\ell HT}} \right\} \tag{2.66}$$

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

$$\tilde{a}_\ell = \begin{cases} \sqrt{\frac{c_1}{b_1}\left[1 - e^{-b_1 \Delta\tilde{\tau}'_\ell/2}\right]}, & \ell = 0 \\ \sqrt{\frac{c_1}{b_1}\left[e^{-b_1\left(\tilde{\tau}'_\ell - \Delta\tilde{\tau}'_\ell/2\right)} - e^{-b_1\left(\tilde{\tau}'_\ell + \Delta\tilde{\tau}'_\ell/2\right)}\right]}, & \ell = 1,2,\ldots,L-2 \\ \sqrt{\frac{c_1}{b_1}\left[e^{-b_1\left(\tilde{\tau}'_{\max} - \Delta\tilde{\tau}'_\ell/2\right)} - e^{-b_1 \tilde{\tau}'_{\max}}\right]}, & \ell = L-1 \end{cases} \tag{2.67}$$

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

$$\tilde{a}_\ell = \begin{cases} \sqrt{\dfrac{c_1}{b_1}\left[1-e^{-b_1\Delta\tilde{\tau}'_\ell/2}\right]}, & \ell = 0 \\ \sqrt{\dfrac{c_1}{b_1}\left[e^{-b_1\left(\tilde{\tau}'_\ell-\Delta\tilde{\tau}'_\ell/2\right)}-e^{-b_1\left(\tilde{\tau}'_\ell+\Delta\tilde{\tau}'_\ell/2\right)}\right]}, & \ell = 1,2,\ldots,L_3-1 \\ \left\{\dfrac{c_1}{b_1}\left[e^{-b_1\left(\tilde{\tau}'_\ell-\Delta\tilde{\tau}'_\ell/2\right)}-e^{-b_1\tau'_1}\right]+\right. & \\ \left.\dfrac{c_2}{b_3}e^{b_2}\left[e^{-b_3\tau'_1}-e^{-b_3\left(\tilde{\tau}'_\ell+\Delta\tilde{\tau}'_\ell/2\right)}\right]\right\}^{1/2}, & \ell = L_3 \\ \sqrt{\dfrac{c_2}{b_3}e^{b_2}\left[e^{-b_3\left(\tilde{\tau}'_\ell-\Delta\tilde{\tau}'_\ell/2\right)}-e^{-b_3\left(\tilde{\tau}'_\ell+\Delta\tilde{\tau}'_\ell/2\right)}\right]}, & \ell = L_3+1,\ldots,L-2 \\ \sqrt{\dfrac{c_2}{b_3}e^{b_2}\left[e^{-b_3\left(\tau'_{\max}-\Delta\tilde{\tau}'_\ell/2\right)}-e^{-b_3\tau'_{\max}}\right]}, & \ell = L-1 \end{cases} \tag{2.68}$$

where

$$L_3 = round\left\{\frac{\tau'_1}{\Delta\tilde{\tau}'_\ell}\right\} \tag{2.69}$$

For HT environment, we get the following expression

$$\tilde{a}_\ell = \begin{cases} \sqrt{\dfrac{c_1}{b_1}\left[1-e^{-b_1\Delta\tilde{\tau}'_{\ell HT}/2}\right]}, & \ell = 0 \\ \sqrt{\dfrac{c_1}{b_1}\left[e^{-b_1\left(\tilde{\tau}'_\ell-\Delta\tilde{\tau}'_{\ell HT}/2\right)}-e^{-b_1\left(\tilde{\tau}'_\ell+\Delta\tilde{\tau}'_{\ell HT}/2\right)}\right]}, & \ell = 1,2,\ldots,L_1-1 \\ \sqrt{\dfrac{c_1}{b_1}\left[e^{-b_1\left(\tilde{\tau}'_\ell-\Delta\tilde{\tau}'_{\ell HT}/2\right)}-e^{-b_1\tau'_1}\right]}, & \ell = L_1 \\ \sqrt{\dfrac{c_2}{b_3}e^{b_2}\left[e^{-b_3\tau'_2}-e^{-b_3\left(\tilde{\tau}'_\ell+\Delta\tilde{\tau}'_{\ell HT}/2\right)}\right]}, & \ell = L_1+1 \\ \sqrt{\dfrac{c_2}{b_3}e^{b_2}\left[e^{-b_3\left(\tilde{\tau}'_\ell-\Delta\tilde{\tau}'_{\ell HT}/2\right)}-e^{-b_3\left(\tilde{\tau}'_\ell+\Delta\tilde{\tau}'_{\ell HT}/2\right)}\right]}, & \ell = L_1+2,\ldots,L-2 \\ \sqrt{\dfrac{c_2}{b_3}e^{b_2}\left[e^{-b_3\left(\tilde{\tau}'_\ell-\Delta\tilde{\tau}'_{\ell HT}/2\right)}-e^{-b_3\tau'_{\max}}\right]}, & \ell = L-1 \end{cases} \tag{2.70}$$

2.2.2.2 Mean-Square-Error Method (MSEM)

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

$$E_{r_{\tau'\tau'}} = \frac{1}{v'_{\max}} \int_0^{v_{\max}} \left| r_{\tau'\tau'}(v) - \tilde{r}_{\tau'\tau'}(v) \right|^2 dv \tag{2.71}$$

The discrete delays:

The discrete delays for the RA, TU and BU environments are given by the equation (2.61), however, for the HT environment they are given by (2.63).

The discrete gains:

After substituting the equation (2.55) in (2.74), we get the following expression for the discrete gains whatever the environment type

$$\tilde{a}_\ell = \sqrt{\operatorname{Re}\left\{ \frac{1}{v'_{\max}} \int_0^{v'_{\max}} r_{\tau'\tau'}(v') e^{j2\pi v' \tilde{\tau}'_\ell} dv' \right\}}, \qquad \ell = 0,1,\ldots,L-1 \tag{2.72}$$

where

$$v'_{\max} = \frac{L-1}{2\tau'_{\max}} \tag{2.73}$$

2.2.2.3 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

$$\tilde{\tau}'_\ell = -\frac{1}{b_1} \cdot \ln\left(-\frac{b_1}{c_1} \sigma_d^2 \frac{\ell}{L} + 1 \right), \quad \ell = 0,1,2,\ldots,L-1 \tag{2.74}$$

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

$$\tilde{\tau}'_\ell = \begin{cases} -\frac{1}{b_1} \cdot \ln\left(-\frac{b_1}{c_1} \sigma_d^2 \frac{\ell}{L} + 1 \right), & \ell \le L_4 \\ -\frac{1}{b_3} \cdot \ln\left(\frac{b_3}{c_2 \cdot e^{b_2}} \left[A - \sigma_d^2 \frac{\ell}{L} \right] + e^{-b_3 \cdot \tau'_2} \right), & \ell > L_4 \end{cases} \tag{2.75}$$

where $A = \frac{c_1}{b_1}\left(1 - e^{-b_1 \cdot \tau'_1}\right)$ (2.76)

$$L_4 = \left\lfloor A \cdot \frac{L}{\sigma_d^2} \right\rfloor \tag{2.77}$$

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 (σ_d=1)

$$\tilde{a}_\ell = \frac{\sigma_d}{\sqrt{L}}, \qquad \ell = 0,1,2,\ldots,L-1 \tag{2.78}$$

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

$$\tau'_\ell = -\frac{1}{b_1} \cdot \ln\left(-\frac{b_1}{c_1} \sigma_d^2 u_\ell + 1 \right), \qquad \ell = 0,1,\ldots,L-1 \tag{2.79}$$

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

$$\tau'_\ell = \begin{cases} -\frac{1}{b_1} \cdot \ln\left(-\frac{b_1}{c_1} \sigma_d^2 u_\ell + 1 \right), & \ell \leq L_4 \\ -\frac{1}{b_3} \cdot \ln\left(\frac{b_3}{c_2 \cdot e^{b_2}} \left[A - \sigma_d^2 \right] u_\ell + e^{-b_3 \cdot \tau'_2} \right), & \ell > L_4 \end{cases} \tag{2.80}$$

The discrete gains:

The delays coefficients are given by the equation (2.73) for all environment types.

2.2.2.5 Lp-norm Method (LPNM)

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

$$E_{r_{\tau'\tau'}}^{(p)} = \left\{ \frac{1}{v'_{\max}} \int_0^{v'_{\max}} \left| r_{\tau'\tau'}(v') - \tilde{r}_{\tau'\tau'}(v') \right|^p dv' \right\}^{1/p} \tag{2.81}$$

Where $$v'_{\max} = \frac{L-1}{2\tau_{\max}} \tag{2.82}$$

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

The Figure 2.26 shows the structure that proposed 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. The present figure exhibits the structure of the toolbox.

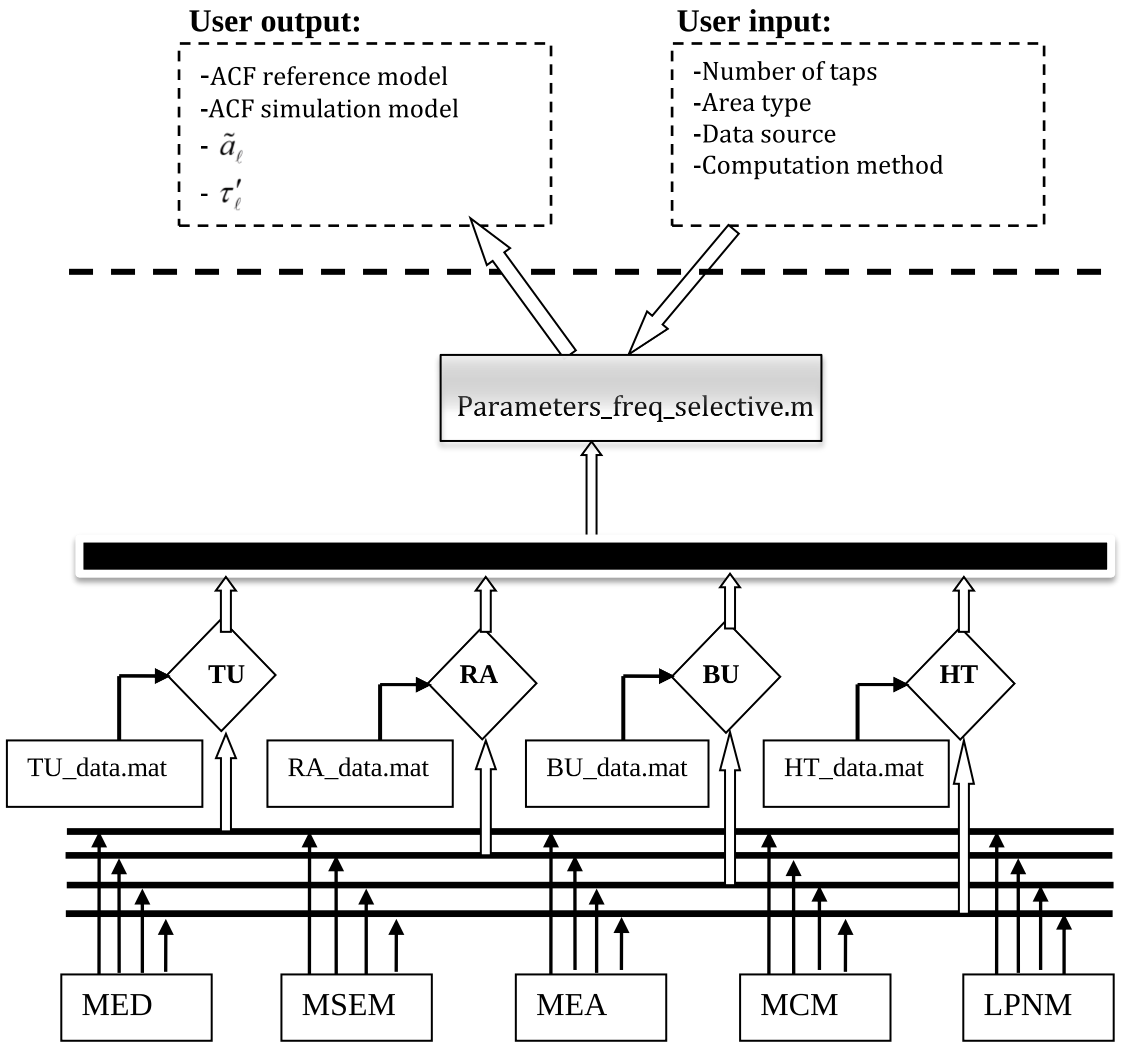


Figure 2.26: The implementation diagram of the parameters computation methods for the frequency-selective channel

According to the number of methods, the program is divided into five parts. 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.

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

Table 2.1: Parameter value under four environmental condition 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;

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

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

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

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

2.2.4 Results

In this section, comparison has been carried out to examine the performance of the parameter computation methods of MED, MESM, MEA, MCM and LPNM based on different environments conditions. The following figures depict the frequency correlation function (FCF) of the reference model towards the frequency correlation function of the simulation. The closeness of both simulation and reference model shows how far the method fulfils our requirements. As parameters values, we take for number of path 20 and as standard, the COST 207. Value of the *p* in LPNM method was set as $p = 2$.

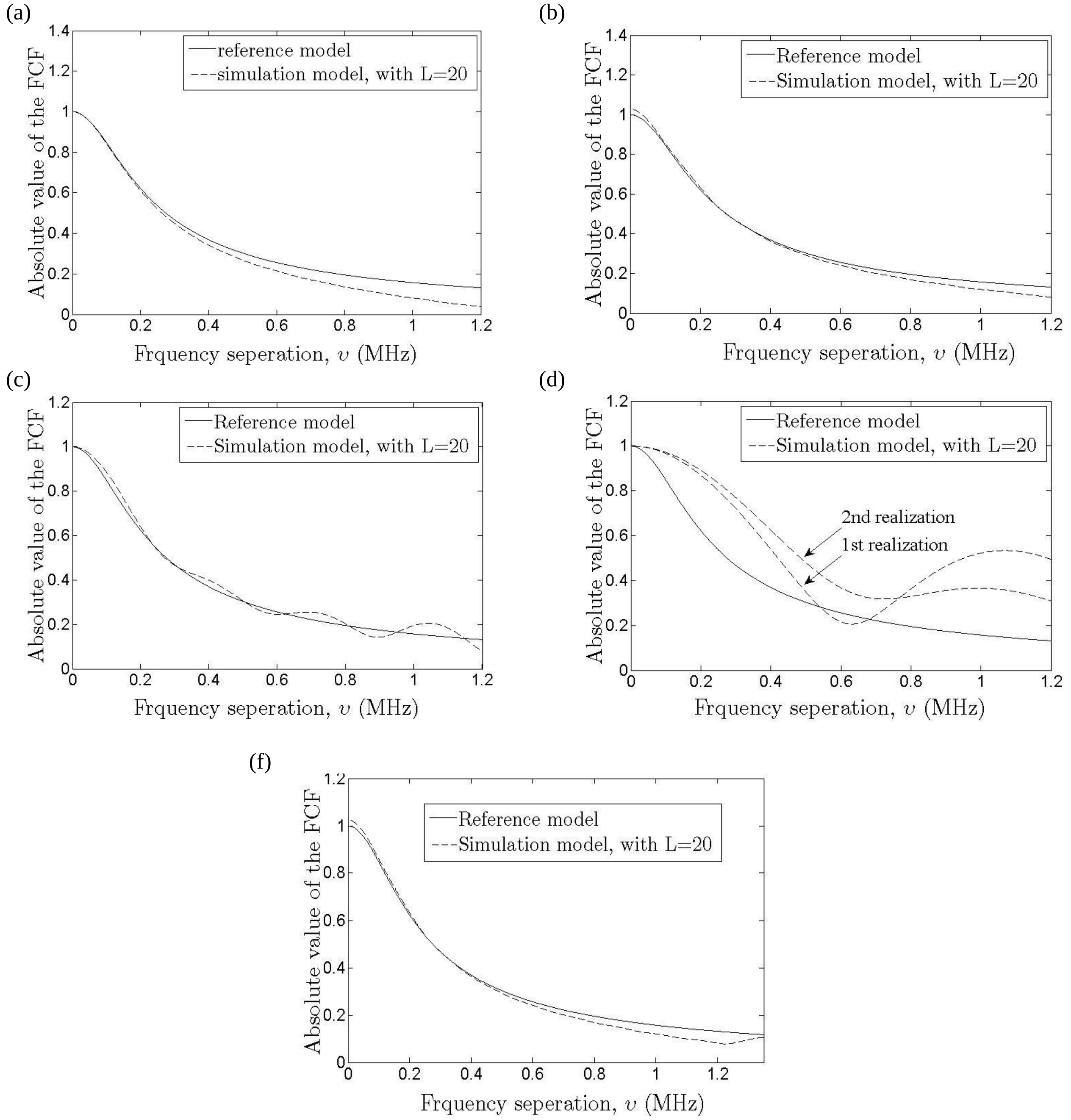
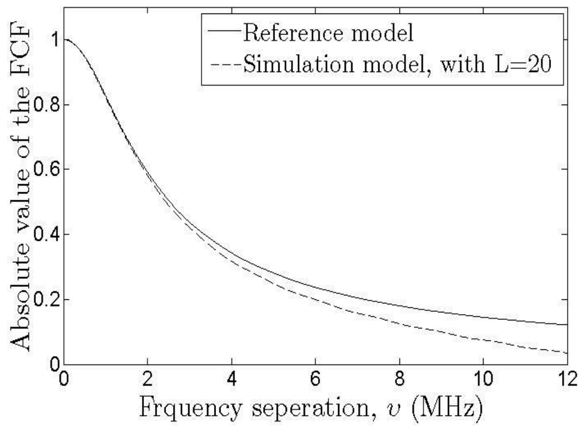
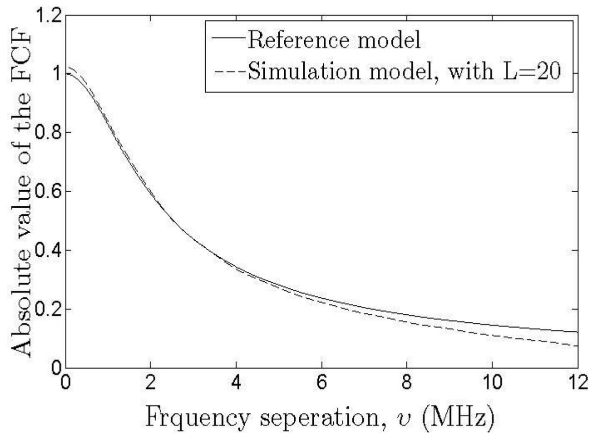


Figure 2.27: Absolute value of the FCF behaviour shown with referred to reference model for the 20 path, TU and under parameter computation methods of (a) MED, (b) MSEM, (c) MEA, (d) MCM and (f) LPNM.

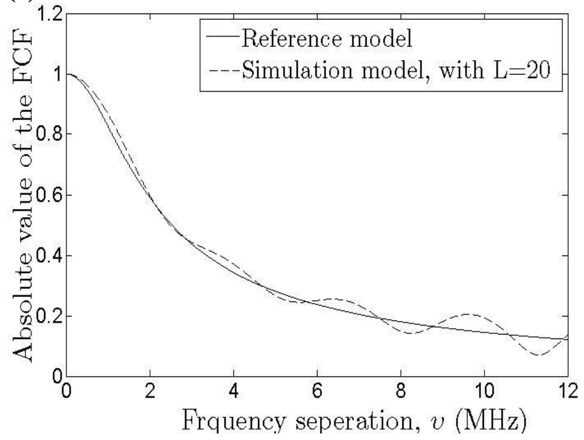
(a)



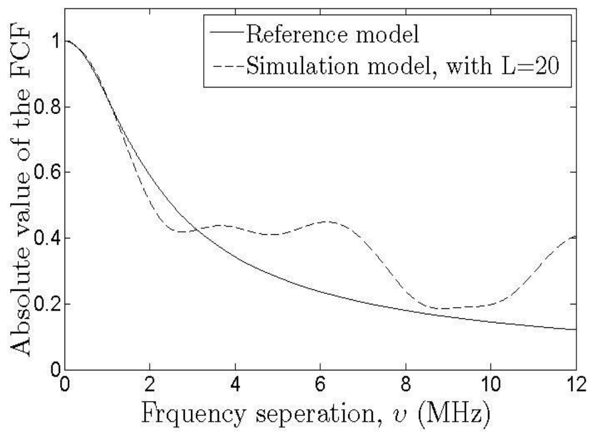
(b)



(c)



(d)



(f)

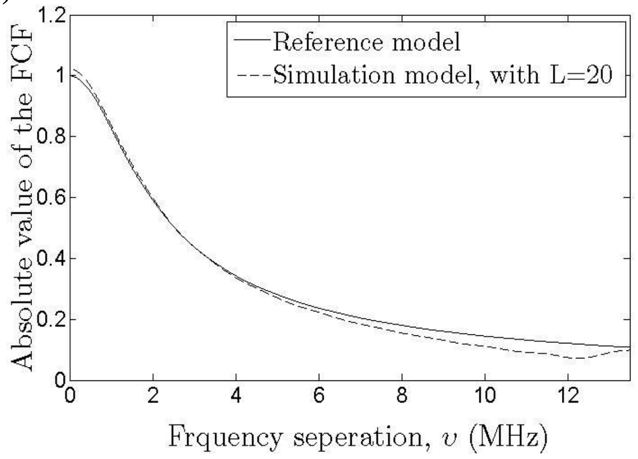


Figure 2.28: Absolute value of the frequency correlation function (FCF) behaviour shown (with referred to reference model for the 20 path, RA) under parameter computation methods of (a) MED, (b) MSEM, (c) MEA, (d) MCM and (f) LPNM.

Figure 2.28 shows that the MSEM and LPNM methods are given good agreement with the reference model of absolute value of the frequency correlation function behaviour under RA condition compared to other methods.

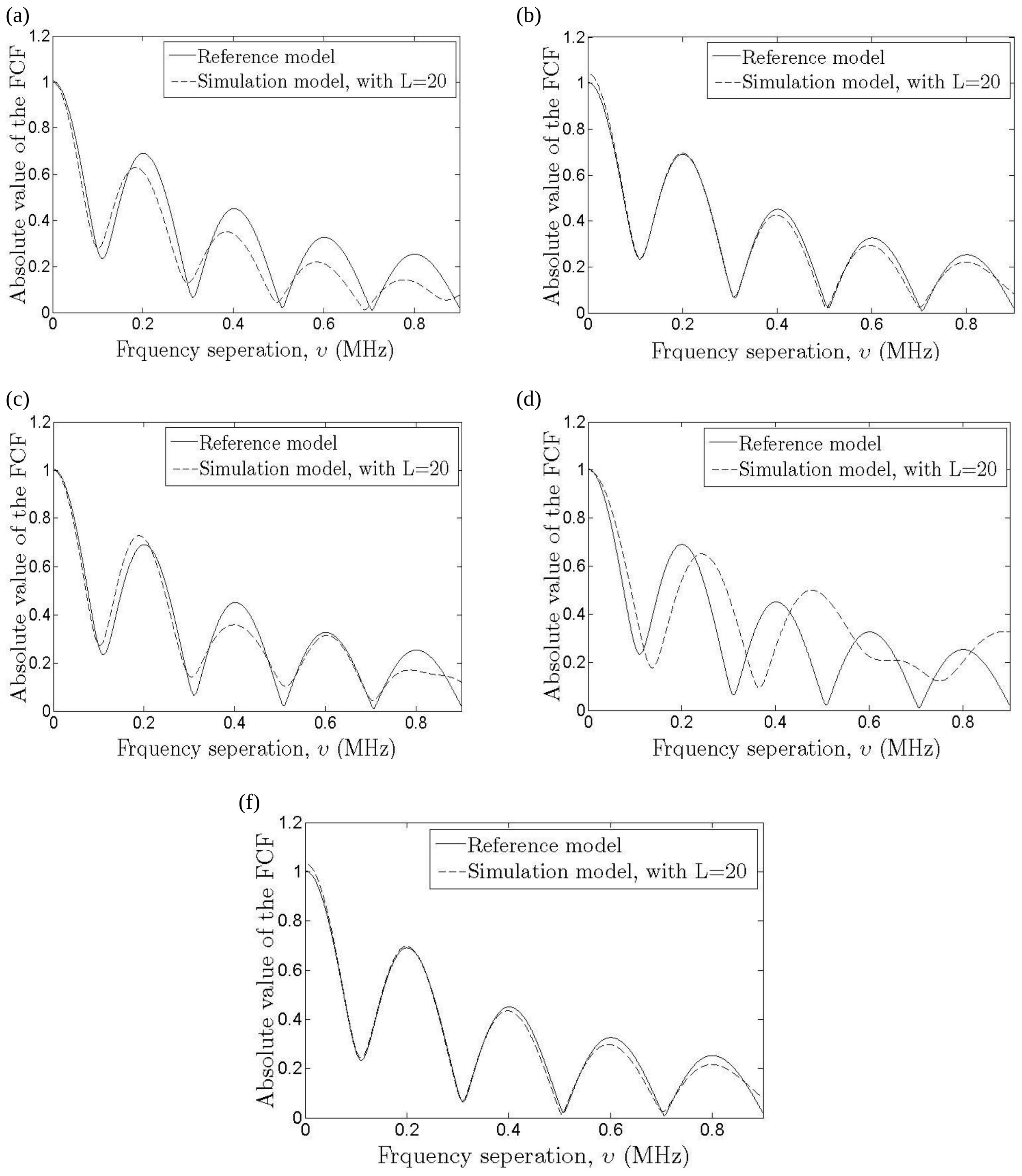


Figure 2.29: Absolute value of the frequency correlation function (FCF) behaviour shown (with referred to reference model for the 20 path, BU) under parameter computation methods of (a) MED, (b) MSEM, (c) MEA, (d) MCM and (f) LPNM.

Figure 2.29 shows that the MSEM and LPNM methods are given good fitting with reference model of absolute value of the frequency correlation function behaviour under BU condition compared to other methods.

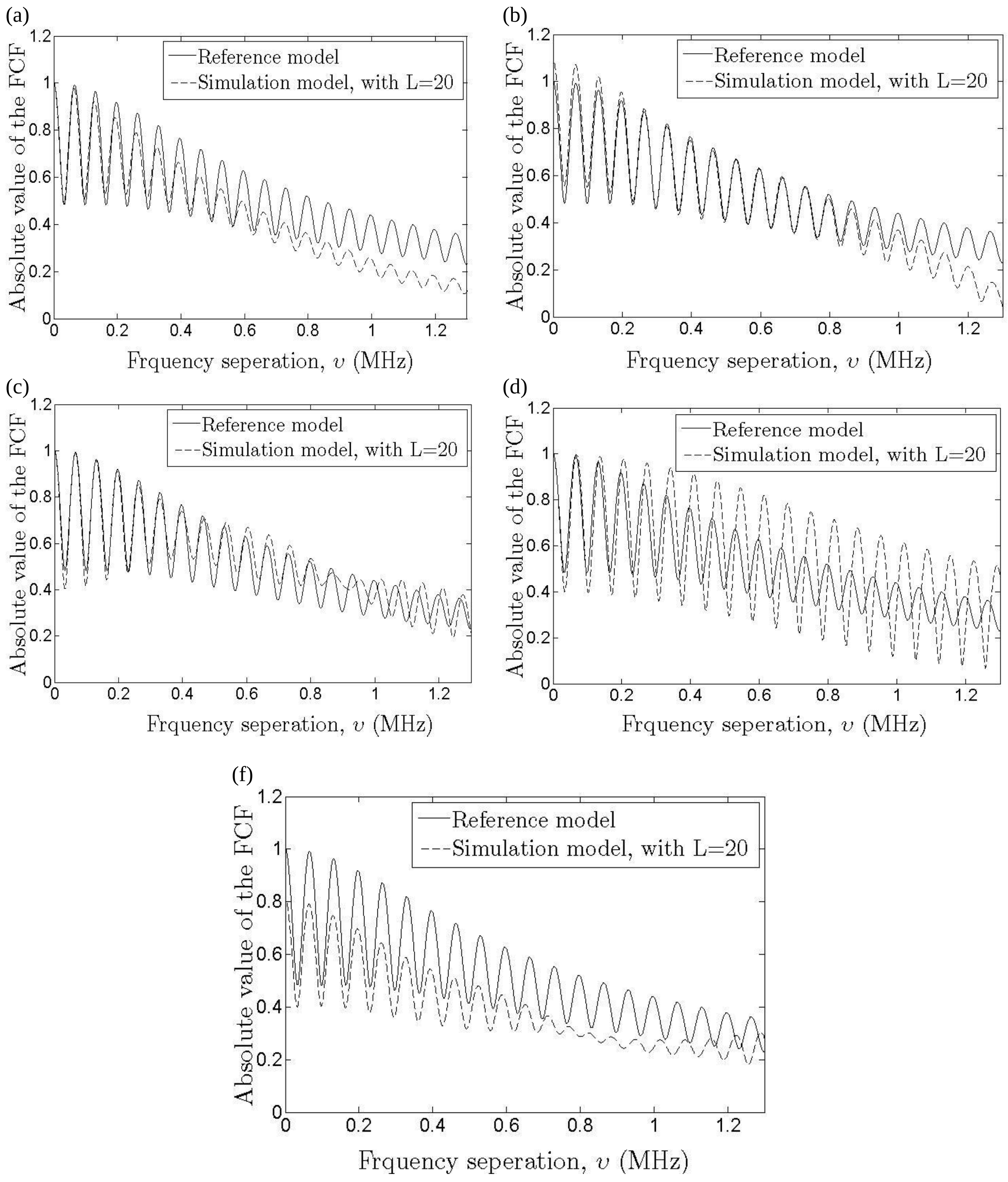


Figure 2.30: Absolute value of the frequency correlation function (FCF) behaviour shown (with referred to reference model for the 20 path, HT) under parameter computation methods of (a) MED, (b) MSEM, (c) MEA, (d) MCM and (f) LPNM.

Figure 2.30 shows that the MSEM methods is given good fitting with reference model of absolute value of the frequency correlation function behaviour under HT condition compared to other methods.

2.3 CONCLUSION OF THE PARAMETER COMPUTATION

In this chapter, emphasis is placed to discuss and implement variety of different methods for the computation of the primary parameters of the simulation models. As a conclusion we considered that the mobile channel fading is recognized in two families of channels: the frequency-non-selective channels, and the frequency-selective channel. In the first part, we described the different well-known methods (MED, MEA, MSEM, MCM, LPNM, MEDS, JM, RMEDS and MEDS-SP), which are used to compute the channels' parameters for the sum-of-sinusoid channel models. In first part we did parameter computation for the sum-of-cisoids channel model also based on the EMEDS, LPNM, and GMEA methods. Nonetheless, to ensure the efficiency of each method separately, the auto-correlation function of both the simulation model and the reference model is plotted to evaluate their closeness.

The second part deals with the parameters computation for the frequency-selective channels that considered 5 methods namely MED, MEA, MSEM, MCM and LPNM. In addition, studies have been carried out to seek the performance using the frequency correlation function of both the simulation model and the reference model. Here FCF behaviour was observed for RA, BU, TU and HT environments conditions. The result reveals that by taking few realizations best fitting parameter for RA, BU, TU and HT under MCM. MED, MEA, MSEM, MCM and LPNM computation methods are given good fitting behaviour for environment conditions of TU and RA. Under BU environment condition MSEM and LPNM computation methods are given better response compared to other computation methods. Under HT environment condition MSEM computation method is given better response compared to other computation methods.

However some computation methods are found to be good agreement with the reference model under different scenario. It has been understood that further studies are needed to enhance the computation method. When this parameter computation method is used for particular channel modelling, it has to be select appropriate computation method. Properly selected computation method will be gave best fitted channel simulator model which fit with reference model or measurement based model.

Next chapter dedicated for implementation of mobile channel simulators and developed parameter computations methods will use in this section.

CHAPTER 3

MOBILE FADING CHANNEL SIMULATORS

In this chapter, different fading channel simulators which were introduced are discussed. The first part concerns implementation of the real world land mobile satellite (LMS) channel based on the extended Suzuki process Type I. Thereafter developed spatial channel simulator is presented. It was implemented based on some adequate models like Gudmundson, Butterworth and Gauss models. The third part of this chapter deals with MIMO channel simulators. In this section, different procedures for finding the different models (one ring, tow ring and elliptical model) are described. It should be noted that the models presented here are the summary of the literature reviewed in this research. After implementation of the MIMO channel models; modelling of the multiple uncorrelated Rayleigh channel simulator was described with its properties. Final section of the channel modelling was implementation of the frequency hopping channel simulator model. It is made clear that for each simulator, we implemented some functions for testing, such as time ACF, CCF, etc. and examined the correctness of the simulator with its reference model or with the measurement model to make sure fittingness.

3.1 NON-STATIONARY LAND MOBILE SATELLITE CHANNEL

In order to provide global coverage with a high signal quality to diverse users, seamless integration of terrestrial and satellites networks are expected to play a vital role in the upcoming era of mobile communications. This section focuses on modelling of Real world land mobile satellite (LMS) channel. Real world LMS channel are becoming always non-stationary if the observation time interval increases. A non-stationary model was introduced by Lutz et al. [10]. The concept of Lutz’s can easily be generalized, leading to an M-state Markov process, where each state is represented by a specific stationary stochastic process. It has been shown that how the statistical properties of the proposed dynamic channel model can be adapted to the statistics of real-world LMS channels in different environments. From the dynamic m-state model, which is considered as reference model, an efficient simulation model is derived by replacing all colored Gaussian noise processes by finite sums of sinusoidal functions. It is demonstrated by several theoretical and simulation results. The performance of the reference model and the simulation model are well tally with respect to the probability density function (PDF) of the fading envelope, level-crossing rate (LCR), and average duration of fades (ADF).

3.1.1 Reference Model

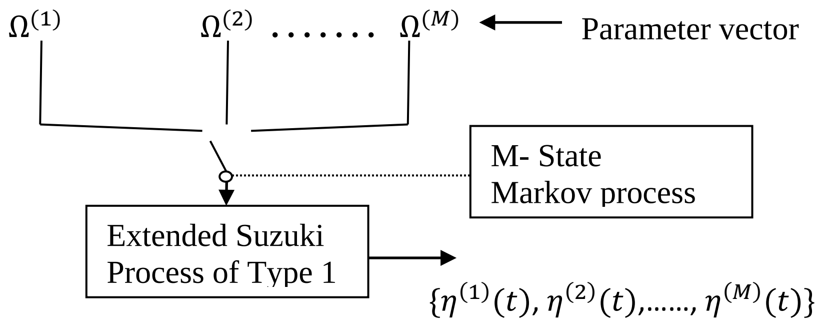

$\Omega^{(1)}$
$\Omega^{(2)}$ ....
$\Omega^{(M)}$
Parameter vector
M- State Markov process
Extended Suzuki Process of Type 1
$\{\eta^{(1)}(t), \eta^{(2)}(t),......, \eta^{(M)}(t)\}$

Figure 3.1: Dynamic model with M states for modelling of non-stationary LMS channel

Instead of using mixture of *m* channel models, same channel with different parameter setting related to each state is used. Proposed dynamic model with M state for modelling of non-stationary LMS channel model shown in Figure 3.1. For each of the state M channel states, a specific set of model parameters has to be determined to assign for the channel model. Change of one the one channel state to another leads configuration of the embedded stationary channel model. The switching of the parameter vector is controlled by a discrete M-state Markov process. In this extended Suzuki process of Type I was embedded with the M-state Markov process such that a transition from one state to another corresponds to an exchange of all those model parameters which characterize the statistics of the Suzuki process [11].

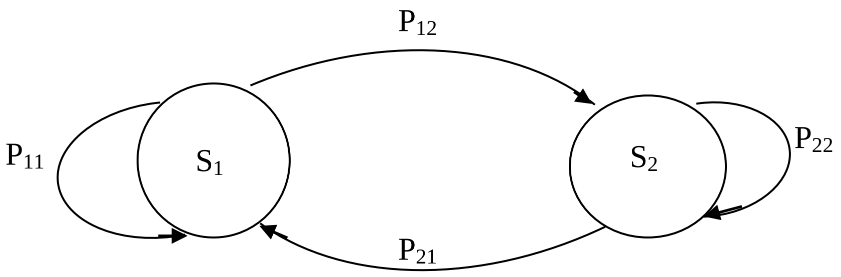


Figure 3.2: State transition of the two-state Markov model

3.1.2 Simulation Model

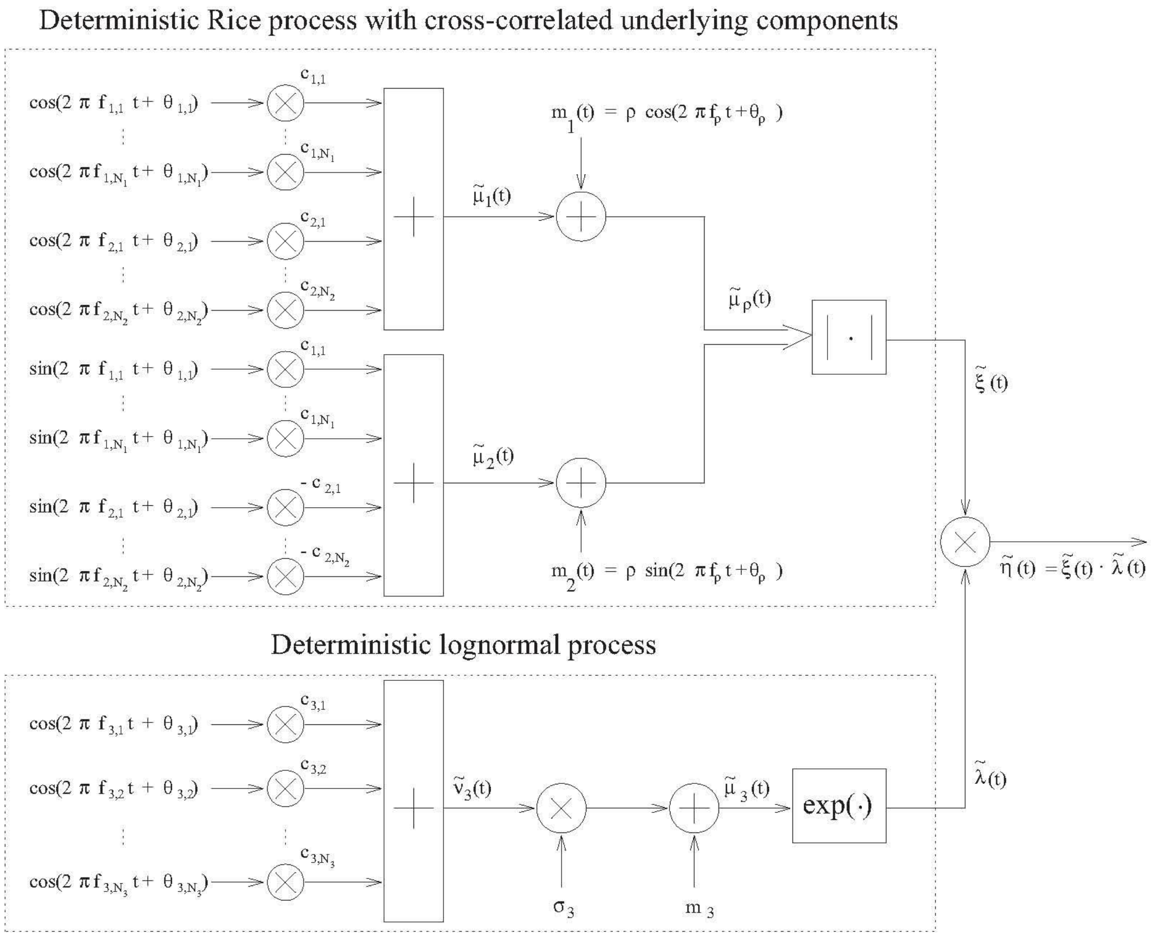

Deterministic Rice process with cross-correlated underlying components
$\cos(2\pi f_{1,1} t + \theta_{1,1})$
$\cos(2\pi f_{1,N_1} t + \theta_{1,N_1})$
$\cos(2\pi f_{2,1} t + \theta_{2,1})$
$\cos(2\pi f_{2,N_2} t + \theta_{2,N_2})$
$\sin(2\pi f_{1,1} t + \theta_{1,1})$
$\sin(2\pi f_{1,N_1} t + \theta_{1,N_1})$
$\sin(2\pi f_{2,1} t + \theta_{2,1})$
$\sin(2\pi f_{2,N_2} t + \theta_{2,N_2})$
$c_{1,1}$
c_{1,N_1}
$c_{2,1}$
c_{2,N_2}
$-c_{2,1}$
$-c_{2,N_2}$
$m_1(t) = \rho\cos(2\pi f_\rho t + \theta_\rho)$
$m_2(t) = \rho\sin(2\pi f_\rho t + \theta_\rho)$
$\tilde{\mu}_1(t)$
$\tilde{\mu}_2(t)$
$\tilde{\mu}_\rho(t)$
$|\cdot|$
$\tilde{\xi}(t)$
$\tilde{\eta}(t) = \tilde{\xi}(t)\cdot\tilde{\lambda}(t)$
Deterministic lognormal process
$\cos(2\pi f_{3,1} t + \theta_{3,1})$
$\cos(2\pi f_{3,2} t + \theta_{3,2})$
$\cos(2\pi f_{3,N_3} t + \theta_{3,N_3})$
$c_{3,1}$
$c_{3,2}$
c_{3,N_3}
$\tilde{\nu}_3(t)$
σ_3
m_3
$\tilde{\mu}_3(t)$
$\exp(\cdot)$
$\tilde{\lambda}(t)$

Figure 3.3: deterministic simulation model for extended Suzuki process (Type I)

Simulation model was based on the extended Suzuki process Type I [2] as shown in Figure 3.3. Simulation model for the extended Suzuki process $\eta(t)$ of Type I replaced by real-valued Gaussian noise processes $\nu_i\ (t)$ as shown in following sum of N_i sinusoid.

$$\tilde{\nu}_i\ (t) = \sum_{n=1}^{N} c_{i,n} \cos(2\pi f_{i,n} t + \theta_{i,n})\ , i = 1, 2, 3. \tag{3.1}$$

$$c_{i,n} = \begin{cases} \sigma_0\sqrt{\frac{1}{N_1}} & , i = 1 \\ \sigma_0\sqrt{\frac{1}{N_2{}'}} & , i = 2 \end{cases} \tag{3.2}$$

$$f_{i,n} = \begin{cases} f_{max}\ sin\left[\frac{\pi}{2N_1}(n - \frac{1}{2})\right] & , i = 1\ , \\ f_{max}\ \ sin\left[\frac{\pi}{2N_2{}'}(n - \frac{1}{2})\right] & , i = 2\ , \end{cases} \tag{3.3}$$

Here $n = 1,2,\ldots,N_i$ and $N_2{}' = \left\lceil \frac{N_2}{\frac{2}{\pi}\arcsin(K_0)} \right\rceil$ (3.4)

Parameter vector Ω is defined by; Ω = ($\sigma_0,\ \kappa_0,\ \sigma_3, m_3, \rho\ \ , f_\rho,\ f_{max}$)

The discrete Doppler frequency of $f_{3,n}$ of the 3rd sum-of-sinusoids $\tilde{\nu}_i$ (t) are obtained numerically by computing the zeros of

$$\frac{2n-1}{2N_3} - erf\left(\frac{f_{3,n}}{\sqrt{2}\sigma_0}\right) = 0 \tag{3.5}$$

For all n = 1, 2,…, N_3 -1 and $c_{3,n} = \sqrt{\frac{2}{N_3}}$.

The phases $\theta_{i,n}$ are outcomes of a random generator which uniformly distributed over (0,2π] for all n = 1, 2,.., N_i (i = 1, 2, 3). The remaining parameters of the simulation model, i.e., σ_3, m_3, ρ, f_ρ, and θ_ρ are identical with reference model.

Under the mapping procedure we fitted the relevant statistical properties (probability density function, level crossing rate, average duration of fades) of the dynamic *M*-state reference model and simulation model very closely to the real-world satellite channel (from the measured signal). By using the reference model the number of state (*M*) is determined and based on that parameter vector $\mathbf{\Omega}^{(m)}$ is determined. The proposed simulation model was implemented for *M* = 2 by doing neglect of the handover process. Figure 3.2 shows the state diagram for the two states. The parameter vectors of $\mathbf{\Omega}^{(1)}$ and $\mathbf{\Omega}^{(2)}$ as well as the time sharing factor was calculated by minimizing the error function. Reference model parameters were calculated based on the measured signal. Parameters which were referred in developing the model are listed in Table 3.1. Time sharing factor A_1, transition probabilities $p_{i,j}$ (i, j = 1, 2) of the 2-state model, Δ_{min} and Δ_{max} determined by the measured signal as shown in Table 3.2. State transition of the two-state Markov model is shown in Figure 3.2.

Table 3.1: Optimized component of the parameter vector $\mathbf{\Omega}^{(m)}$ (m = 1, 2) of the reference model for an equivalent satellite channel in a suburban and urban area [11].

Area	m	$\sigma_0^{(m)}$	$\kappa_0^{(m)}$	$\sigma_3^{(m)}$	$m_3^{(m)}$	$\rho^{(m)}$	$f_0^{(m)}$ (Hz)	$f_{max}^{(m)}$ (Hz)
Sub-Urban	1	1.1133	0.5976	0.37967	-1.3428	0.75684	7.3062	14.175
	2	0.2717	0.7305	0.00833	0.3826	0.73937	1.9173	30.1185
Urban	1	0.1392	0.9997	0.52197	-0.3438	0.0631	6.3	20.1323
	2	0.43121	0.9934	0.09586	-0.2033	1.4767	3.9353	23.6614

Table 3.2: Parameters of the dynamic (2+2)-state model for an equivalent satellite channel in suburban and urban area [11]

Area	Δ_{min} (s)	Δ_{max} (s)	A_1	P_{11}	P_{12}	P_{21}	P_{22}
Suburban	0.016	1.066	0.40	0.99873	0.00127	0.000847	0.99915
Urban	0.02	1.082	0.55	0.99931	0.00069	0.000843	0.99916

3.1.3 Implementation

The 2-states non-stationary LMS simulation channel simulator which is shown in Figure 3.4, was implemented under the Matlab environment as a set of m-files containing the different functions and procedures. The implementation was organized following a logical structure.

- Computation of the parameters: ueberg2.m and prob.m m-file use to determine reference model parameters such as number of transition of the process from the measured data file, The prob.m, computation of the transition probabilities, $\varDelta_{min}$, $\varDelta_{max}$, Mean $\varDelta$ etc. from the measured signal data file.
- Determination of the processes: The simu_fun.m file generates the simulated signal by using simu.m file which based on the extended Suzuki type I process. The mzust.m program use to switch both the extended Suzuki processes based on transition probabilities and time sharing factor to generate equivalent satellite signal of $\eta^{(1)}(t)$ and $\eta^{(2)}(t)$.
- 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 complimentary CDF, LCR and ADF, each in a single Matlab file.

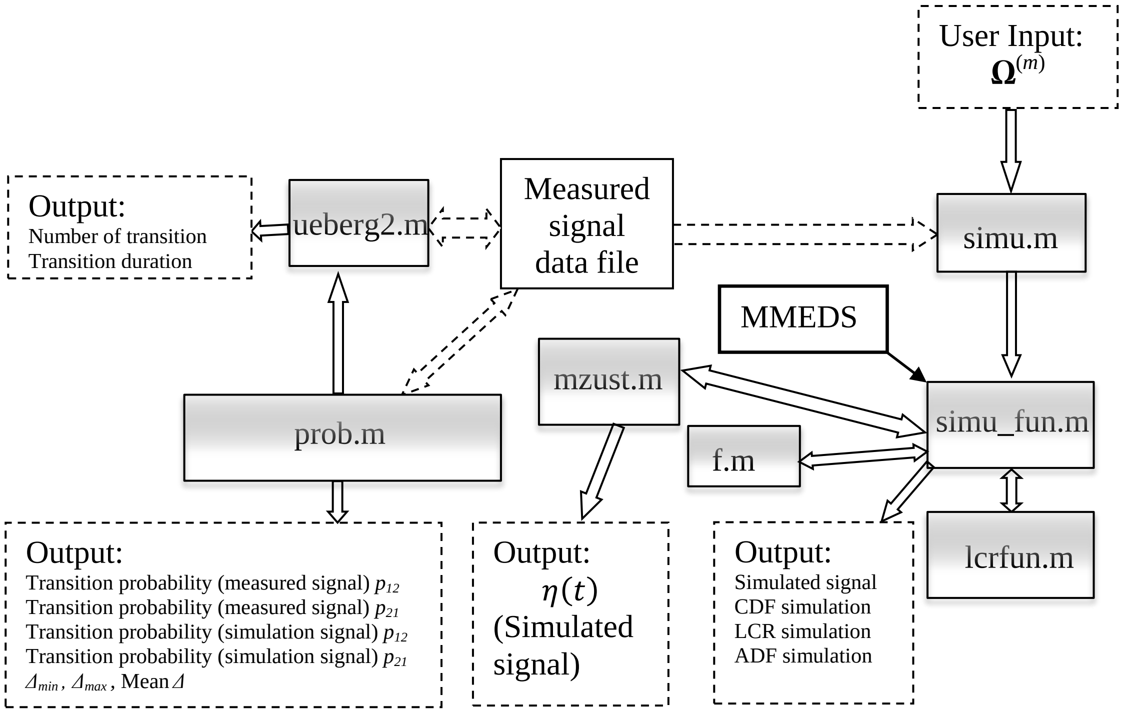

User Input: $\boldsymbol{\Omega}^{(m)}$
Output: Number of transition Transition duration
ueberg2.m
Measured signal data file
simu.m
MMEDS
mzust.m
prob.m
simu_fun.m
f.m
lcrfun.m
Output: Transition probability (measured signal) p_{12} Transition probability (measured signal) p_{21} Transition probability (simulation signal) p_{12} Transition probability (simulation signal) p_{21} $\varDelta_{min}$, $\varDelta_{max}$, Mean $\varDelta$
Output: $\eta(t)$ (Simulated signal)
Output: Simulated signal CDF simulation LCR simulation ADF simulation

Figure 3.4: Block diagram for implementation of the 2 states non-stationary LMS simulation channel

3.1.4 Results and Interpretation

Resulting complementary cumulative distribution function $F_{\eta+}(r)$, level crossing rate $N_{\eta}(r)$, and average duration of fades $T_{\eta+}(r)$, of the reference model and simulation results are shown in Figures 3.5 to 3.8. Simulated fading envelop under suburban area (15° elevation angle) obtained by using the dynamic (2+2)-state model with embedded extended Suzuki process of Type I shown in Figure 3.2. Corresponding simulation results have been obtained by designing the simulation model with the modified MEDS using N_1, N_2, N_3 = 15 sinusoids. Figure 3.7 and 3.8 show theoretical and simulation results are in line with the corresponding measurement for both urban and suburban area conditions. Figure 3.5 illustrates the generated fading envelop by using channel simulator. Relevant m-files are attached to Appendix 3.

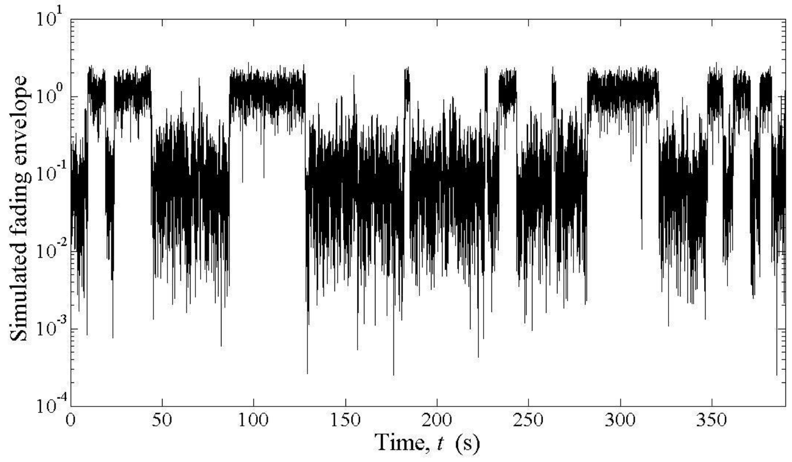


Figure 3.5: Simulated fading envelop under suburban area (15° elevation angle) obtained by using the dynamic (2+2)-state model with embedded extended Suzuki process of Type I.

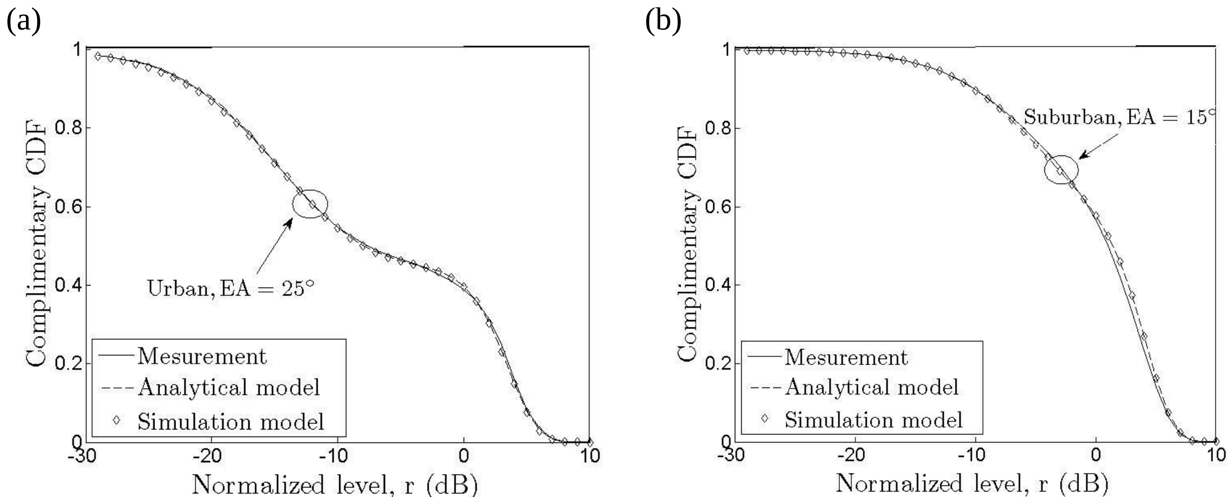


Figure 3.6: Complementary cumulative distribution function for (a) urban area (EA 25°) (b) suburban area (EA 15°) obtained by embedding the extended Suzuki process of Type I in dynamic (2+2)-state model.

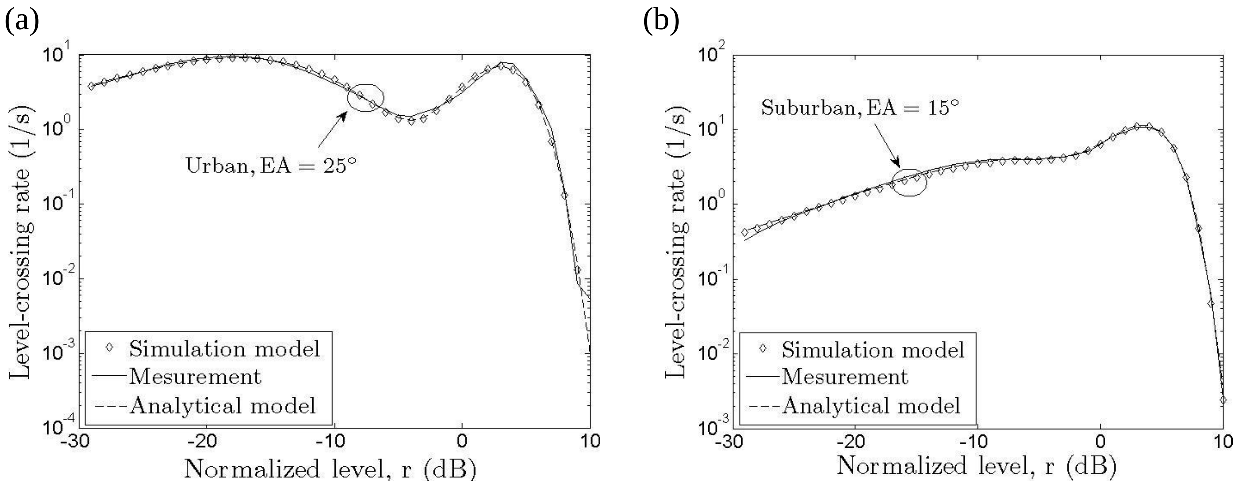


Figure 3.7: Level-crossing rate for (a) urban area (EA 25°) (b) suburban area (EA 15°) obtained by embedding the extended Suzuki process of Type I in dynamic (2+2)-state model

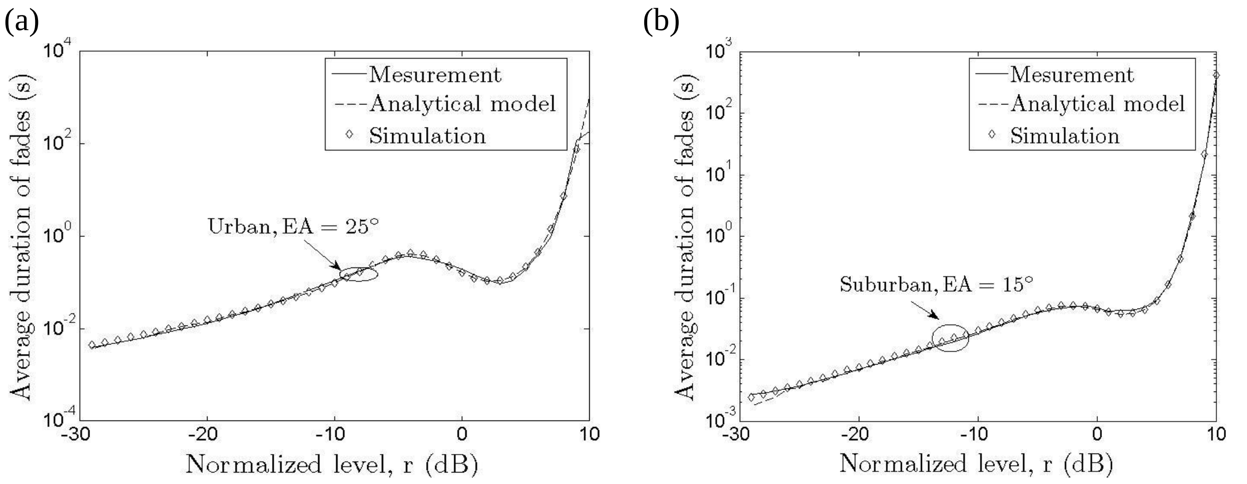


Figure 3.8: Average duration of fades for (a) urban area (EA 25°) (b) suburban area (EA 15°) obtained by embedding the extended Suzuki process of Type I in dynamic (2+2)-state model.

Reviewing the Figures 3.6, 3.7, and 3.8, it can be seen that the second order statistical properties of LCR, ADF and CDF are best fit the simulation model signal with measured signal. This simulator and its testing results are one of the best examples for our research question called "*How can a channel simulator be tested?"*. We can see that second order statistical properties of the ADF, LCR and complimentary CDF of simulation model are almost overlapping with measured model as well as analytical model. These results are given good implication for our research question. Simulator parameters computed by using measured signal and channel simulator were implemented by referring the measured signal and its properties. Simulated fading envelop under suburban area (15° elevation angle) which obtained by using the dynamic (2+2)-state model behave almost all equivalent to the measured signal as shown in Figure 3.5. In addition to that these all evidence provides us regarding correctness of the simulation model.

3.2 SPATIAL CHANNEL SIMULATOR FOR SHADOWING FADING

Received signal may be subjected to slow fading and fast fading. Slow fading or so called shadowing channel model is introduced in this section. The 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 [2] 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. 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 describe here. Next, shadowing process based on the channel parameters (gains and spatial frequencies) obtained from the MEA method is evaluated.

3.2.1 Reference Model

The effect of shadowing generally modelled through a lognormal process $\lambda(x)$ which given by (3.6) referred to distance. In this section, we define the probability density function as well as the cumulative distribution function of the shadowing process. Next, the level-crossing rate and average duration of fades which are considered as advanced statistics are introduced.

3.2.1.1 The PDF and CDF of Spatial Shadowing Process

The reference model of the spatial shadowing process is given by.

$$\lambda(x)=10^{\left(\sigma_L \nu(x)+m_L\right)/20} \tag{3.6}$$

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

$$P_\lambda\left(x\right)=\frac{20}{\sqrt{2\pi}\ln\left(10\right)\sigma_L z}e^{\frac{-\left(20\ln\left(z\right)-m_L\right)^2}{2\sigma_L{}^2}}, z\geq 0 \tag{3.7}$$

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

$$F_\lambda\left(r\right)=P_r\left(\lambda\left(x\right)\leq r\right)=\int_0^r P_r\left(z\right)dz \tag{3.8}$$

The lognormal process plays an important role and it is considered as a key in the performance, analysis of handover procedures as explained in [12]. 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.

3.2.1.2 Spatial Correlation

The spatial autocorrelation function $r_{vv}(\Delta x)$ of *v(x)* provides an insight into how fast the shadow fading process *λ(x)* changes with the distance. Therefor accurate modelling of the $r_{vv}(\Delta x)$ is important and simulation results are compared with four spatial correlations model so called the Gudmundson correlation model, the Gaussian correlation model, the Butterworth correlation model and measurement-based correlation model[2].

The Gudmundson Correlation Model:
According to the Gudmundson correlation model The spatial autocorrelation function $r_{vv}(\Delta x)$ of *v(x) given by* $e^{-|\Delta x|/D_c}$.

The Gaussian Correlation Model

According to the Gaussian correlation model The spatial autocorrelation function $r_{vv}(\Delta x)$ of *v(x) given by* $e^{\left(-|\Delta x|/D_c\right)^2}$.

The Butterworth Correlation Model:
According to the Butterworth correlation model The spatial autocorrelation function $r_{vv}(\Delta x)$ of *v(x) given by* $\sqrt{2}.e^{\left(-\pi\sqrt{2}|\Delta x|/D_2\right)}.\sin(\pi\sqrt{2}|\Delta x|/D_2+\frac{\pi}{4})$.

3.2.1.3 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 to obtain a good approximation of LCR. After some steps of calculation, we derive the following formula corresponding to the spatial shadowing process.

$$N_\lambda(r)=\frac{\sqrt{\gamma}}{2\pi}e^{\frac{(20\ln(r)-m_L)^2}{2\sigma_L^2}},r\geq 0 \tag{3.9}$$

where

$$\gamma = -\frac{d^2}{d\Delta x^2} r_{\nu\nu}(\Delta x)\big|_{\Delta x=0} = -r_{\nu\nu}(0) \tag{3.10}$$

$$T_{\lambda-}(r) = \frac{F_{\lambda-}(r)}{N_\lambda(r)} \tag{3.11}$$

3.2.3 Simulation Model

Structure of the stochastic spatial shadowing simulator is shown in Figure 3.9. According to the sum-of-sinusoids principal $\tilde{\nu}\ (x)$ can denoted by

$$\tilde{\nu}\ (x) = \sum_{n=1}^{N} c_n \cos(2\pi s_n t + \theta_n) \tag{3.12}$$

Where c_n and s_n are constants

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

$$\tilde{r}\ (\Delta x) = E\{\tilde{\nu}\ (x)\ \tilde{\nu}\ (x + \Delta x)\ \}$$

$$= \sum_{n=1}^{N} \frac{{c_n}^2}{2} \cos(2\pi s_n \Delta x) \tag{3.13}$$

$$\tilde{p}_\nu(x) = \begin{cases} 2\int_0^\infty \left[\prod_{n=1}^{N} J_0(2\pi c_n z)\right] \cos(2\pi x z)\, dz & , x \in [\hat{\nu}_{\min}, \hat{\nu}_{\max}] \\ 0 & , otherwise \end{cases} \tag{3.14}$$

Where; $c_n = \sqrt{2/3}$.

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

By applying the transformation of random variable obtained the following:

$$\begin{aligned} \hat{p}_\lambda(y) &= \frac{20\hat{p}_\nu\left(\frac{20\log_{10} y - m_L}{\sigma^2}\right)}{y\sigma_L \ln 10} \\ &= \frac{40}{\sigma_L \ln 10 y} \int_0^\infty \left[\prod_{n=1}^{N} J_0(2\pi c_n z)\right] \cos\left[\frac{2\pi z(20\log_{10} y - m_L)}{\sigma_L}\right] dz \end{aligned} \tag{3.15}$$

And finally the level-crossing rate is defined as:

$$\hat{N}_\lambda(r) = \frac{\sqrt{\hat{\gamma}}}{2\pi} e^{\frac{-(20\log_{10} r - m_L)}{2\sigma_L^2}} \quad , r \geq 0 \tag{3.16}$$

Where; $\tilde{\gamma}\ (x) = 2\pi^2 \sum_{n=1}^{N} (c_n\, s_n)^2$.

The schema below describe the structure of the spatial shadowing simulator

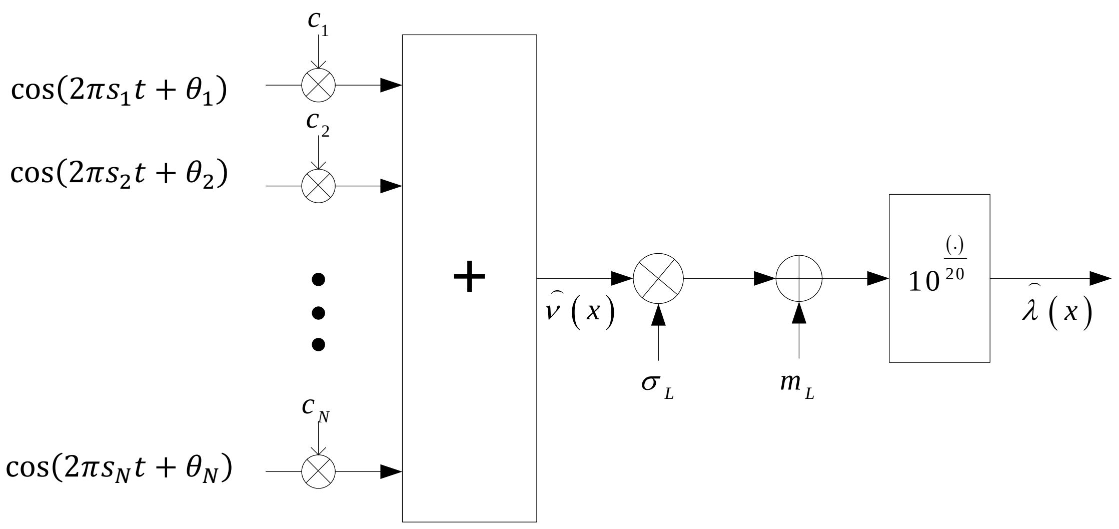

c_1
$\cos(2\pi s_1 t + \theta_1)$
c_2
$\cos(2\pi s_2 t + \theta_2)$
c_N
$\cos(2\pi s_N t + \theta_N)$
+
$\hat{\nu}(x)$
σ_L
m_L
$10^{\frac{(.)}{20}}$
$\hat{\lambda}(x)$

Figure 3.9: Structure of the stochastic spatial shadowing simulator with constant gain c_n constant spatial frequencies s_n , and random phases θ_n.

3.2.4 Implementation

The shadowing processes simulator shown in the Figure 3.9 is implemented under the Matlab environment as a set of m-files containing the different functions and procedures. In the figure below, the possible interactions between the user and our program or simulator are described. 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 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.

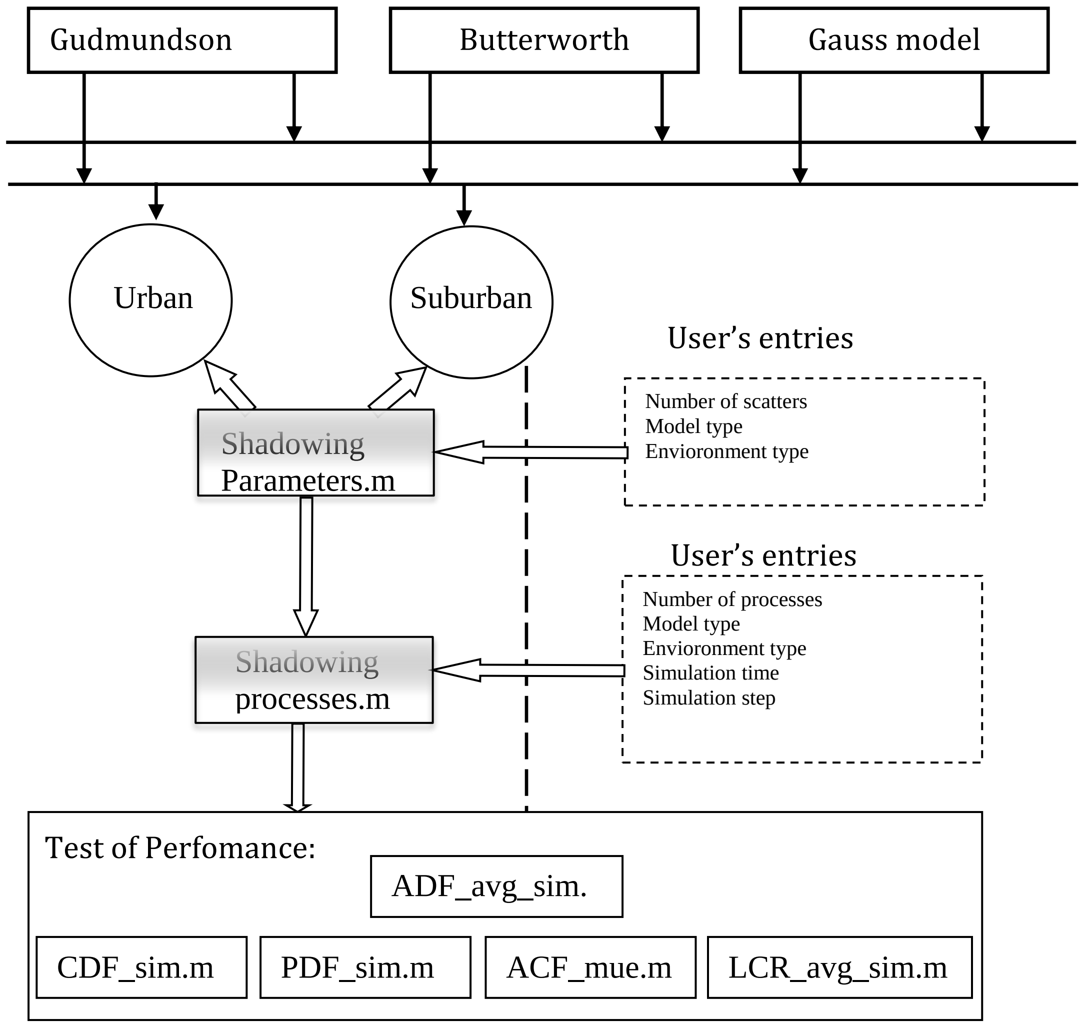


Figure 3.10: The diagram for the implementation of the spatial shadowing process

3.2.5 Results and Interpretation

As a result of the above programs, the following results are obtained. 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 = 25, 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 3.11 shows the probability density function of the simulated signal of $\hat{\lambda}(x)$. It shows the lognormal distribution and we can see that the reference model almost overlap with the simulated model for both suburban and urban area condition for N = 25.

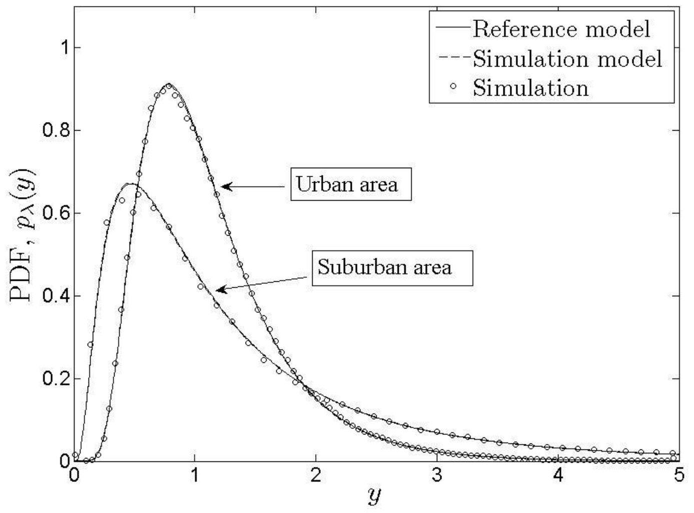
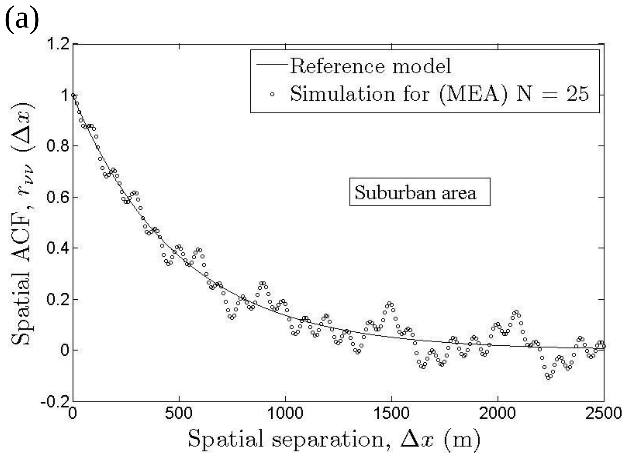


Figure 3.11: The lognormal distribution of the reference model comparison with the simulation model for suburban and urban area ($c_n = \sqrt{2/N}$ and $N = 25$)



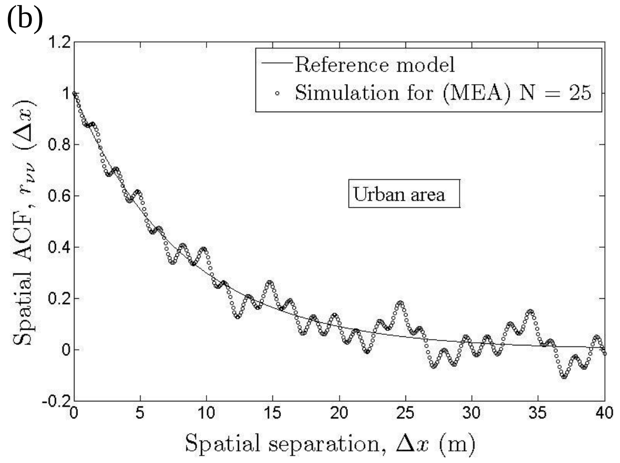
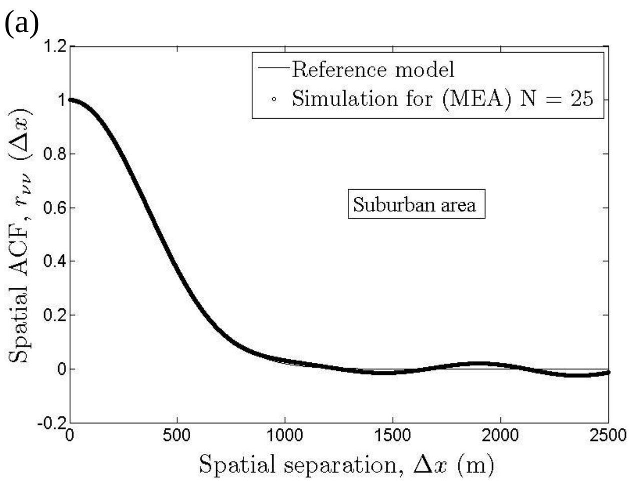


Figure 3.12: Spatial autocorrelation functions of the reference model and simulation for (a) Suburban environment and b) Urban environment under Gudmundson computation method, MEA with $N = 25$.



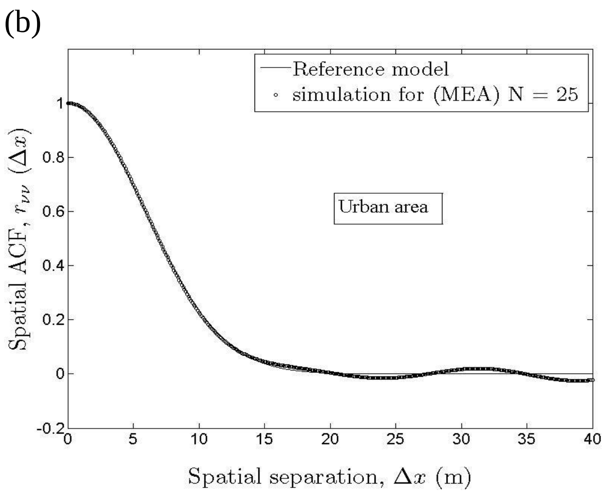
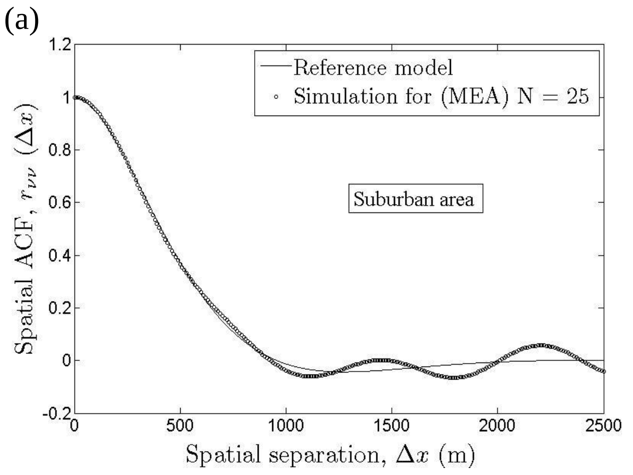


Figure 3.13: Spatial autocorrelation functions of the reference model and simulation for (a) Suburban environment and (b) Urban environment under Gaussian computation method, MEA with N = 25.



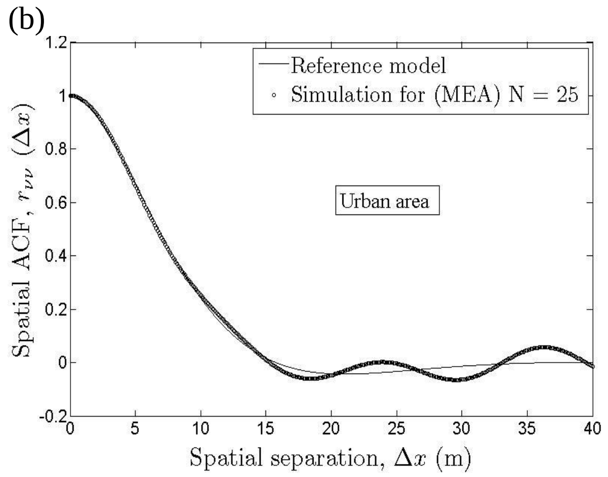
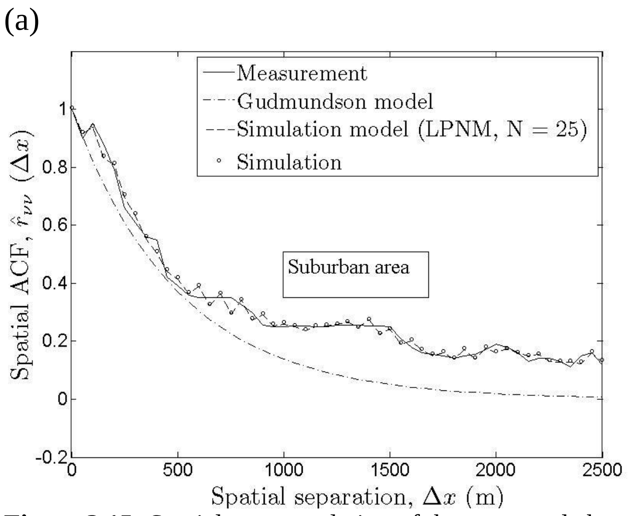


Figure 3.14: Spatial autocorrelation functions of the reference model and simulation for (a) Suburban environment and (b) Urban environment under Butterworth computation method, MEA with N = 25.



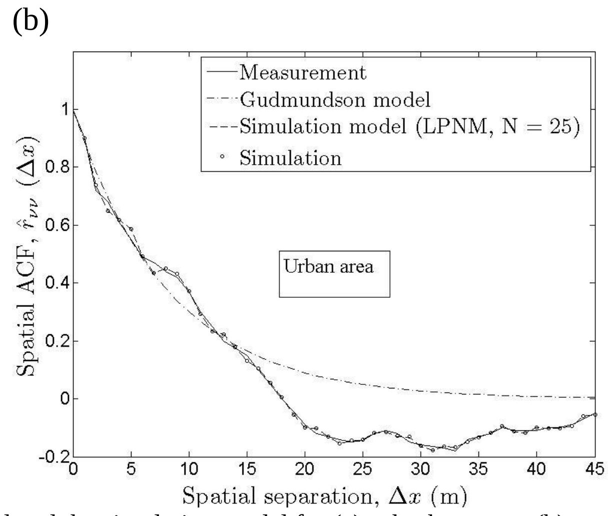


Figure 3.15: Spatial autocorrelation of the measured channel and the simulation model for (a) suburban areas (b) urban areas (Measurement-based correlation model, LPNM with N = 25)

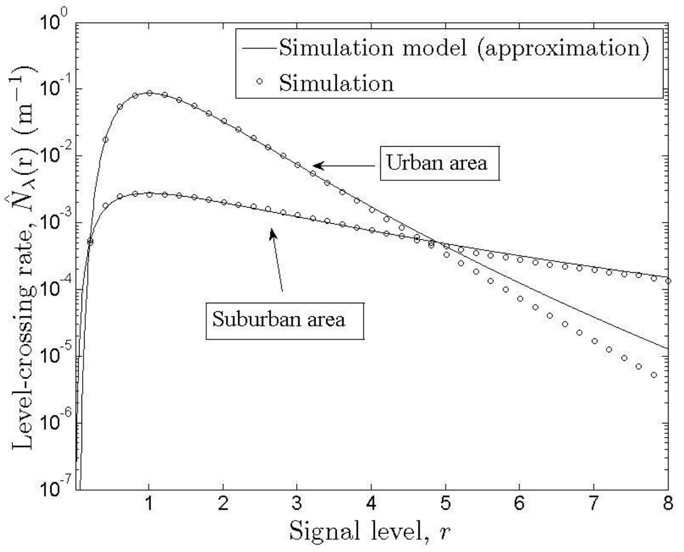


Figure 3.16: Comparison of the exact and approximated solutions for level crossing rate of the simulation model for shadow fading in urban and suburban areas (Measurement-based correlation model, LPNM with N = 25)

Figure 3.15 shows that spatial autocorrelation of the measured channel and the simulation model for (a) suburban areas (b) urban areas under measurement-based correlation model. Simulation results and reference model of the level crossing rate are almost inline for N = 25 and it is shown in Figure 3.16.

It can be concluded that the spatial shadowing process simulator under certain conditions runs efficiently by using some closed models (Gudmundson, Gauss, and Butterworth). The Gaussian model of the shadowing process showed the best performance based on the Figure 3.13 (a) and (b). On the contrary, Gudmundson model is the worst one since, as is shown, some small fluctuations exist on reference model towards simulation model.

Measurement based correlation model developed with L_p-norm method (LPNM). Advantage of the LPNM versus the MEA is that this powerful procedure enables the fitting of the statistical properties of the channel simulator to real world channels. All relative programs for spatial shadowing simulator are listed in Appendix 3.

3.3 MIMO FADING CHANNEL SIMULATORS

Multipath is vital characteristic of data transmission in wireless communication systems. Wireless channel contains different impairment to transmitted signal and channel response. It affects the signal to travel in multipath between transmitter and receiver. Receiver gets the reflection of same symbols in delay versions. Delays or fading occurs due to reflection, refractions, diffractions, shadowing etc. due to the buildings, trees, aircrafts, humidity, temperature etc. Delay or fading could be in form of changing phase or magnitude of signals. Multipath affects and delay profile reduce the channel efficiency, through put and cause corrupted information at receiver. Intelligently multipath effect of MIMO is used to increase capacity of system.

MIMO systems offered a significant improvement of quality of the signal at the receiver side. It increases the channel capacity. MIMO techniques have been invented to overcome some drawbacks on SISO systems. As a consequence, many MIMO channel models have been proposed. The figure below shows the fundamental concept of MIMO systems.

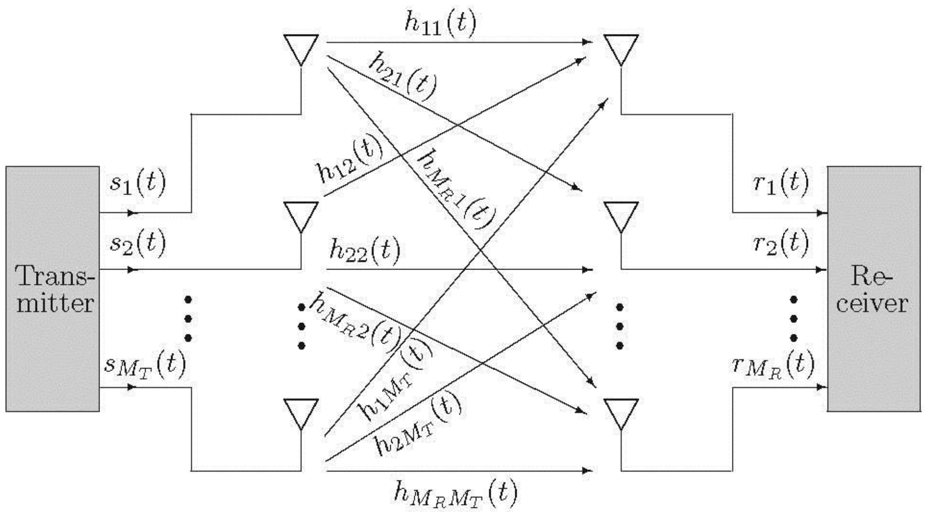


Figure 3.17: Example of MIMO system based on smart antennas

According to antennas system, there are four basic models SISO (Single Input Single Output), SIMO (Single Input Multiple Output), MISO (Multiple Input Single Output), MIMO (Multiple Input Multiple Output). One of the techniques deployed to counter act the multipath fading is MIMO to retrieve the strongest signal from the channel. MIMO assumes to be increase through put, transmission distance, coverage area, BER improvement and reliability of transmission in multipath propagation. MIMO is sending and receiving multiple signals simultaneously. So it is better support to diversity.

Channel modelling is process and method of designing MIMO system to address the issues, problems relevant to the system, and provide analysis to enhance and develop the system by simulating. In order to model MIMO following factors are taken into consideration to get maximum required results [14]

- Free space loss and path loss
- Trees, building which cause Shadowing
- For mobile environment Doppler shift and delay spread due to multi path
- Joint correlation of antenna at sending and receive end
- Channel matrix singular value distribution

The principle of the MIMO system holds onto the fact that it involves multiple inputs and multiple outputs. The above example is the most commonly used in the mobile fading channel in order to find out the appropriate model for MIMO channels.
In this section, initially the one ring model and its main features are shown. Secondly, the tow ring model and the elliptical model are described.

3.3.1 MIMO Channel Simulator Based on One-Ring Model

This model is based on the geometrical one-ring scattering model. A detailed description of this model is as given in the references [15] and [16]. The main idea is to assume that all scatters are located on the same ring around the Mobile station. In this section, some features of one ring model are introduced briefly. In this study, the generalized principle of deterministic channel modelling is used. This model was developed based on some assumptions such as base station is elevated, no obstructed by local scatters and infinite number of local scatters randomly distributed on a ring around the mobile station.

3.3.1.1 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 3.18 contains a detailed description of all components of the model with local scatterers around the mobile station (receiver).

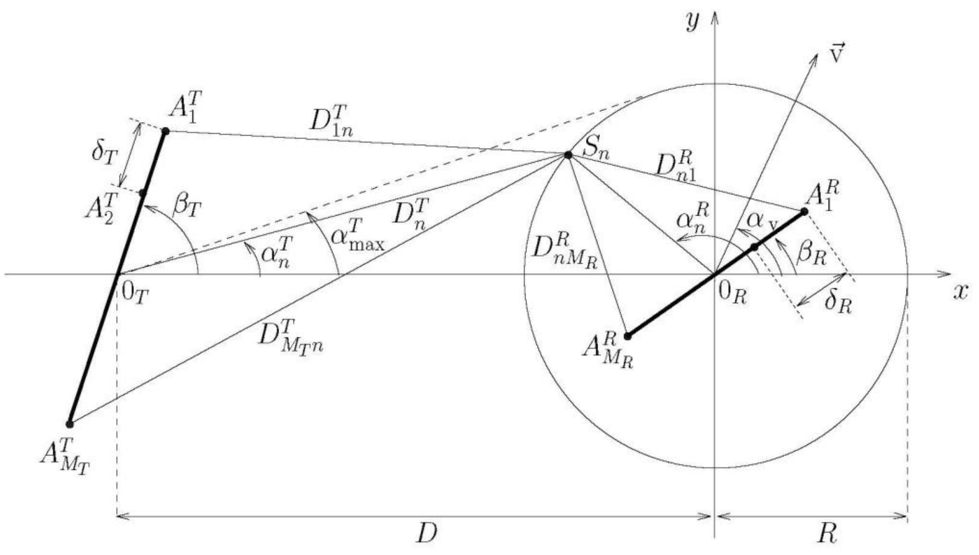


Figure 3.18: Geometrical model (one-ring model)

Where;

δ_T : Antenna element spacing at base station
δ_R : Antenna element spacing at mobile station
β_T : Multi-element antenna tilt angle at base station
β_R : Multi-element antenna tilt angle at mobile station
α_v : Angle of motion
α_n^T : Angle of departure at base station
α_n^R : Angle of arrival at mobile station
$\alpha_{\max}^T$: 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 : Radius of the ring of scatters around the mobile station

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

3.3.1.2 Reference Model

As an example of one-ring based MIMO channel, $M_T \times M_R$ channel with local scatters lying on a ring around the mobile station has been used as a reference model. Later on, considerations are given to the Time ACF and Space-Time CCF only for 2×2 antenna elements and different scattering scenarios.
The channel gains matrix is given by:

$$H_{pq}(t) = \begin{pmatrix} h_{11}(t) & \cdots & h_{1q}(t) \\ \vdots & \ddots & \vdots \\ h_{p1}(t) & \cdots & h_{pq}(t) \end{pmatrix} \tag{3.17}$$

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 $A_l^T - A_k^R$ link is approximated by:

$$h_{kl}(t) = \lim_{N\to\infty} \frac{1}{\sqrt{N}} \sum_{n=1}^{N} g_{kln} e^{j(2\pi f_n t + \theta_n + \theta_0)} \tag{3.18}$$

When implementing the model, we consider the number of scatters N finite. Therefore, the equation (1.18) will be modified as follows:

$$\hat{h}_{kl}(t) = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} a_{ln} b_{kn} e^{j(2\pi f_n t + \theta_n)} \tag{3.19}$$

Where;

$$a_{ln} = e^{j\pi(M_T - 2l+1)\frac{\delta_T}{\lambda_0}\left[\cos(\beta_T) + \alpha_{\max}^T \sin(\beta_T)\sin\left(\alpha_n^R\right)\right]}, \tag{3.20}$$

$$b_{kn} = e^{j\pi(M_R - 2k+1)\frac{\delta_R}{\lambda_0}\cos\left(\alpha_n^R - \beta_R\right)}, \qquad k = 1,\ldots,M_R \tag{3.21}$$

$$f_n = f_{\max}\cos\left(\alpha_n^R - \alpha_v\right) \tag{3.22}$$

The phases θ_n are independent and identically distributed (i. i. d) random variables uniformly distributed over $\left[0,\ 2\pi\right]$. λ_0 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:

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

by just substituting a_n and b_n respectively by the complex conjugate value a_n^* and b_n^* where $a_{ln} = e^{j\pi(M_T - 2l+1)\frac{\delta_T}{\lambda_0}\left[\cos(\beta_T) + \alpha_{\max}^T \sin(\beta_T)\sin\left(\alpha_n^R\right)\right]}$, and $b_{kn} = e^{j\pi(M_R - 2k+1)\frac{\delta_R}{\lambda_0}\cos\left(\alpha_n^R - \beta_R\right)}$, . Thus, $h_{12}\left(t\right)$ and $h_{21}\left(t\right)$ are obtained by replacing respectively a_n and b_n by a_n^* and b_n^* . However, $h_{22}\left(t\right)$ 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 [17]:

$$\rho_{11,22}\left(\delta_T,\delta_R,\tau\right) := E\left\{h_{11}\left(t\right)h_{22}^*\left(t+\tau\right)\right\}|_{\theta_n,\alpha_n^R}, \tag{3.24}$$

$$\rho_{11,22}\left(\delta_T,\delta_R,\tau\right) = \lim_{N\to\infty}\frac{1}{N}\sum_{n=1}^{N} E\left\{a_{1n}^2 b_{1n}^2 e^{-j2\pi f_n \tau}\right\}|_{\alpha_n^R}, \tag{3.25}$$

Assume that the discrete AoA α^R is a continuous random variable with $p\left(\alpha^R\right)$. 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:

$$\rho_{11,22}\left(\delta_T,\delta_R,\tau\right) = \int_{-\pi}^{\pi} a^2\left(\delta_T,\alpha^R\right)b^2\left(\delta_R,\alpha^R\right)e^{-j2\pi f_n(\alpha^R)\tau}p\left(\alpha^R\right)d\alpha^R, \tag{3.26}$$

Where

$$p_{\alpha^R}\left(\alpha^R\right) = \frac{1}{2\pi I_0\left(\kappa\right)}e^{\kappa\cos\left(\alpha^R - m_\alpha\right)}, \quad \alpha^R \in \left(0,\ 2\pi\right], \tag{3.27}$$

The time auto-correlation function can be derived from the space-time cross-correlation function simply by setting $\delta_R = \delta_T$ =0. Then, the Time ACF is expressed by:

$$
\begin{aligned}
r_{h_{kl}}(\tau) &:= E\left\{h_{kl}(t)h_{kl}^{*}(t+\tau)\right\} = \rho_{11,22}(0,0,\tau) \\
&= \int_{-\pi}^{\pi} e^{-j2\pi f_{\max}\cos\left(\alpha^{R}-\alpha_{v}\right)\tau} p_{\alpha^{R}}\left(\alpha^{R}\right)d\alpha^{R}
\end{aligned}
\tag{3.28}
$$

3.3.1.2 Simulation Model

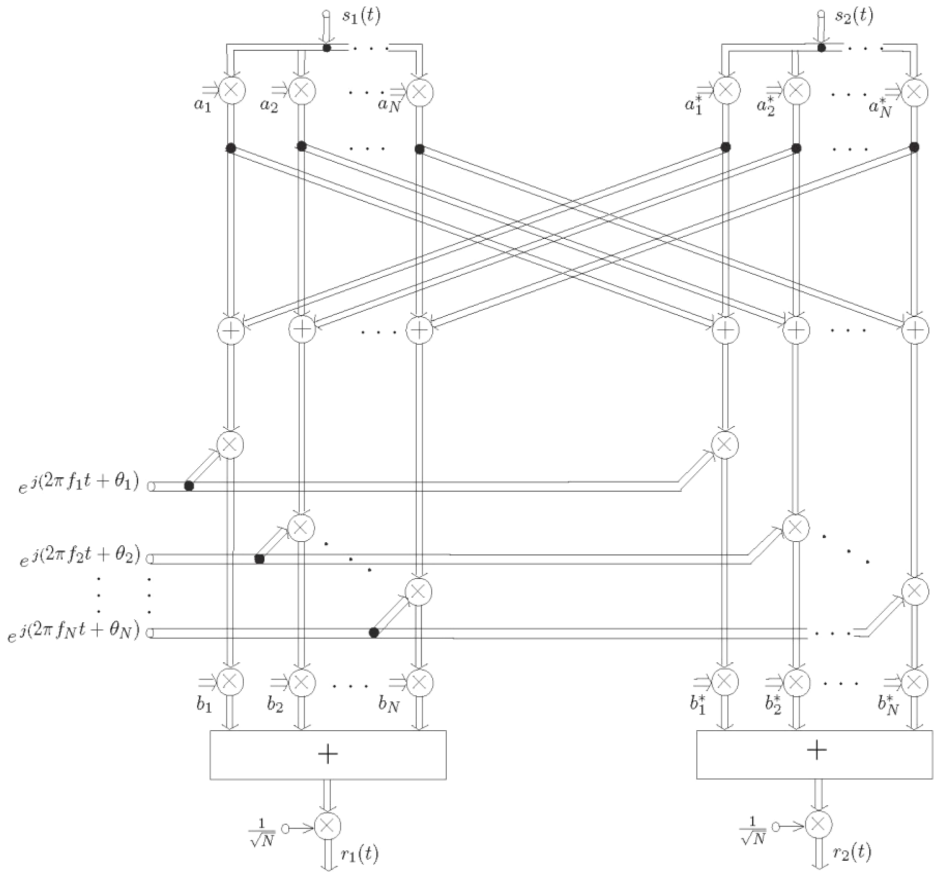


Figure 3.19: Structure of the one-ring 2x2 MIMO channel simulator [2].

The structure of the one-ring 2x2 MIMO channel simulator is shown in Figure 3.19. The simulation model is obtained by assuming that the number of scatters N is finite. The expression of the simulation model is shown here. The complex channel gain of the link from A_1^T to A_1^R link is modelled as:

$$
\hat{h}_{11}(t) = \frac{1}{\sqrt{N}}\sum_{n=1}^{N} a_n b_n e^{j(2\pi f_n t+\theta_n)}
\tag{3.29}
$$

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 3.3.1.1.

The space-time cross-correlation function of the simulation is expressed by the equation below:

$$\begin{aligned}\tilde{\rho}_{11,22}\left(\delta_T,\delta_R,\tau\right) &= \left\langle \tilde{h}_{11}\left(t\right)\tilde{h}_{22}^{*}\left(t+\tau\right)\right\rangle \\ &= \frac{1}{N}\sum_{n=1}^{N} a_n^2\left(\delta_T\right) b_n^2\left(\delta_R\right) e^{-j2\pi f_n \tau}\end{aligned} \tag{3.30}$$

Similarly to the reference model, the Time auto-correlation function can be obtained by replacing $\delta_T = \delta_R = 0$ in Space-Time cross-correlation function. Thus, we obtained:

$$\begin{aligned}\hat{r}_{h_{kl}}\left(\tau\right) &= \left\langle \hat{h}_{kl}\left(t\right)\hat{h}_{kl}^{*}\left(t+\tau\right)\right\rangle \\ &= \frac{1}{N}\sum_{n=1}^{N} e^{-j2\pi f_{\max}\cos\left(\alpha_n^R-\alpha_v\right)\tau}\end{aligned} \tag{3.31}$$

The idea behind the EMEDS for isotropic scattering around the receiver is to employ the EMEDS to determine the set of discrete AOAs. This method is a special case of the generalized method of exact Doppler spread (GMEDS$_q$) which is described in details in [2].

$$\alpha_n^R = \frac{2\pi}{N}\left(n-\frac{1}{2}\right)+\alpha_0^R,\ n=1,\ldots,N, \tag{3.32}$$

Where α_0^R is called angle-of-rotation. The angle-of-rotation α_0^R is defined as

$$\alpha_0^R := \frac{\alpha_n^R-\alpha_{n-1}^R}{4} = \frac{\pi}{2N}. \tag{3.33}$$

In this study, for the LPNM method, to compute the AoA, two error functions related respectively to the time auto-correlation function and the space-time cross-correlation function was optimised. Therefore, we minimized the following two L_p-norms:

$$E_1^{(p)} := \left\{\frac{1}{\tau_{\max}}\int_0^{\tau_{\max}}\left|r_{h_{11}}\left(\tau\right)-\hat{r}_{h_{11}}\left(\tau\right)\right|^p d\tau\right\}^{1/p} \tag{3.34}$$

$$E_2^{(p)} = \left\{\frac{1}{\delta_{\max}^T\delta_{\max}^R}\int_0^{\delta_{\max}^T}\int_0^{\delta_{\max}^R}\left|\rho\left(\delta_T,\delta_R\right)-\hat{\rho}\left(\delta_T,\delta_R\right)\right|^p d\delta_R d\delta_T\right\}^{1/p} \tag{3.35}$$

We found out that when we implemented these tow functions, which we called weighted function $E^{(p)} = w_1E_1^{(p)} + w_2E_2^{(p)}$ $\left(w_1, w_2 \in \left[0,1\right]\right)$, 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.

3.3.1.3 Implementation

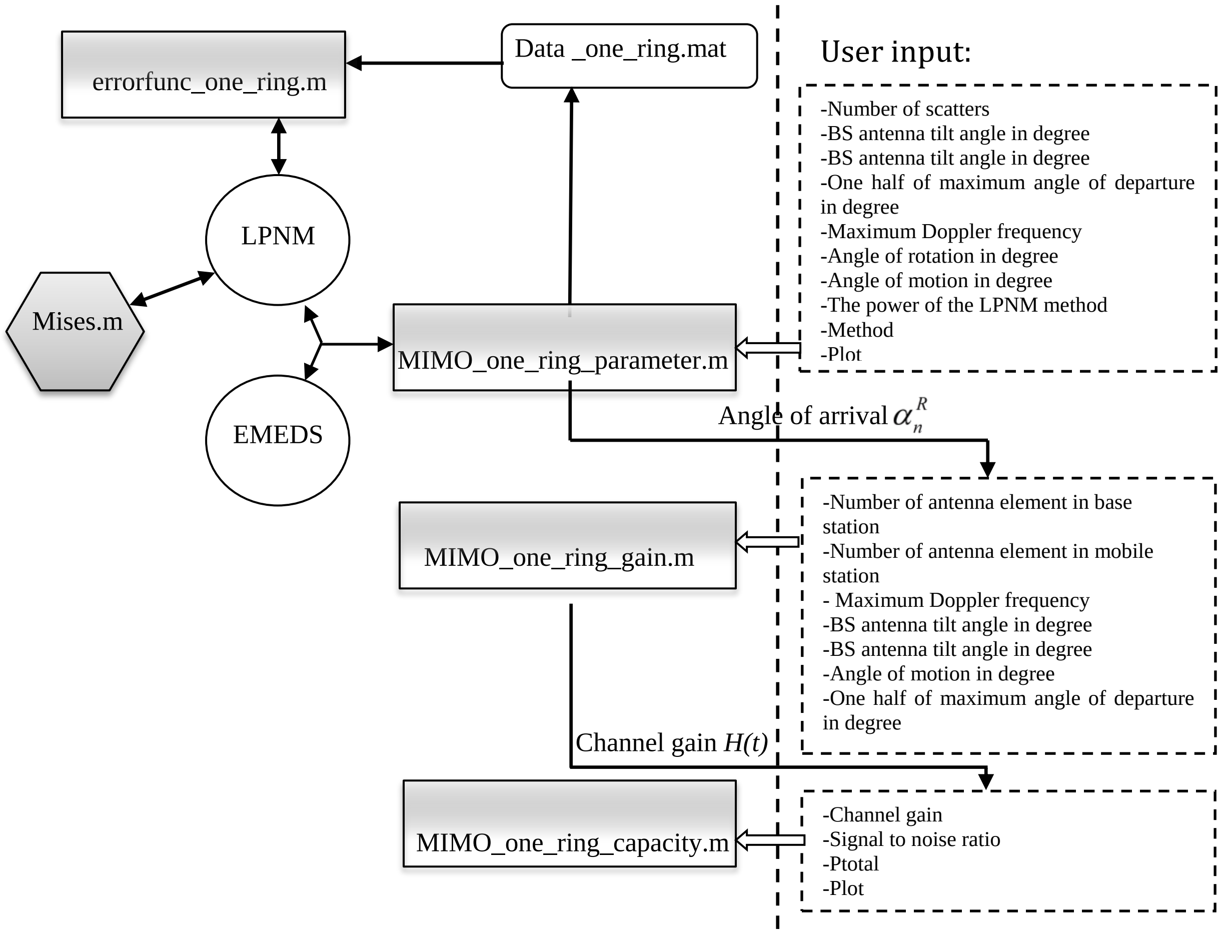

errorfunc_one_ring.m
Data _one_ring.mat
User input:
-Number of scatters
-BS antenna tilt angle in degree
-BS antenna tilt angle in degree
-One half of maximum angle of departure in degree
-Maximum Doppler frequency
-Angle of rotation in degree
-Angle of motion in degree
-The power of the LPNM method
-Method
-Plot
LPNM
Mises.m
MIMO_one_ring_parameter.m
EMEDS
Angle of arrival α_n^R
MIMO_one_ring_gain.m
-Number of antenna element in base station
-Number of antenna element in mobile station
- Maximum Doppler frequency
-BS antenna tilt angle in degree
-BS antenna tilt angle in degree
-Angle of motion in degree
-One half of maximum angle of departure in degree
Channel gain $H(t)$
MIMO_one_ring_capacity.m
-Channel gain
-Signal to noise ratio
-Ptotal
-Plot

Figure 3.20: The diagram for the implementation of a MIMO channel simulator based on one-ring geometrical model

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

- First, the angles of arrivals using two fundamental methods are determined. 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, the channel gains by using the obtained parameters for multiple uncorrelated fading waveforms are computed. To ensure this independency, the parameters using the LPNM method with different values of p are computed.

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

3.3.1.4 Results and Interpretations

The performance of the simulation model can be tested by comparing its statistical properties with its reference model. In this study, time ACF and 2D space CCF are considered to compare both models. In the following, we illustrate the simulation results for different parameters values coming from the EMEDS 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. We have chosen parameters as follows. $\beta_R = \beta_T = 90$, $\alpha^T_{max} = 2^0$ and $f_{max} = 91$ Hz. Here we used the von Mises density to compute AOA with parameter κ and m_α.

Attention: Results were based on the given parameters for Matlab program.

Isotropic scattering (κ =0):

```
[alpha]=MIMO_one_ring_parameters(25,90,90,2,0,91,0,180,1,'emed',1)
```

Non-isotropic scattering (κ =10):

```
[alpha]=MIMO_one_ring_parameters(25,90,90,2,10,91,0,180,2,'lpnm',1)
```

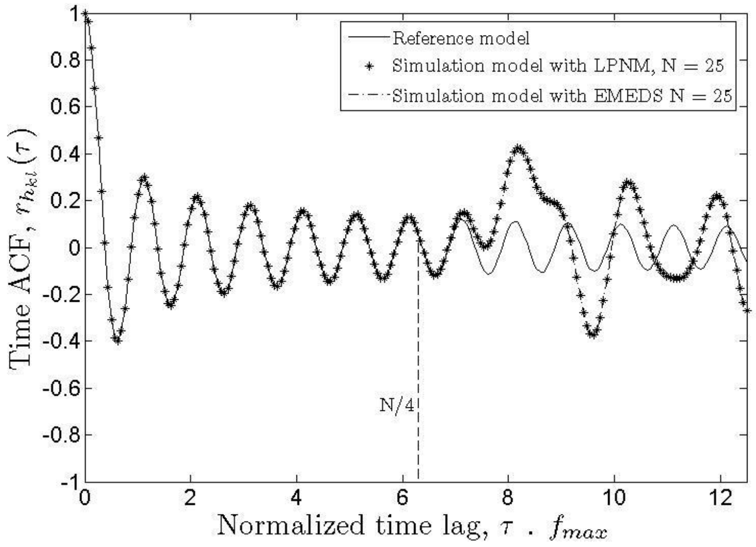


Figure 3.21: Time ACF reference model and simulation model for isotropic scattering (κ = 0)

The figures shown 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 $\delta_R/\lambda = N/4$ and $\delta_T/\lambda = N/4$. Thus, we prove the consistency of the new method to find the best performance. With the increase of number of scatters N, it can be obtained close mapping of the models. Therefore need is arise to achieve an efficient simulator.

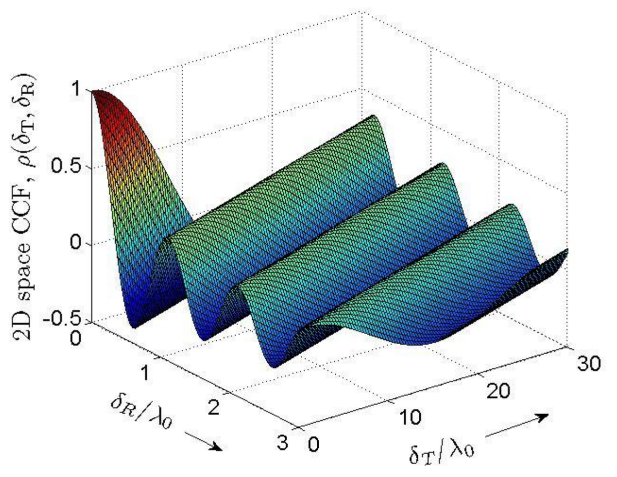


Figure 3.22: The 2D space CCF $\rho(\delta_T, \delta_R)$ of the reference model for isotropic scattering (κ =0, β_R = β_T = 90, α^T_{max}= 2^0)

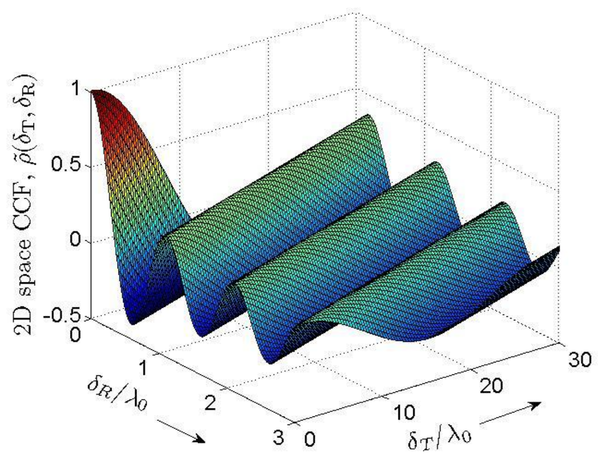


Figure 3.23: The 2D space CCF $\tilde{\rho}(\delta_T, \delta_R)$ of the simulation model for isotropic scattering (κ = 0, β_R = β_T= 90, α^T_{max}= 2^0, EMEDS, N = 25)

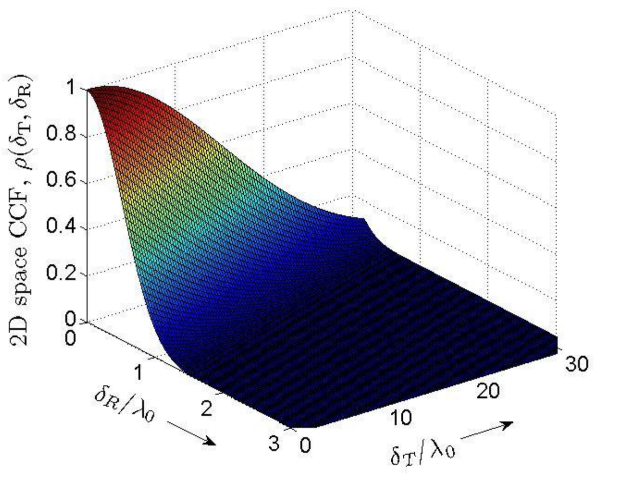


Figure 3.24: The 2D space CCF $\rho(\delta_T, \delta_R)$ of the reference model for non-isotropic scattering (κ = 0, β_R = β_T = 90, α^T_{max}= 2^0)

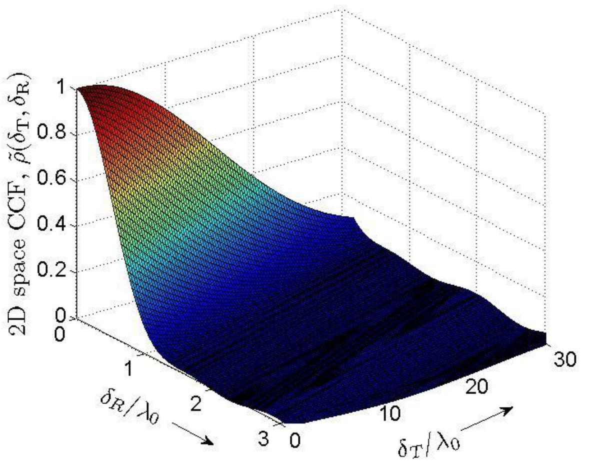


Figure 3.25: The 2D space CCF $\tilde{\rho}(\delta_T, \delta_R)$ of the simulation model for non-isotropic scattering (κ = 0, β_R = β_T = 90, α^T_{max} = 2^0, LPNM, N = 25)

By referring Figures 3.21 to 3.25 it can concluded that the space-time simulation model has nearly the same temporal and spatial correlation properties as reference model.

In this section studies are carried out to find how reference model derived and how its spatial/ temporal correlation properties can be analysed. Here, scattering environment around the receiver is modelled by invoking the geometrical one ring scattering model. Then we learn how ergodic MIMO channel simulator can be derived from the reference model and how its model parameter can be computed and then employed computed parameter in simulation model. All relative programs for MIMO one-ring channel simulator are listed in the Appendix 3.

3.3.2 The Two-Ring MIMO Mobile to Mobile Channel Model

Figure 3.26 illustrates propagation scenario of the geometrical two-ring scattering model for a narrowband mobile to mobile 2x2 MIMO channel. Assumptions are made that there is no line of sight component and only consider local scatters [19]. One of the objectives of this research is to find a relation between the AoA and the rest of parameters [2]. For simplicity, it is assumed that both transmitter and receiver are equipped with only two omnidirectional antennas. The both transmitter and receiver are assumed to be moved as shown in Figure 3.26. In this model, only local scattering is considered, because it is assumed that the contribution of remote scatters to the total received power can be neglected due to high path loss.

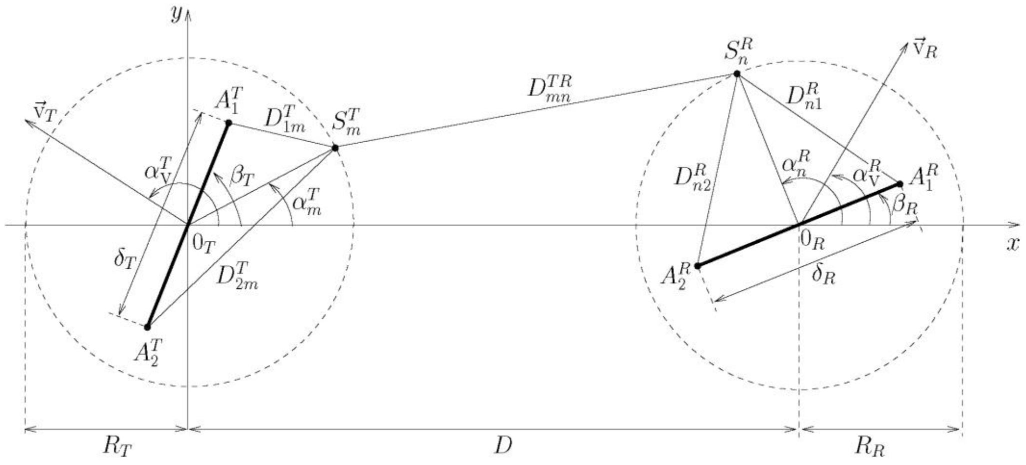


Figure 3.26: The geometrical two-ring model for a 2x2 MIMO channel with local scatters around a mobile transmitter (left) and mobile receiver (right).

Where;

δ_T : Antenna element spacing at base station

δ_R : Antenna element spacing at mobile station

β_T : Multi-element antenna tilt angle at base station

β_R : Multi-element antenna tilt angle at mobile station

α_v : Angle of motion

α_m^T : Angle of departure at base station

α_n^R : Angle of arrival at mobile station

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

R_T : Radius of the ring of scatters around the base station

R_R : Radius of the ring of scatters around the mobile station

3.3.2.1 Reference Model

Starting from the geometrical two-ring model, the number of scatters around the receiver and transmitter is assumed to be infinite. Expansion has been done to the general expression of the gains presented in reference referring [2] to the parameters of the two-ring model. Thus the expression for the channel gains is as follows.

$$h_{11}(t) = \lim_{\substack{M\to\infty \\ N\to\infty}} \frac{1}{\sqrt{MN}} \sum_{m=1}^{M}\sum_{n=1}^{N} g_{mn} e^{j\left[\left(2\pi\left(f_m^T + f_n^R\right)t + \theta_{mn} + \theta_0\right)\right]}, \tag{3.37}$$

Where;

$$a_m = e^{j\pi(\delta_T/\lambda_0)\cos\left(\alpha_m^T - \beta_T\right)} \tag{3.38}$$

$$b_n = e^{j\pi(\delta_R/\lambda_0)\cos\left(\alpha_n^R - \beta_R\right)} \tag{3.39}$$

$$c_{mn} = e^{j\frac{2\pi}{\lambda_0}\left(R_T\cos(\alpha_m^T) - R_R\cos\alpha_n^R\right)}, \tag{3.40}$$

$$g_{mn} = a_m b_n c_{mn} \tag{3.41}$$

$$f_m^T = f_{\max}^T \cos\left(\alpha_m^T - \alpha_v^T\right) \tag{3.42}$$

$$f_n^R = f_{\max}^R \cos\left(\alpha_n^R - \alpha_v^R\right) \tag{3.43}$$

$$\theta_0 = -\frac{2\pi}{\lambda_0}\left(R_T + D + R_R\right) \tag{3.44}$$

The phase θ_0 can be set to zero in (3.37). 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, stochastic channel matrix $H(t)$ which describes completely the reference model of the proposed two-ring frequency-non-selective mobile-to-mobile MIMO fading channel [2, 19, and 21] is formed.

$$H(t) = \begin{pmatrix} h_{11}(t) & h_{12}(t) \\ h_{21}(t) & h_{22}(t) \end{pmatrix}, \tag{3.45}$$

In addition, the capacity that characterizes the channel is designated here [22]:

$$C(t) = \log_2\left[\det\left(I_2 + \frac{P_T}{2N_0} H(t) H^H(t)\right)\right] \tag{3.46}$$

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:

$$\rho_{11,22}\left(\delta_T,\delta_R,\tau\right)=\rho_T\left(\delta_T,\tau\right)\cdot\rho_R\left(\delta_R,\tau\right) \tag{3.47}$$

Where;

$$\rho_T\left(\delta_T,\tau\right)=\int_{-\pi}^{\pi}a^2\left(\delta_T,\alpha^T\right)e^{-j2\pi f^T(\alpha^T)\tau}p_{\alpha^T}(\alpha^T)d\alpha^T \tag{3.48}$$

$$\rho_R\left(\delta_R,\tau\right)=\int_{-\pi}^{\pi}b^2\left(\delta_R,\alpha^R\right)e^{-j2\pi f^R(\alpha^R)\tau}p_{\alpha^R}(\alpha^R)d\alpha^R \tag{3.49}$$

Where;

$$a\left(\delta_T,\alpha^T\right)=e^{j\pi(\delta_T/\lambda_0)\cos\left(\alpha^T-\beta_T\right)} \tag{3.50}$$

$$b\left(\delta_R,\alpha^R\right)=e^{j\pi(\delta_R/\lambda_0)\cos\left(\alpha^R-\beta_R\right)} \tag{3.51}$$

$$f^T\left(\alpha^T\right)=f_{\max}^T\cos\left(\alpha^T-\alpha_v^T\right) \tag{3.52}$$

$$f^R\left(\alpha^R\right)=f_{\max}^R\cos\left(\alpha^R-\alpha_v^R\right) \tag{3.53}$$

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

$$r_{h_{kl}}\left(\tau\right)=\rho_T\left(0,\tau\right)\cdot\rho_R\left(0,\tau\right) \tag{3.54}$$

3.3.2.2 Simulation Model

The Structure of the two-ring 2x2 MIMO channel simulator for mobile-to-mobile channels is shown in Figure 3.27. The simulation model should be as closed as possible to the reference model. Therefore, the numbers of scatters are set to finite number. Then, the diffuse component of the $A_1^T-A_1^R$ link can be expressed as

$$\hat{h}_{11}(t)=\frac{1}{\sqrt{MN}}\sum_{m=1}^{M}\sum_{n=1}^{N}g_{mn}e^{j\left[\left(2\pi\left(f_m^T(\alpha_m^T)+f_n^R(\alpha_n^R)\right)t+\theta_{mn}\right)\right]}, \tag{3.55}$$

Where g_{mn}, f_m^T, f_n^R, and θ_{mn} keep the same expressions as in the previous section. Here θ_m and θ_n are random variables independent and uniformly distributed over [0, 2π) . Therefore, θ_{mn} should be uniformly distributed over the same interval.

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

$$\hat{\rho}_{11,22}\left(\delta_T,\delta_R,\tau\right)=\hat{\rho}_T\left(\delta_T,\tau\right)\cdot\hat{\rho}_R\left(\delta_R,\tau\right) \tag{3.56}$$

where

$$\hat{\rho}_T\left(\delta_T,\tau\right)=\frac{1}{M}\sum_{m=1}^{M}a_m^2\left(\delta_T\right)e^{-j2\pi f_m^T\tau} \tag{3.57}$$

$$\hat{\rho}_R\left(\delta_R,\tau\right)=\frac{1}{N}\sum_{m=1}^{M}b_n^2\left(\delta_T\right)e^{-j2\pi f_n^R\tau} \tag{3.58}$$

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

$$r_{h_{kl}}\left(\tau\right)=\hat{\rho}_T\left(0,\tau\right)\cdot\hat{\rho}_R\left(0,\tau\right), \qquad \forall k,l\in\{1,2\}. \tag{3.59}$$

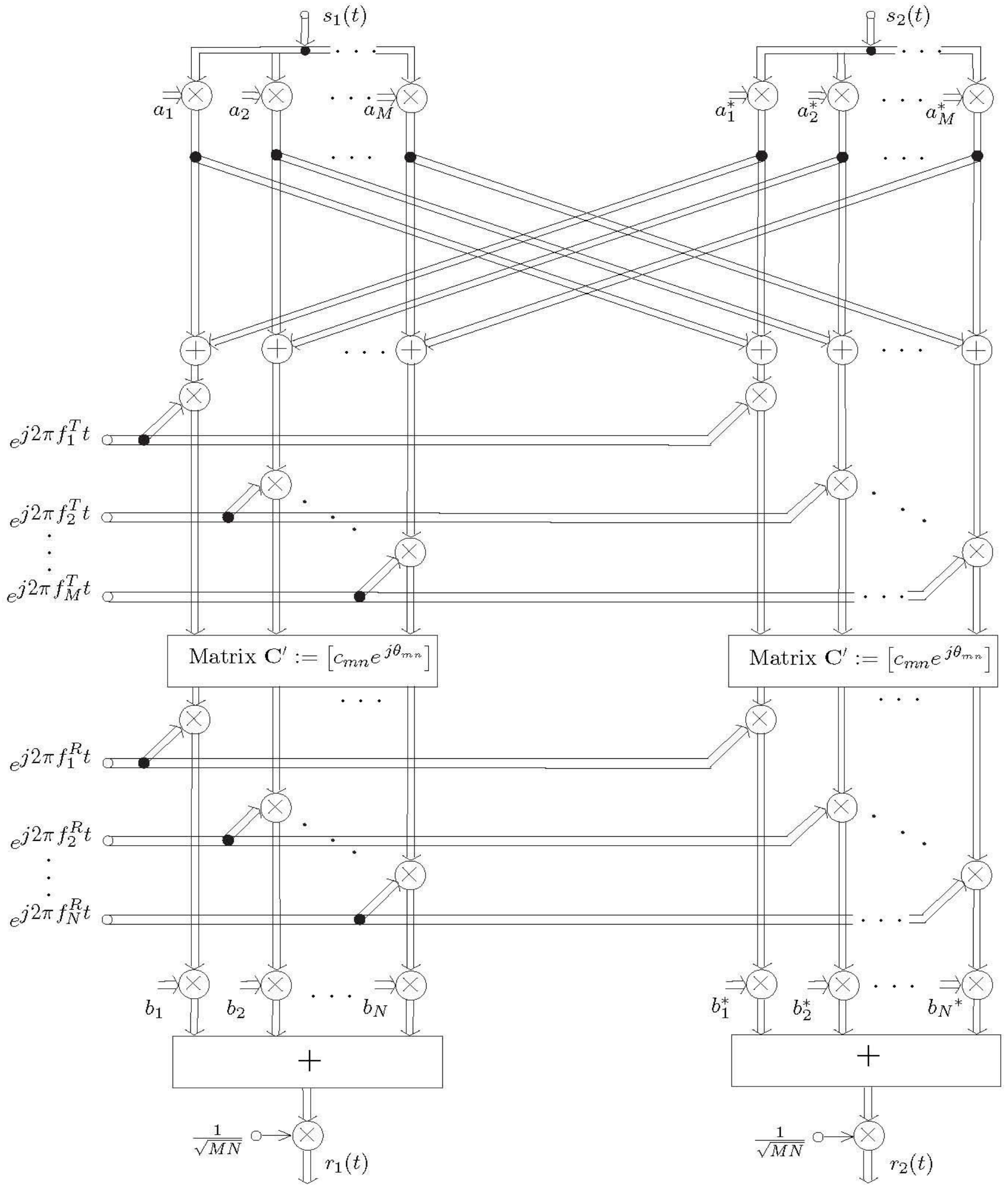


Figure3.27: Structure of the two-ring 2x2 MIMO channel simulator for mobile-to-mobile channels [2].

Similarly to one-ring model, we proceed the same way to compute the AoA and AoD using the same parameters computation method for isotropic scattering. Thus, this method (EMEDS) explained previously. However,

$$\alpha_m^T = \frac{2\pi}{M}\left(m - \frac{1}{4}\right) + \alpha_v^T, \qquad m = 1, 2, ..., M, \tag{3.60}$$

$$\alpha_n^R = \frac{2\pi}{N}\left(n - \frac{1}{4}\right) + \alpha_v^R, \qquad n = 1, 2, ..., N, \tag{3.61}$$

For non-isotropic scattering we can't use EMEDS to compute AoA and AoD, Therefore. We use MMEA to compute the AoA and AoD for non-isotropic scattering by using equations (3.62) and (3.63).

$$\frac{m-1/4}{M} - \int_{m_\alpha^T - \pi}^{\alpha_m^T} p_{\alpha^T}(\alpha^T)d\alpha^T = 0, \qquad m = 1, 2, ..., M, \tag{3.62}$$

$$\frac{n-1/4}{N} - \int_{m_\alpha^R - \pi}^{\alpha_n^R} p_{\alpha^R}(\alpha^R)d\alpha^R = 0, \qquad n = 1, 2, ..., N. \tag{3.63}$$

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

3.3.2.2 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). By combining these two results the channels' gains as well as the impulse response which characterizes the behaviour of the model are determined.

The Figure 3.28 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_alphaT_two_ring.m and gen_alphaR_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 EMEDS method.
- The option PLOT is provided to sketch the performance criteria, such as temporal ACF and space-time CCF.

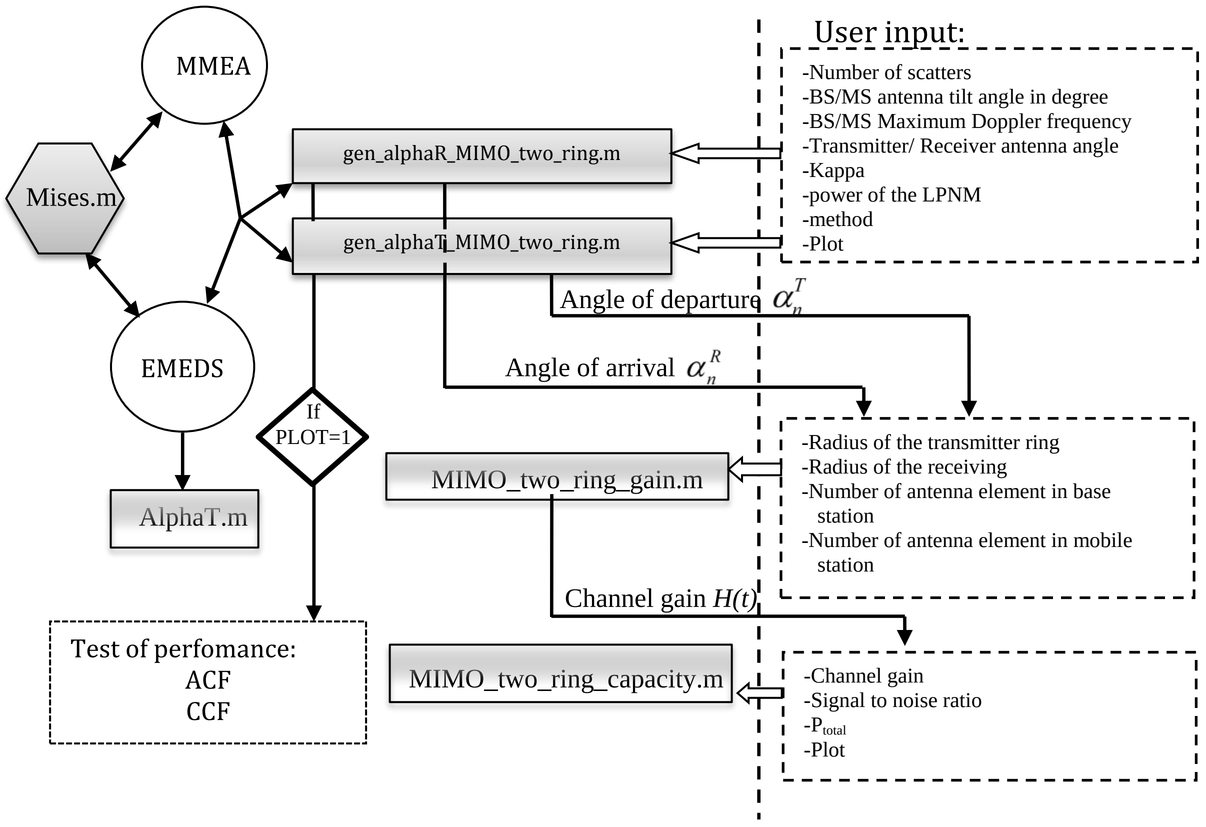

MMEA
Mises.m
EMEDS
AlphaT.m
gen_alphaR_MIMO_two_ring.m
gen_alphaT_MIMO_two_ring.m
If PLOT=1
Test of perfomance:
ACF
CCF
MIMO_two_ring_gain.m
MIMO_two_ring_capacity.m
Angle of departure α_n^T
Angle of arrival α_n^R
Channel gain $H(t)$
User input:
-Number of scatters
-BS/MS antenna tilt angle in degree
-BS/MS Maximum Doppler frequency
-Transmitter/ Receiver antenna angle
-Kappa
-power of the LPNM
-method
-Plot
-Radius of the transmitter ring
-Radius of the receiving
-Number of antenna element in base station
-Number of antenna element in mobile station
-Channel gain
-Signal to noise ratio
-P_{total}
-Plot

Figure 3.28: The diagram for the implementation of a MIMO channel simulator based on two-ring geometrical model

Attention: Results were based on the given parameters for Matlab program.

Isotropic scattering (κ = 0):

```
[alphaT]=gen_alphaT_MIMO_tow_ring(40,180,91,90,0,2,'emed',1)
```

Non-isotropic scattering (κ = 40):

```
[alphaT]=gen_alphaT_MIMO_tow_ring(40,180,91,90,40,2,'mmea',1)
```

3.3.2.3 Results and Interpretation

By analysing the diagram shown above, the simulation results for different parameter values coming from the EMEDS method for isotropic scattering and the MMEA method for non-isotropic are found. 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 between both. The first set of figures are for the computation of the angels of arrivals for the transmit side with $\alpha_v^T = 0^\circ$, $\beta_T = 90^\circ$, $\kappa = 0\, and\, 40$ and $f_{\max}^T = 91 Hz$.

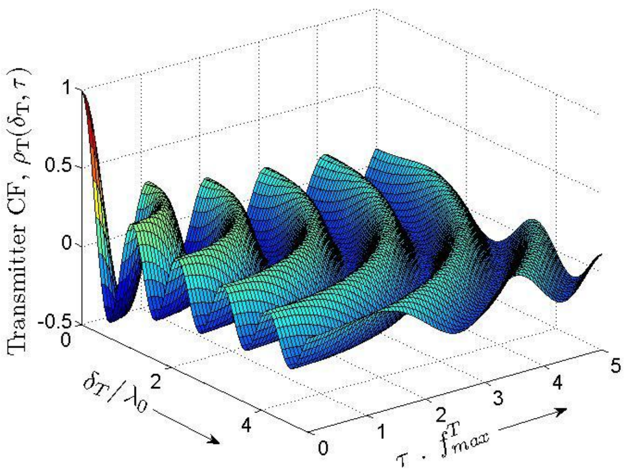


Figure 3.29: The transmitter CF $\rho_T\left(\delta_T, \tau\right)$ of the 2x2 MIMO mobile-to-mobile reference model (isotropic scattering)

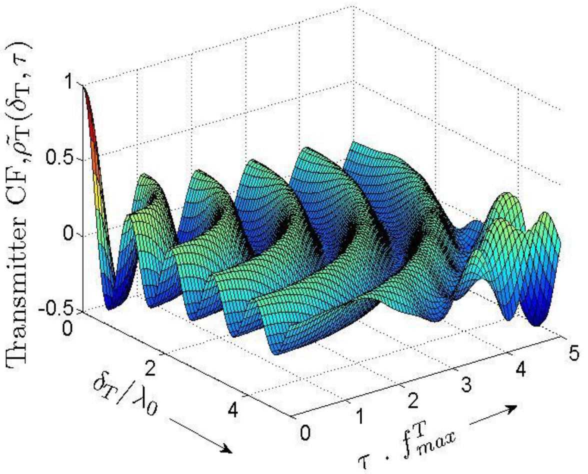


Figure 3.30: The transmitter CF $\tilde{\rho}_T\left(\delta_T, \tau\right)$ of the 2x2 MIMO mobile-to-mobile channel simulator based on EMEDS with M = 40 (isotropic scattering)

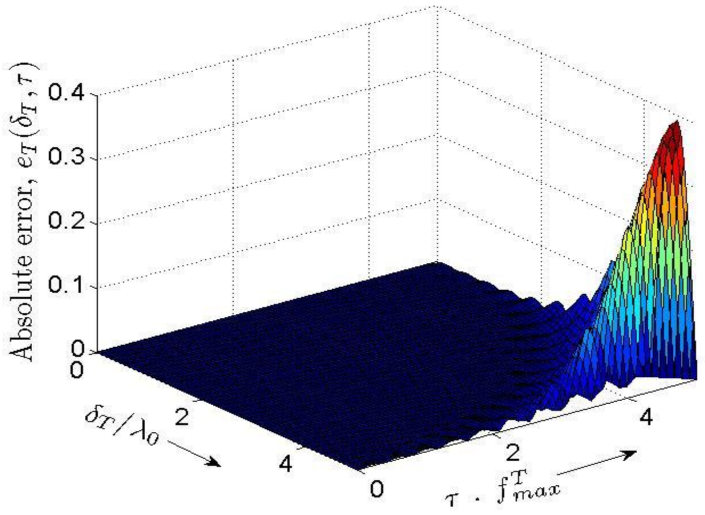


Figure 3.31: Absolute error | $e_T(\delta_T,\tau)$ | = | $\rho_T(\delta_T,\tau)$ - $\tilde{\rho}_T(\delta_T,\tau)$ | By using the EMEDS with M = 40 (isotropic scattering)

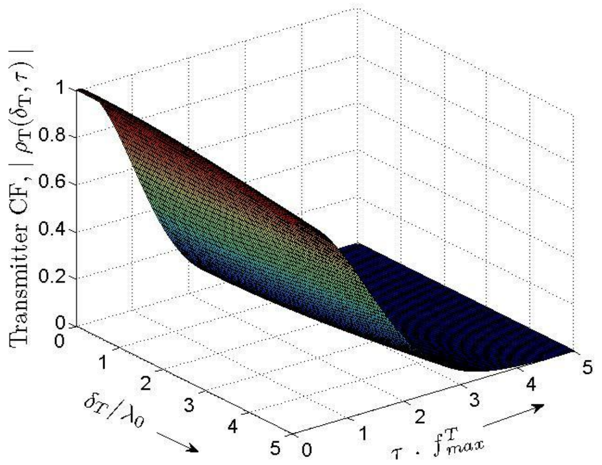


Figure 3.32: Absolute value of the transmitter CF $|\rho_T(\delta_T,\tau)|$ of the 2x2 MIMO mobile to mobile reference channel model under non-isotropic scattering condition (Von Mises density with $m_\alpha^T = 60$ and $\kappa_c = 40$).

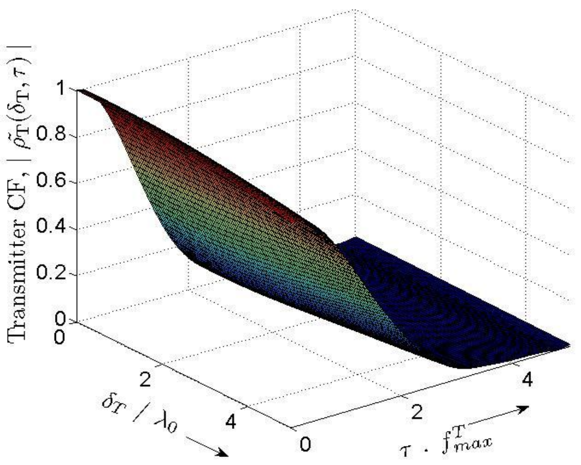


Figure 3.33: Absolute value of the transmitter CF $|\tilde{\rho}_T(\delta_T,\tau)|$ of the 2x2 MIMO mobile to mobile channel simulator design based on MMEA with $M = 50$ (non-isotropic scattering, Von Mises density with $m_\alpha^T = 60$ and $\kappa_c = 40$).

In this section how the reference model of the mobile to mobile MIMO fading channel can be derived and how its spatial/ temporal correlation properties can be analysed are studied. Here, scattering environment around the transmitter and receiver was modelled by invoking the geometrical two ring scattering model. Here we implemented the simulator by assuming that both terminals are moving. Then we learn how ergodic MIMO channel simulator can be derived from the reference model. After that investigate how its model parameter can be computed and then employed computed parameter in simulation model. All relative programs for the MIMO two-ring channel simulator are listed in appendix 3.

3.3.3 MIMO Channel Simulator Based On Elliptical Model

For the elliptical model for MIMO channels, we assume that all scatters 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 [2, 23] and [24].

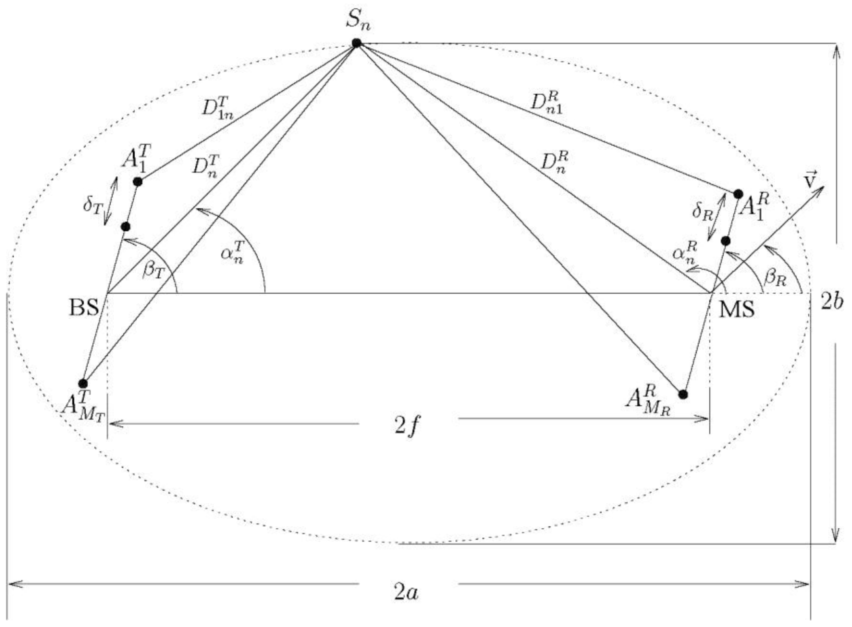


Figure 3.34 : Geometrical elliptical scattering model for an M_T x M_R MIMO channel with local scatters S_n placed on an ellipse

Where

δ_T	: Antenna element spacing at base station
δ_R	: Antenna element spacing at mobile station
β_T	: Multi-element antenna tilt angle at base station
β_R	: Multi-element antenna tilt angle at mobile station
α_v	: Angle of motion
α_n^T	: Angle of departure at base station
α_n^R	: Angle of arrival at mobile station
M_T	: Number of antenna elements at base station
M_R	: Number of antenna elements at mobile station

3.3.3.1 Reference Model

As shown in [2, 23], the channel gains of MIMO based on elliptical model is given by the following expression

$$h_{kl}(t) = \lim_{N\to\infty} \frac{1}{\sqrt{N}} \sum_{n=1}^{N} a_{\ln} b_{kn} e^{j(2\pi f_n t + \theta_n + \theta_0)}, \tag{3.64}$$

where

$$a_{l\,n} = e^{j\pi(M_T - 2l + 1)(\delta_T/\lambda_0)\cos(\alpha_n^T - \beta_T)}, \tag{3.65}$$

$$b_{kn} = e^{j\pi(M_R - 2k + 1)(\delta_R/\lambda_0)\cos(\alpha_n^R - \beta_R)}, \tag{3.66}$$

$$f_n = f_{\max}\cos(\alpha_n^R - \alpha_v) \tag{3.67}$$

$$\theta_0 = -\frac{4\pi a}{\lambda_0}. \tag{3.68}$$

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

$$\alpha_n^T = \begin{cases} f(\alpha_n^R) & \text{if} \quad 0 < \alpha_n^R \le \alpha_0, \\ f(\alpha_n^R) + \pi & \text{if} \quad \alpha_0 < \alpha_n^R \le 2\pi\text{-}\alpha_0, \\ f(\alpha_n^R) + 2\pi & \text{if} \quad 2\pi\text{-}\alpha_0 < \alpha_n^R \le 2\pi, \end{cases} \tag{3.69}$$

where

$$f(\alpha_n^R) = \arctan\left[\frac{(\kappa_0^2 - 1)\sin(\alpha_n^R)}{2\kappa_0 + (\kappa_0^2 + 1)\cos(\alpha_n^R)}\right] \tag{3.70}$$

and

$$\alpha_0 = \pi - \arctan\left(\frac{\kappa_0^2 - 1}{2\kappa_0}\right) \tag{3.71}$$

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 $p_{\alpha^R}(\alpha_R)$, which defines the scattering type, as the Von Mises distribution given in (3.27). Then, the 3D space-time CCF of the reference model demonstrated in [2, 23] is written as

$$\rho_{kl,k'l'}(\delta_T, \delta_R, \tau) = \int_{-\pi}^{\pi} a_{l'l}^2(\delta_T, \alpha^T) b_{kk'}^2(\delta_R, \alpha^R) e^{-j2\pi f(\alpha^R)\tau} p_{\alpha^R}(\alpha^R)\, d\alpha^R \tag{3.72}$$

where

$$a_{l'l}\left(\delta_T,\alpha^T\right)=e^{-j2\pi(l-l')\left(\delta_T/\lambda_0\right)\cos\left(\alpha^T-\beta_T\right)} \tag{3.73}$$

$$b_{kk'}\left(\delta_R,\alpha^R\right)=e^{-j2\pi(k-k')\left(\delta_R/\lambda_0\right)\cos\left(\alpha^R-\beta_R\right)}, \tag{3.74}$$

$$f\left(\alpha^R\right)=f_{\max}\cos\left(\alpha_R-\alpha_v\right) \tag{3.75}$$

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. The integral above can be solved analytically leading to $r_{h_{kl}}\left(\tau\right)=J_0\left(2\pi f_{\max}\tau\right)$.

$$r_{h_{kl}}\left(\tau\right)=\int_{-\pi}^{\pi}e^{-j2\pi f_{\max}\cos(\alpha^R-\alpha_v)\tau}\,p_{\alpha^R}\left(\alpha^R\right)d\alpha^R \tag{3.76}$$

3.3.3.2 Simulation Model

The simulation model is derived for the reference model by just reducing the number of scatters 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 $A_l^T\left(l=1,2,\ldots,M_T\right)$ to the receive antenna element $A_k^R\left(k=1,2,\ldots,M_R\right)$ as mentioned in [19] are

$$\hat{h}_{kl}\left(t\right)=\frac{1}{\sqrt{N}}\sum_{n=1}^{N}a_{1n}b_{kn}e^{j\left(2\pi f_n t+\theta_n\right)}, \tag{3.77}$$

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

$$\hat{\rho}_{kl,k'l'}\left(\delta_T,\delta_R,\tau\right)=\frac{1}{N}\sum_{n=1}^{N}a_{ll'n}^2\left(\delta_T\right)b_{kk'n}^2\left(\delta_R\right)e^{-j2\pi f_n\tau}, \tag{3.78}$$

where

$$a_{ll'n}=e^{-j2\pi(l-l')\left(\delta_T/\lambda_0\right)\cos\left(\alpha_n^T-\beta_T\right)} \tag{3.79}$$

$$b_{kk'n}=e^{-j2\pi(k-k')\left(\delta_R/\lambda_0\right)\cos\left(\alpha_n^R-\beta_R\right)} \tag{3.80}$$

The temporal ACF of the simulation model is given by

$$r_{h_{kl}}\left(\tau\right)=\frac{1}{N}\sum_{n=1}^{N}e^{-j2\pi f_{\max}\cos(\alpha_n^R-\alpha_v)\tau} \tag{3.81}$$

To compute the AoA used to generate the channel gains, two fundamental methods approved by their efficiency as described in the previous sections are used. The MMEA method is defined by solving (3.27). The LPNM method is supposed to optimize the two following errors.

$$E_1^{(p)} = \left\{ \frac{1}{\tau_{\max}} \int_0^{\tau_{\max}} \left| r_{h_{11}}(\tau) - \tilde{r}_{h_{11}}(\tau) \right|^p d\tau \right\}^{1/p} \tag{3.82}$$

$$E_2^{(p)} = \left\{ \frac{1}{\delta_T^{\max}\delta_R^{\max}} \int_0^{\delta_T^{\max}} \int_0^{\delta_R^{\max}} \left| \rho_{kl,k'l'}(\delta_T,\delta_R) - \tilde{\rho}_{kl,k'l'}(\delta_T,\delta_R) \right|^p d\delta_R d\delta_T \right\}^{1/p} \tag{3.83}$$

3.3.3.3 Implementation

According to the generalized concept of deterministic channel modelling, a stochastic simulation model is obtained from the reference model. 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 (3.69). For this purpose, we employ the von Mises density with parameter κ = 10 and m_α = 0 for non-isotropic scattering. κ was set to Zero for isotropic scattering.

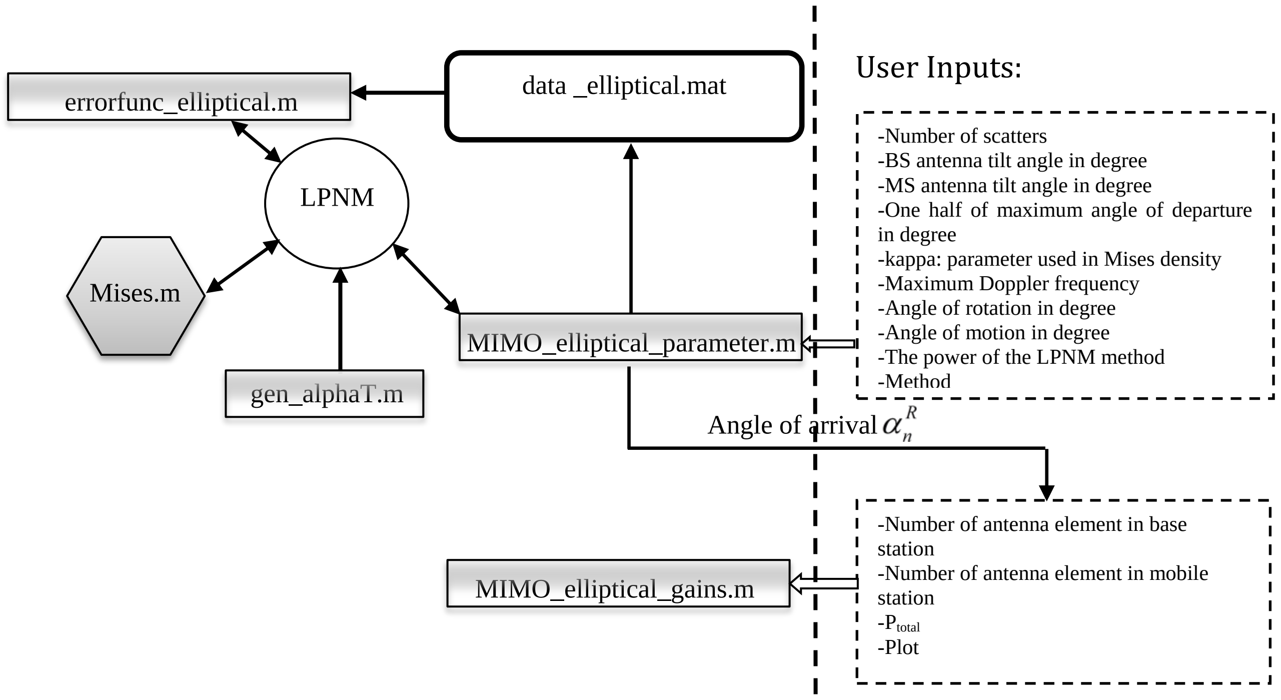

errorfunc_elliptical.m
data _elliptical.mat
User Inputs:
-Number of scatters
-BS antenna tilt angle in degree
-MS antenna tilt angle in degree
-One half of maximum angle of departure in degree
-kappa: parameter used in Mises density
-Maximum Doppler frequency
-Angle of rotation in degree
-Angle of motion in degree
-The power of the LPNM method
-Method
LPNM
Mises.m
gen_alphaT.m
MIMO_elliptical_parameter.m
Angle of arrival α_n^R
-Number of antenna element in base station
-Number of antenna element in mobile station
-P_{total}
-Plot
MIMO_elliptical_gains.m

Figure 3.35: Block diagram for the implementation of a MIMO channel simulator based on elliptical geometrical model

The Figure 3.35 illustrates the block diagram 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_alphaT.m which generates the angles of departure starting from the angles of arrival with using the LPNM method.
- 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.

Attention: Results were based on the given parameters for Matlab program.

Non-isotropic scattering (*κ* = 10):

```
[alphaR]=MIMO_elliptical_parameters(25,90,90,10,91,180,0,0,'lpnm',1)
```

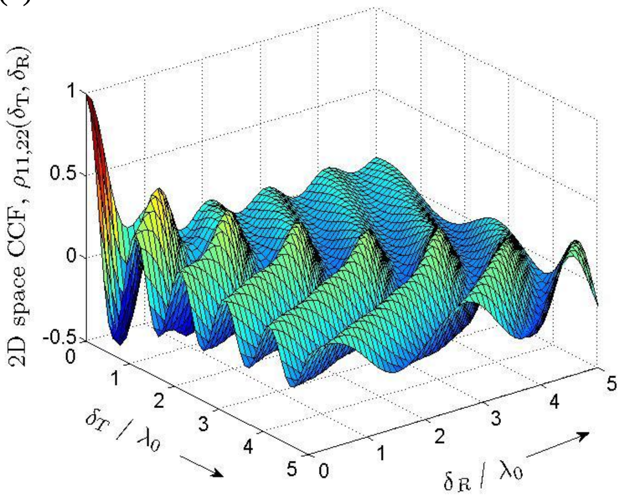
Isotropic scattering (*κ* = 0):

```
[alphaR]=MIMO_elliptical_parameters(25,90,90,0,91,180,0,0,'lpnm',1)
```

3.3.3.4 Results and Interpretation

In this section, some results for the 2D space CCF and time ACF under isotropic and non-isotropic conditions are presented. Reference model and simulation model are shown below for both isotropic and non-isotropic. For this purpose, the von Mises density with parameter κ and m_α. Parameter $\kappa = 10$ and $m_\alpha = 0$ were assigned for non-isotropic scattering.

(a)



(b)

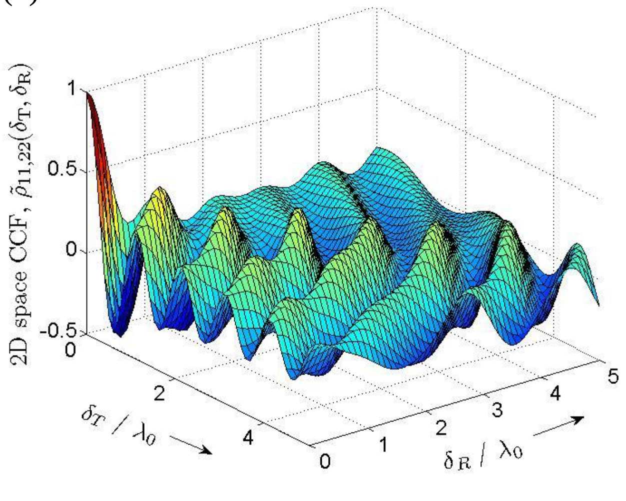


Figure 3.36: The 2D space CCF (a) $\rho_{11,22}(\delta_T,\delta_R)$ of the reference model (isotropic scattering) (b) $\tilde{\rho}_{11,22}(\delta_T,\delta_R)$ of the simulation model(isotropic) for $\kappa = 0$, $N = 25$.

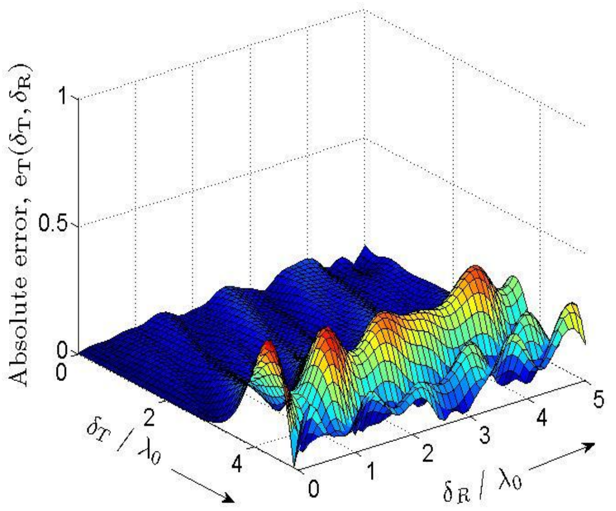


Figure 3.37: Absolute error of $| e_T (\delta_T,\tau) | = | \rho_{11,22}(\delta_T,\delta_R) - \tilde{\rho}_{11,22}(\delta_T, \delta_R) |$ By using the LPNM with (isotropic scattering) for $\kappa = 0$, $N = 25$.

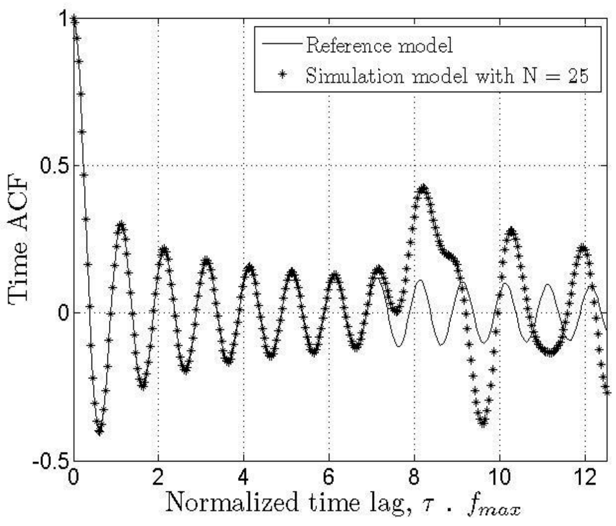


Figure 3.38: The time ACF $r_{h_{kl}}(\tau)$ of the reference model (isotropic scattering) for $\kappa = 0$, $N = 25$.

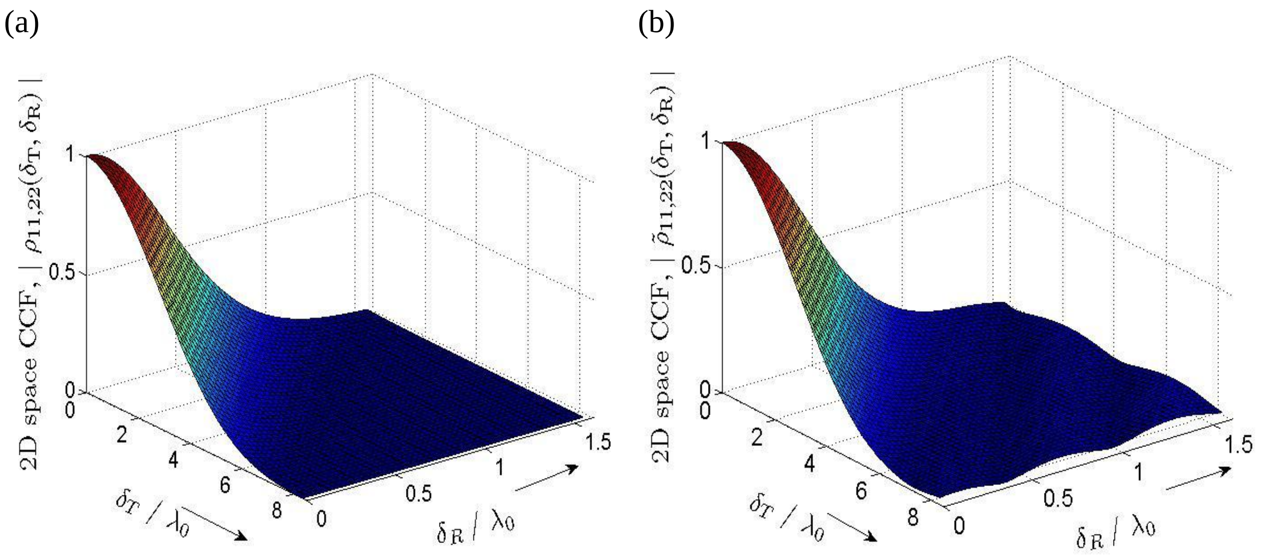


Figure 3.39: Absolute value of the 2D space CCF (non-isotropic scattering, von Mises density with parameter $\kappa = 10$ and $m_\alpha = 0$) (a) $| \rho_{11,22}(\delta_T, \delta_R) |$ of the reference model (b) $| \tilde{\rho}_{11,22}(\delta_T,\delta_R) |$ of the simulation model for $N = 25$.

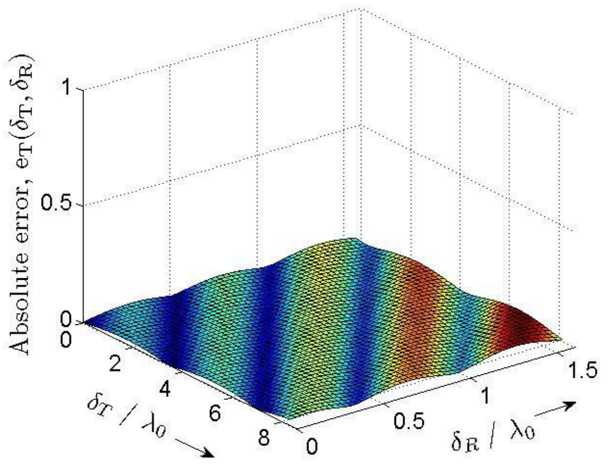


Figure 3.40: Absolute error of | e_T (δ_T,τ) | = | $\rho_{11,22}(\delta_T,\delta_R)$ - $\tilde{\rho}_{11,22}(\delta_T,\delta_R)$ | By using the LPNM (non- isotropic scattering)

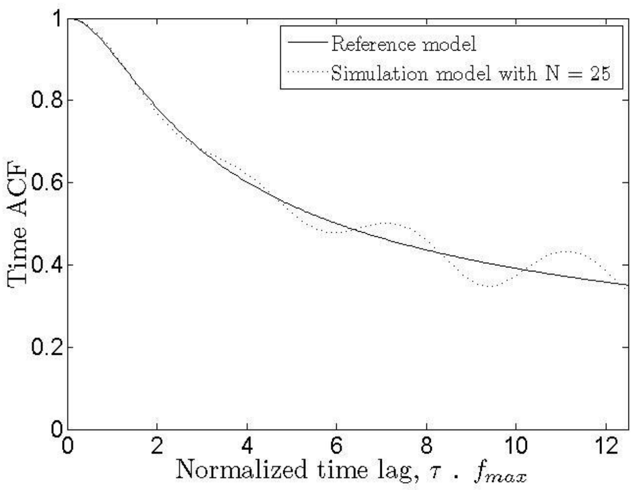


Figure 3.41: Absolute value of the ACF $|r_{h_{kl}}(\tau)|$ of the reference model and $|\tilde{r}_{h_{kl}}(\tau)|$ simulation model (non-isotropic scattering, Von Mises density with parameter κ = 10 and m_α = 0)

When we compare CCF of above figures it shows that best performance given in simulation results for non-isotropic scattering model compared to isotropic scattering model. Reason behind this better performance is LPNM methods given best fitting parameter values for non-isotropic scatter mode. But time autocorrelation behaviour is not showed good agreement with reference model. All relative programs for the MIMO elliptical model based simulator are listed in Appendix 3.

3.4 UNCORRELATED RAYLEIGH CHANELS

In this section, an improved sum-of-sinusoids simulation model is proposed for uncorrelated Rayleigh fading channels. The new model employs random initial phase, and conditional random Doppler frequency for all individual sinusoids. There are many different approaches to the modelling and simulation of mobile radio channels and the references therein. Among them, the well-known mathematical reference model of Clarke [8] and its simplified simulation model of Jakes [9] have been widely used for Rayleigh fading channels for about 30 years. However, Jakes' simulator is a deterministic model, and it has difficulty to create multiple uncorrelated fading waveforms for frequency selective fading channels and multiple-input multiple-output (MIMO) channels. Generation of a set of multiple uncorrelated Rayleigh fading waveforms is important for the development and analysis of diversity scheme and MIMO-OFDM channel with uncorrelated sub channel.

3.4.1 Problem Description

Concept behind the uncorrelated Rayleigh fading is generation of wave forms which having Rayleigh fading properties [25, 26]. These generated waveforms are uncorrelated to each other. We want to simulate *K* mutually uncorrelated Rayleigh fading waveforms as given in (3.84).

$$\tilde{\zeta}^{(k)}(t) = |\tilde{\mu}^{(k)}(t)| = |\tilde{\mu}_1^{(k)}(t) + j\tilde{\mu}_2^{(k)}(t)|, \qquad k = 1, 2, \ldots, K \qquad (3.84)$$

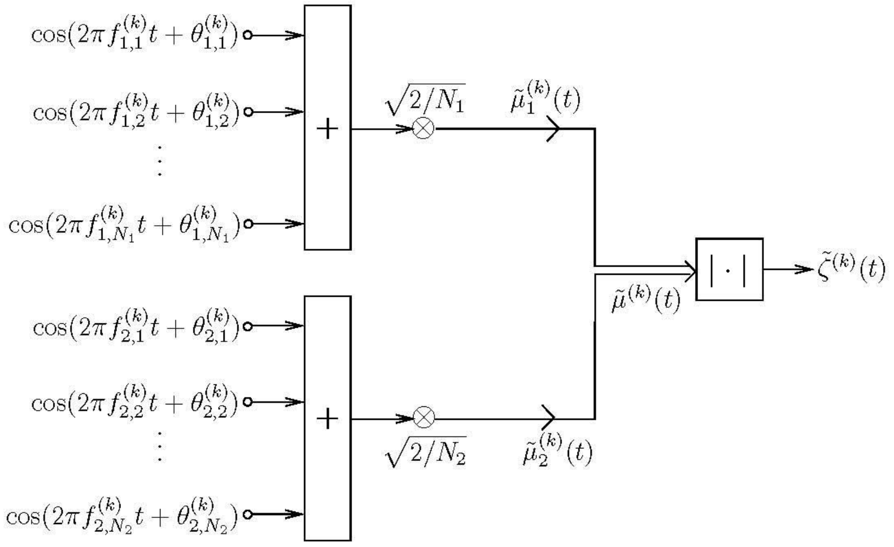

$\cos(2\pi f_{1,1}^{(k)}t + \theta_{1,1}^{(k)})$
$\cos(2\pi f_{1,2}^{(k)}t + \theta_{1,2}^{(k)})$
$\cos(2\pi f_{1,N_1}^{(k)}t + \theta_{1,N_1}^{(k)})$
+
$\sqrt{2/N_1}$
$\tilde{\mu}_1^{(k)}(t)$
$\cos(2\pi f_{2,1}^{(k)}t + \theta_{2,1}^{(k)})$
$\cos(2\pi f_{2,2}^{(k)}t + \theta_{2,2}^{(k)})$
$\cos(2\pi f_{2,N_2}^{(k)}t + \theta_{2,N_2}^{(k)})$
+
$\sqrt{2/N_2}$
$\tilde{\mu}_2^{(k)}(t)$
$\tilde{\mu}^{(k)}(t)$
$|\cdot|$
$\tilde{\zeta}^{(k)}(t)$

Figure: 3.42: Structure of the SOS channel simulator generating Rayleigh fading waveforms (*k = 1, 2,…, K*)

In this pace of study, SOC concept to develop the simulator is considered. The structure of the SOS channel simulator is shown in Figure 3.42. Developed SOS channel simulator generates the waveforms as given in equation (3.85).

$$\tilde{\mu}_i^{(k)}(t) = \sqrt{\frac{2}{N_i}} \sum_{n=1}^{N_i} \cos\left(2\pi f_{i,n}^{(k)} t + \theta_{i,n}^{(k)}\right), \qquad i = 1, 2. \qquad (3.85)$$

Where N_i denotes the number of sinusoids, $f_{i,n}^{(k)}$ is called the discrete Doppler frequency, and $\theta_{i,n}^{(k)}$ is the phase of the nth sinusoid of the inphase component $\tilde{\mu}_1^{(k)}(t)$or quadrature component $\tilde{\mu}_2^{(k)}(t)$. The phases $\theta_{i,n}^{(k)}$are considered as outcomes of independent and identically distributed (i. i. d.) random variable, each having a uniform distribution over the interval $(0, 2\pi\ ]$.

3.4.2 Generalized Method of Exact Doppler Spread (GMEDS$_q$)

We consider the generalized MEDS (GMEDS$_q$) [2, 20] which represents a class of parameter computation methods for the design of SOS Rayleigh fading channel simulators. According to this method, the discrete Doppler frequencies $f_{i,n}^{(k)}$ are given by:

$$f_{i,n}^{(k)} = f_{\max}\cos(\alpha_{i,n}^{(k)}) \tag{3.86}$$

$$f_{i,n}^{(k)} = f_{\max}\cos\left[\frac{q\pi}{2N_i}\left(n-\frac{1}{2}\right)+\alpha_{i,0}^{(k)}\right] \tag{3.87}$$

Where $\alpha_{i,0}^{(k)}$ is called the angle-of-rotation that will be defined subsequently and $q\in\{0,1,...,4\}$.
Note that the parameter q mainly determines the range of values for quantities $\alpha_{i,n}^{(k)}$.
With this simulation we will use q = 1 and the angel-of-rotation $\alpha_{i,0}^{(k)}$, GMEDS$_q$ reduced to the original MED with q = 1 and $\alpha_{i,0}^{(k)} = 0$.

By using the GMEDS$_1$ method can generate any number K of multiple uncorrelated Rayleigh fading waveforms without increasing the model complexity, the angle-of-rotation $\alpha_{i,0}^{(k)}$ defined as:

$$\alpha_{i,0}^{(k)} = (-1)^{i-1}\frac{\pi}{4N_i}.\frac{k}{K+2} \tag{3.88}$$

Where i = 1, 2 and k = 1, 2, …, K.
The autocorrelation function $\tilde{r}_{\mu_i\mu_i}^{(k)}(\tau)$ of the k^{th} waveform $\tilde{\mu}_i^{(k)}(t)$ can be expressed as:

$$\tilde{r}_{\mu_i\mu_i}^{(k)}(\tau) = \lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T}\tilde{\mu}_i^{(k)}(t)\tilde{\mu}_i^{(k)}(t+\tau)\,dt \tag{3.89}$$

$$\tilde{r}_{\mu_i\mu_i}^{(k)}(\tau) = \frac{1}{N_i}\sum_{n=1}^{N_i}\cos(2\pi f_{i,n}^{(k)}\tau) \tag{3.90}$$

For i = 1, 2 and k = 1, 2,…, K. Analogously, the time-average cross-correlation function $\tilde{r}_{\mu_i\mu_\lambda}^{(k,l)}(\tau)$ of $\tilde{\mu}_i^{(k)}(t)$ and $\tilde{\mu}_\lambda^{(l)}(t)$ can obtained as:

$$\tilde{r}^{(k,l)}_{\mu_i\mu_\lambda}(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} \tilde{\mu}^{(k)}_i(t)\tilde{\mu}^{(l)}_\lambda(t+\tau)\,dt \tag{3.91}$$

`

$$= \begin{cases} \dfrac{1}{N_i N_\lambda} \displaystyle\sum_{n=1}^{N_i}\sum_{m=1}^{N_\lambda} \cos(2\pi f^{(k)}_{i,n}\tau - \theta^{(k)}_{i,n} \pm \theta^{(l)}_{\lambda,m}) \text{ if } f^{(k)}_{i,n} = \pm f^{(l)}_{\lambda,m}, \\ 0 \qquad\qquad f^{(k)}_{i,n} \neq \pm f^{(l)}_{\lambda,m}, \end{cases} \tag{3.92}$$

The error function for the angle-of-rotation $E_2\left(\alpha^{(k)}_{i,0}\right)$ and we can use the equation *E₂* for L$_p$-norm

$$E_2 = \left[\frac{1}{\tau_{\max}} \int_0^{\tau_{\max}} |\, \tilde{r}^{(k)}_{\mu_i\mu_i}(\tau) - r^{(k)}_{\mu_i\mu_i}(\tau)\,|^2 \right]^{1/2} \tag{3.93}$$

The cross-correlation functions $\tilde{r}^{(k,l)}_{\mu_i\mu_\lambda}(\tau)$ of $\tilde{\mu}^{(k)}_i$ and $\tilde{\mu}^{(l)}_\lambda$ must be equal to zero.

We can express the autocorrelation function $\tilde{r}^{(k)}_{\mu\mu}(\tau)$ of the k^{th} complex waveform $\tilde{\mu}^{(k)}(\tau) = \tilde{\mu}^{(k)}_1(t) + j\tilde{\mu}^{(k)}_2(t)$ as

$$\tilde{r}^{(k)}_{\mu\mu}(\tau) = \tilde{r}^{(k)}_{\mu_1\mu_1}(\tau) + \tilde{r}^{(k)}_{\mu_2\mu_2}(\tau) \tag{3.94}$$

The reference model to Rayleigh model under two-dimensional isotropic scattering condition with $\sigma_0^2 = 1$ is

$$r^{(k)}_{\mu_i\mu_i}(\tau) = J_0(2\pi f_{\max}\tau), \quad i = 1, 2 \tag{3.95}$$

Where $J_0(.)$ denotes the zeroth-order Bessel function of the first kind. In this case the autocorrelation function of the complex process $\mu^{(k)}(\tau) = \mu_1(t) + j\mu_2(t)$ equal $r_{\mu\mu}(\tau) = 2r_{\mu_i\mu_i}(\tau) = 2J_0(2\pi f_{\max}\tau),$ since the cross-correlation function $r_{\mu_1\mu_2}(\tau)$ and $r_{\mu_2\mu_1}(\tau)$ are zero.

3.4.3 Implementation

- The program errorfunE2.m shows the effective of changing the angle-of-rotation with using the GMEDS$_1$ for different value for N$_i$. In this case we use N_i = 20, 25, 30.
- Program file ACF_Uncorr.m computes the autocorrelation function $r_{\mu_i\mu_i}(\tau)$ of the reference model in comparison to autocorrelation function $\tilde{r}_{\mu_i\mu_i}(\tau)$ of simulation model by using the MEDS with $\alpha_{i,0}^{(k)} = 0$ and the GMEDS$_1$ with $\alpha_{i,0}^{(k)} = \pm\pi/120$, the number of sinusoid N_i = 20, and $k = K$= 4.
- Program file ACF_GAU_Uncorr.m computes the autocorrelation function $r_{\mu\mu}(\tau)$ of the complex Gaussian process of the reference model in comparison with corresponding autocorrelation function $\tilde{r}_{\mu\mu}(\tau)$ of simulated model designed by using the MEDS with $\alpha_{i,0}^{(k)} = 0$ and the GMEDS$_1$ with $\alpha_{i,0}^{(k)} = \pm\pi/120$ and $\alpha_{i,0}^{(k)} = \pm\pi/480$, the number of sinusoid $N_1 = N_2 = N$ = 20 with K = 4.
- Program f_uncorr.m shows the simulation of uncorrelated Rayleigh fading waveform for several sample function which is selected K = 4, and N_1 = N_2 = 20, the phases $\theta_{i,n}^{(k)}$ are considered as outcomes of independent and identically distributed (i.i.d) random variable each having a uniform distribution over the interval (0,2π].
  The discrete Doppler frequencies $f_{i,n}^{(k)}$ is defined by using GMEDS$_1$.

Attention: Results were based on the given parameters for Matlab programs.

```
errorfunE2([20;25;30],91,1,2,1)

ACF_Uncorr(20,0,pi/120,-pi/120)

ACF_GAU_Uncorr(20,0,pi/160,pi/480)

f_uncorr.m(20,20,4,91,1)
```

3.4.4 Results and Interpretation

Figure 3.43 shows the mean square error for limited number of N_i. It can see that when the numbers of sinusoids are increased error will come down and similarly angle of rotation have considerable impact on the mean square error. It shows angle of rotation $\alpha_{i,0}^{(k)}$ at 0 and $\pm\pi/2$ is given zero error. Figure 3.44 shows the worst approximation results of the autocorrelation function of $\tilde{r}_{\mu\mu}^{(k)}(\tau)$ for the GMEDS$_1$ when $\alpha_{i,0}^{(k)} = \pm\pi/120$. It shows the better results for GMEDS$_1$ compared with parameter computation method of the MEDS. Figure 3.45 plots of the autocorrelation function $\tilde{r}_{\mu\mu}^{(k)}(\tau)$ for four selected values of $\alpha_{i,0}^{(k)}$ computed by GMEDS$_1$ with N_i = 20 and K = 4. It can be noticed that the performance of the MEDS is worse than the performance of the GMEDS$_1$.

Generated uncorrelated waveforms are shown in Figure 3.46 under GMEDS$_1$ method. It illustrate temporal behaviour of four uncorrelated fading envelops for $K = 4$.

Finally it can be concluded that advantage of the GMEDS$_1$ over the original MEDS is that the GMEDS$_1$ can be used to design a very large number of mutual uncorrelated Rayleigh fading waveform by keeping the number of sinusoids constant. Complexity of the GMEDS$_1$ method is low although the numbers of mutual uncorrelated Rayleigh fading waveform increase.

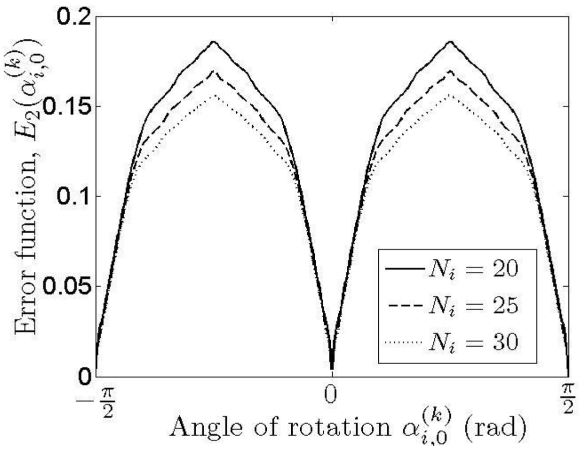


Figure 3.43: The error function $E_2 = E_2\left(\alpha_{i,0}^{(k)}\right)$ in terms of the angle-of rotation $\alpha_{i,0}^{(k)}$ by using the GMEDS$_1$ for various values of N_i

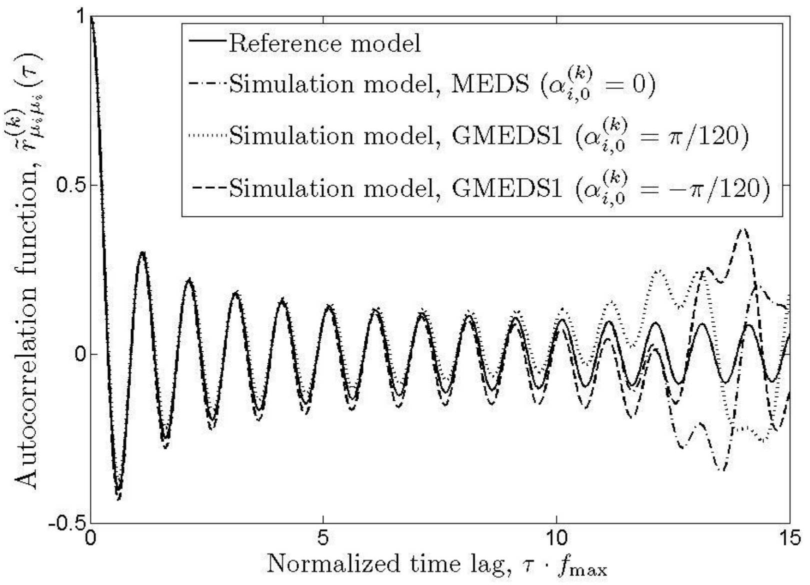


Figure 3.44: The autocorrelation function $r_{\mu_i\mu_i}(\tau)$ of the reference model in comparison to the autocorrelation function $\tilde{r}_{\mu_i\mu_i}(\tau)$ of the simulation model by using the MEDS with $\alpha_{i,0}^{(k)} = 0$ and the GMEDS$_1$ with $\alpha_{i,0}^{(k)} = \pm\pi/120$ ($N_i = 20$, $k = K = 4$)

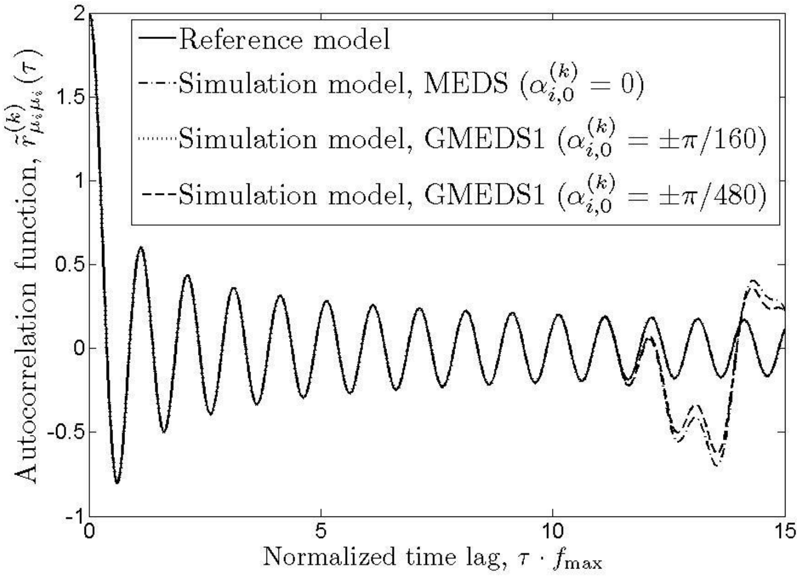


Figure3.45: The autocorrelation function $r_{\mu_i\mu_i}(\tau)$ of the complex Gaussian process of the reference model in comparison with the corresponding autocorrelation function $\tilde{r}_{\mu_i\mu_i}(\tau)$ of the simulation model designed by using the MEDS with $\alpha_{i,0}^{(k)} = \pm\pi/160$ and $\alpha_{i,0}^{(k)} = \pm\pi/480$ ($N_1 = N_2 = 20$, $K = 4$)

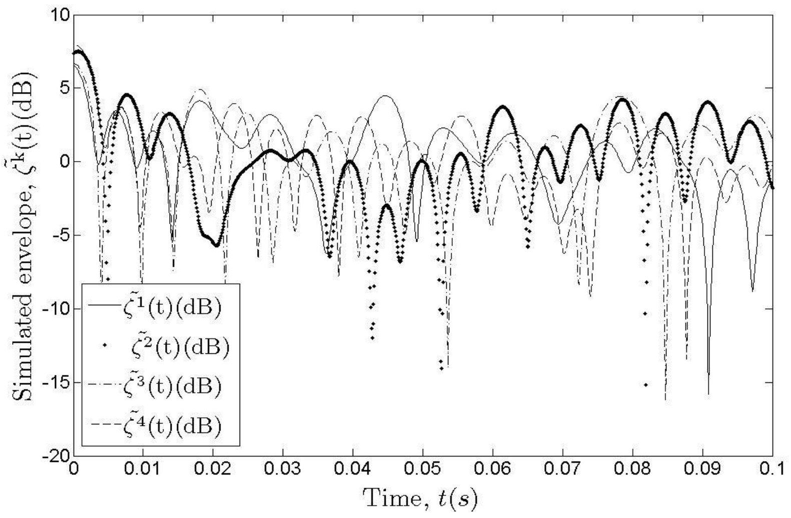


Figure 3.46: Simulated uncorrelated Rayleigh fading waveform by using GMEDS$_1$ (f_{max}= 91 Hz, N_1=N_2 =*20, K*= 4)

3.5 FREQUENCY HOPPING MOBILE RADIO CHANNELS

Frequency hopping (FH) techniques is used in the Global System for Mobile (GSM) communication system. Frequency hopping combined with error protection techniques and interleaving is a very effective means in combating fading in mobile radio communication. Principle of frequency hopping in GSM is that the carrier frequency changes with every time division multiple accesses.
A channel simulator that models accurately the physical channel statistics determined by the frequency separation between two carriers in FH systems is called a FH channel simulator.
Such a simulator is important for the design, performance evaluation, optimization, and test of modern FH wireless communication systems.

3.5.1 Reference Model

Reference model was derived for frequency hopping mobile radio channel in the complex base band and for the simplicity it restricts only for frequency-non-selective channel and assumed that no line-of-sight component is present. Under this condition, the sum of all received scattered and reflected component subjected to two different carrier frequencies, denoted by f_0 and f_0'.This one modelled by using following complex valued Gaussian random process [1, 2]. Here $2\pi\tau_n'\chi$ is phase rotation of the nth scattered component. Here χ is frequency separation variable.

$$\mu(t) = \lim_{N\to\infty} \sum_{n=1}^{N} c_n \; e^{j(2\pi f_n t+\theta_n)} \text{, at } f_0 \tag{3.96}$$

$$\mu'(t) = \lim_{N\to\infty} \sum_{n=1}^{N} c_n \; e^{j(2\pi f_n t+\theta_n-2\pi\tau_n'\chi)} \text{, at } f_0' \tag{3.97}$$

$$\chi = f_0 - f_0' \tag{3.98}$$

The reference model for the received envelop at two different carrier frequencies is given by

$$\varsigma(t) = |\mu(t)| = |\mu_1(t) + j\,\mu_2(t)| \text{, at } f_0 \tag{3.99}$$

$$\varsigma'(t) = |\mu'(t)| = |\mu_1'(t) + j\mu_2'(t)| \text{, at } f_0' \tag{3.100}$$

Equation (3.81) and (3.82) describe autocorrelation and cross correlation functions:

$$\gamma_{\mu_i\mu_i}(\tau) \coloneqq E\{\mu_i(t)\mu_\lambda(t+\tau)\} \tag{3.101}$$

$$\gamma_{\mu_i\mu'_i}(\tau,\chi) \coloneqq E\{\mu_i(t)\mu'_\lambda(t+\tau)\} \tag{3.102}$$

3.5.2 Simulation Model

Frequency hopping channel simulation model is based on the concept of deterministic channel modelling with sum-of sinusoids [2, 27]. Here we use zero-mean real-valued Gaussian random process μ (t) and μ'(t) by the following superposition of sinusoids. μ (t) and μ'(t) are deterministic because model parameters are constant quantities.

$$\tilde{\mu}_i(t) = \sum_{n=1}^{N} c_{i,n} \cos\left(2\pi f_{i,n} t + \theta_{i,n}\right) \text{, at } f_0 \tag{3. 103}$$

$$\tilde{\mu}_i' \text{ (t)} = \sum_{n=1}^{N} c_{i,n} \cos\left(2\pi f_{i,n} t + \theta_{i,n}'\right) \text{ , at } f_0' \tag{3. 104}$$

$$c_{i,n} = \sigma_0 \sqrt{\frac{2}{N_i}} , \tag{3. 105}$$

$$f_{i,n} = f_{max} \, sin\left[\frac{\pi}{2N_i}(n - \frac{1}{2})\right] , \tag{3. 106}$$

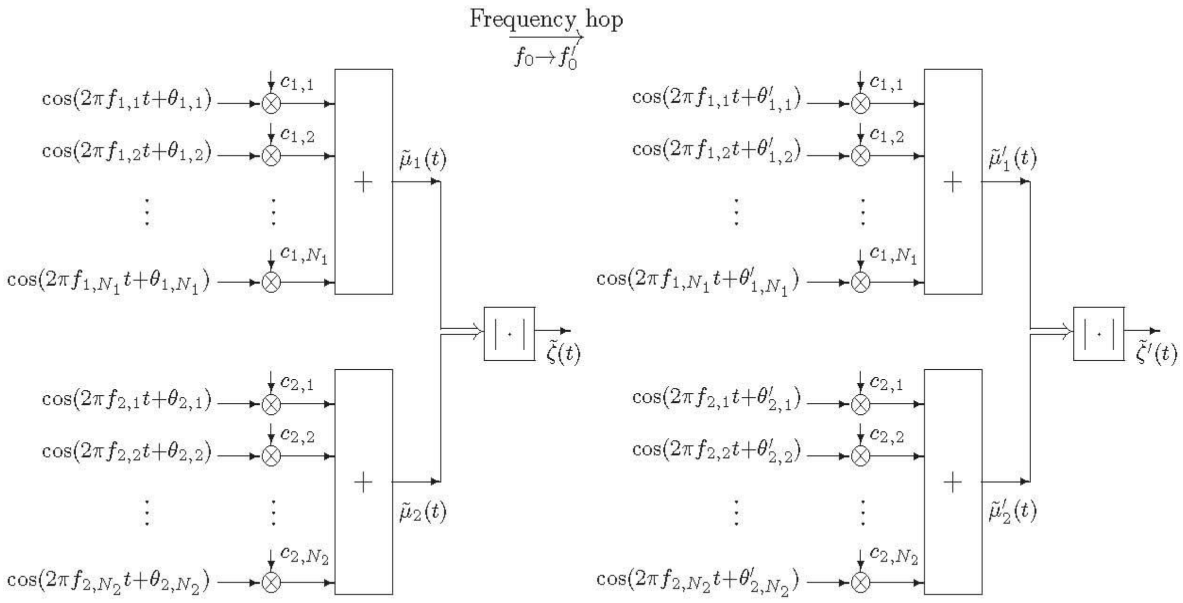

Frequency hop
$f_0 \rightarrow f_0'$
$\cos(2\pi f_{1,1}t+\theta_{1,1})$
$\cos(2\pi f_{1,2}t+\theta_{1,2})$
$\cos(2\pi f_{1,N_1}t+\theta_{1,N_1})$
$c_{1,1}$
$c_{1,2}$
c_{1,N_1}
$\tilde{\mu}_1(t)$
$\cos(2\pi f_{2,1}t+\theta_{2,1})$
$\cos(2\pi f_{2,2}t+\theta_{2,2})$
$\cos(2\pi f_{2,N_2}t+\theta_{2,N_2})$
$c_{2,1}$
$c_{2,2}$
c_{2,N_2}
$\tilde{\mu}_2(t)$
$|\cdot|$
$\tilde{\zeta}(t)$
$\cos(2\pi f_{1,1}t+\theta'_{1,1})$
$\cos(2\pi f_{1,2}t+\theta'_{1,2})$
$\cos(2\pi f_{1,N_1}t+\theta'_{1,N_1})$
$\tilde{\mu}'_1(t)$
$\cos(2\pi f_{2,1}t+\theta'_{2,1})$
$\cos(2\pi f_{2,2}t+\theta'_{2,2})$
$\cos(2\pi f_{2,N_2}t+\theta'_{2,N_2})$
$\tilde{\mu}'_2(t)$
$\tilde{\zeta}'(t)$

Figure 3.47: Frequency hopping Rayleigh fading channel simulator.

For *n=1, 2,..., N*$_i$ and *i =1, 2.* In order to assure that $\tilde{\mu}_i$ (t) and $\tilde{\mu}_i'$ (t) uncorrelated, we imposed on the number of sinusoids N$_1$ and N$_2$ that they are related by $N_2 := N_1 + 1$. $\chi = f_0' - f_0$. Here $\theta_{i,n}$ *and* $\theta_{i,n}'$ calculated based on;

$$\theta_{i,n} = 2\pi f_0 \varphi_{i,n} \; , \tag{3. 107}$$

$$\theta_{i,n}' = 2\pi(f_0 + \chi)\varphi_{i,n} \; , \tag{3. 108}$$

$$\varphi_{i,n} = a \ln\left(\frac{1}{1-\frac{n-\frac{1}{2}}{N_i}}\right) \; , \tag{3. 109}$$

Consequently, if a frequency hops f_0 and f_0' occurs, then envelop of the channel simulator jumps without any transient behaviour.

$$\tilde{\varsigma}\,(t) = \left|\tilde{\mu}_{\,1}(t) + j\,\tilde{\mu}_{\,2}(t)\right| \longrightarrow \tilde{\varsigma}'(t) = \left|\tilde{\mu}'_{\,1}(t) + j\,\tilde{\mu}'_{\,2}(t)\right| , \tag{3.110}$$

Due to deterministic nature of $\tilde{\mu}_i$ (t) and $\tilde{\mu}_i'$ (t) we can investigate cross correlation properties by using time average and finally we can obtain

$$\tilde{\gamma}_{\mu_i\mu_i}(\tau) \; = \sum_{n=1}^{N_i} \frac{c_{i,n}^2}{2} \cos(2\pi f_{i,n}\tau)\} \tag{3.111}$$

$$\tilde{\gamma}_{\mu_1\mu_2}(\tau) = \tilde{\gamma}_{\mu_1{}'\mu_2{}'}(\tau) = 0, \tag{3.112}$$

$$\tilde{\gamma}_{\mu_i\mu'_i}(\tau,\chi) = \sum_{n=1}^{N_i} \frac{c_{i,n}^2}{2} \cos(2\pi f_{i,n}\tau + 2\pi\,\varphi_{i,n}\,\chi) \tag{3.113}$$

$$\tilde{\gamma}_{\mu_1\mu'_2}(\tau,\chi) = \tilde{\gamma}_{\mu_2\mu'_1}(\tau,\chi) = 0\,, \qquad \text{for } i = 1, 2. \tag{3.114}$$

Here N_1 and N_2 select as $N_2 = N_1 +1$. Therefore it follows from (3.115) that $f_{1,n} \neq f_{2,m}$ for all $n = 1, 2, \ldots, N_1$ and $m = 1, 2, \ldots, N_2$. $\chi = f_0' - f_0$.

NB: for more detail refer [2] 10.3.2

3.5.3 Implementation

The implementation of the frequency-hopping channel simulator was executed by applying Raleigh channel concept with sum of sinusoids. In Figure (3.49a) shows Simulation of the received envelops $\tilde{\varsigma}\ (t)$ under GSM system with Rural Area profile which describes in CEPT-COST 207 [18], where the propagation delays are negative exponential distributed with parameter $a = 0.1086\mu s$. By selecting carrier frequency in GSM system without frequency hopping with f_0 = 941.2 MHz the maximum Doppler frequency is 91 Hz. The number of sinusoids N_1, N_2 was fixed with $N_2 = N_1 + 1$, and in this figure $N_1 = 20$, and $N_2 = 21$. The variance was selected σ_0^2 and applying the principles of slow frequency hopping $f_0 \rightarrow f_0'$and the carrier frequency is change every TDMA frame duration of $4.615\mu s$. The change of $f_0 \rightarrow f_0'$ cause changing in phase $\theta \rightarrow \theta'$ and it is shown in Figure (3.49b).

- The function FH_channel_simulator.m is used to compute the simulated envelope $\hat{\zeta}(t)$ for both with and without frequency hopping to show its effect on the channel envelope $\hat{\zeta}(t)$.
- The function FH_ch_model_test.m is used to test the behaviour of channel with applying frequency hopping and compare between reference model and simulation model by using ACF and CCF, also show the effective of different values of $\chi = f_0{}' - f_0$.
- The option PLOT is provided to sketch the performance criteria, such as temporal ACF and space-time CCF as shown in Figure 3.50 to 3.53.

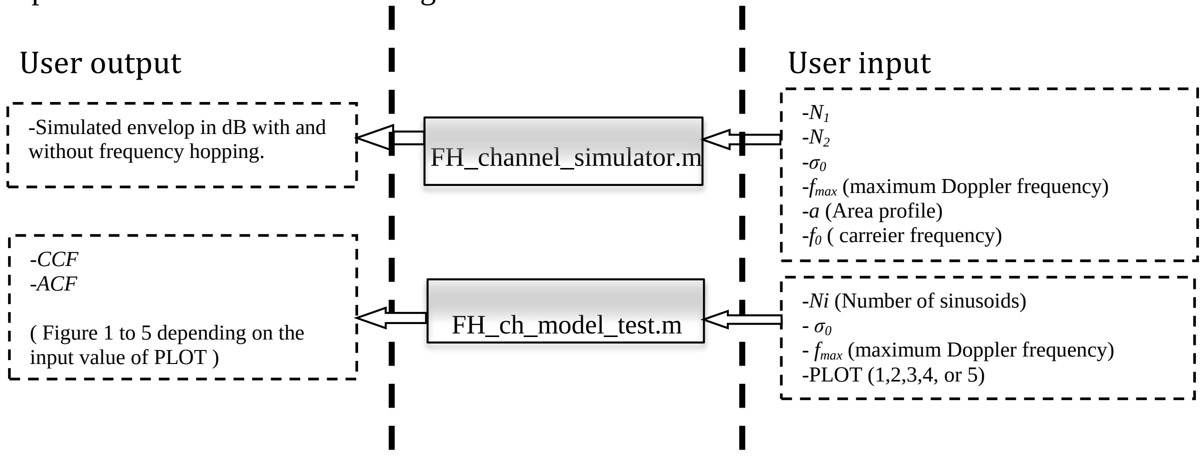

User output
User input
-Simulated envelop in dB with and without frequency hopping.
FH_channel_simulator.m
-N_1
-N_2
-σ_0
-f_{max} (maximum Doppler frequency)
-a (Area profile)
-f_0 (carreier frequency)
-CCF
-ACF
(Figure 1 to 5 depending on the input value of PLOT)
FH_ch_model_test.m
-Ni (Number of sinusoids)
- σ_0
- f_{max} (maximum Doppler frequency)
-PLOT (1,2,3,4, or 5)

Figure 3.48: The diagram for the implementation of a Frequency hopping channel simulator.

3.5.3 Results and Interpretation

Simulation was based on rural area profile specified for GSM system by CEPT-COST 207 [18]. Simulation of the received envelop for the rural area environment is shown in Figure 3.49(a) without frequency hopping. Frequency hopping in GSM is changed with every TDMA frame of duration 4.615ms. The resulting hopped output signal of the envelope is shown in 3.49(b). Figure 3.51 and 3.52 illustrate cross correlation function obtained for COST207 rural area for reference model and Simulation model. Parameter computation was done based on MEDS and parameters were defined in 3.105 and 3.106.

Attention: Results were based on the given parameters for Matlab program.

```
FH_channel_simulator(20,21,1,91,0.1086,941.2)
```

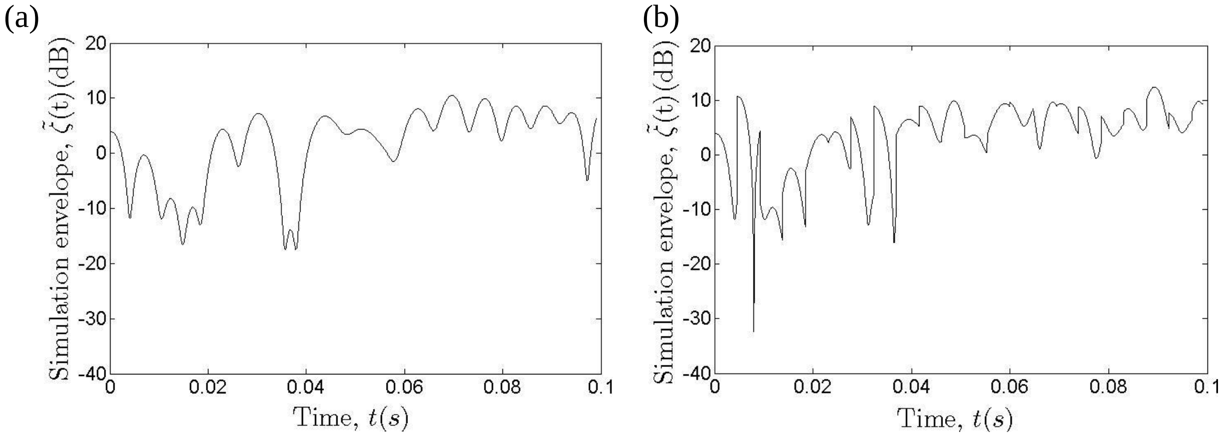
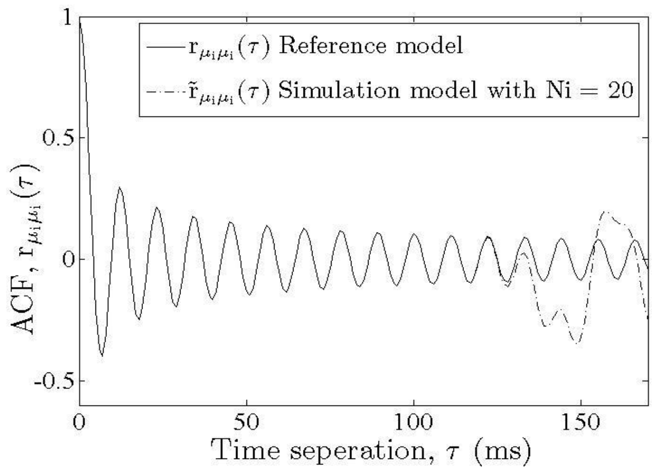


Figure 3.49: Simulation of the received envelops $\tilde{\varsigma}\ (t)$ for the rural area environment: (a) Without frequency hopping (b) with frequency hopping.



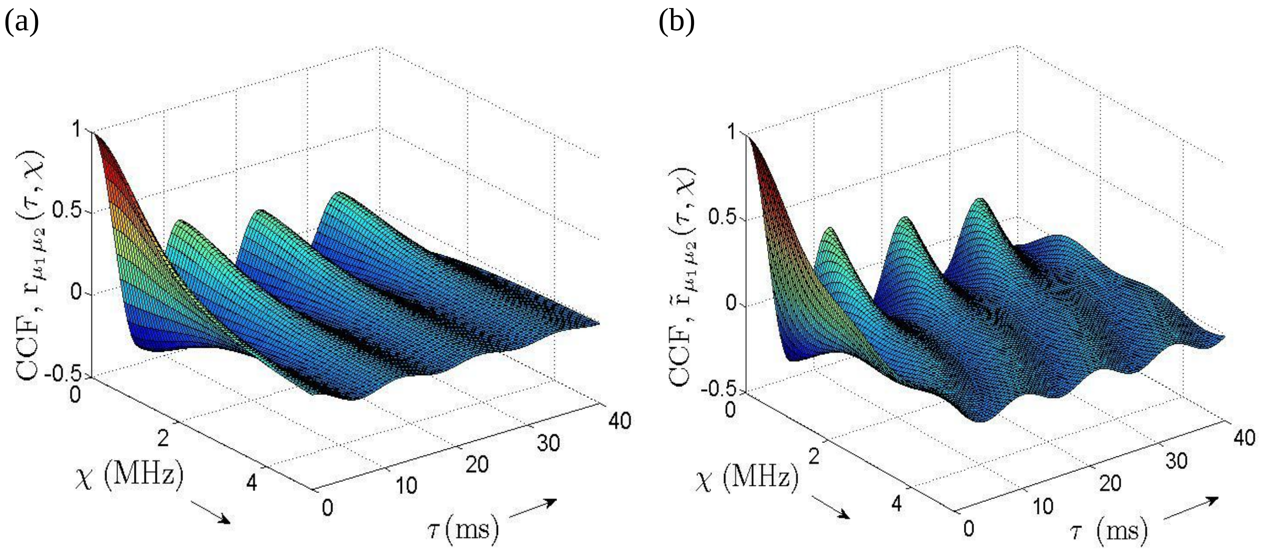
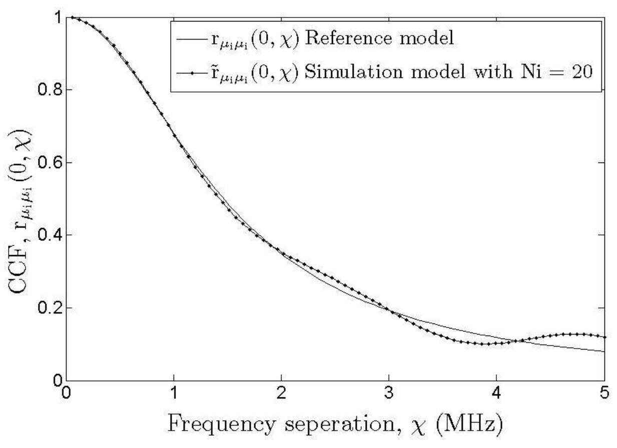


Figure 3.51 : Cross correlation function obtained for COST207 rural area profile (a = 0.1086μs, f_{max} = 91 Hz, σ_0^2 = 1) (a) Reference model (b) Simulation model for N_i = 20



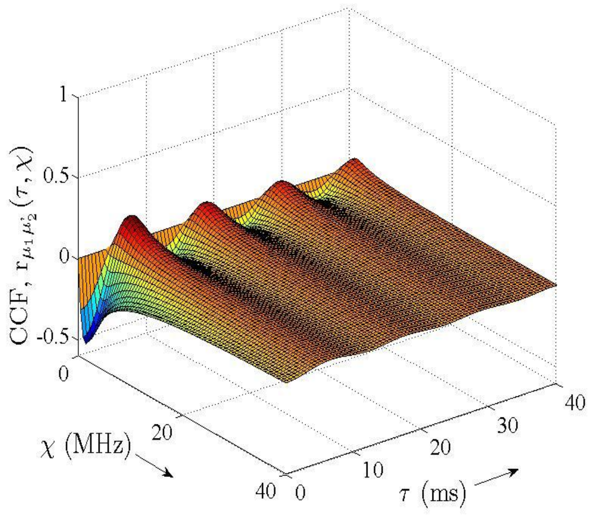


Figure 3.52: Cross-correlation function of reference model comparison with the simulated model (N_i = 20) for COS207 Rural area profile (a = 0.1086μs, σ_0^2=1)

Figure 3.53: Cross-correlation function of reference model for COST 207 Rural area profile (a = 0.1086μs, f_{max} = 91 Hz, σ_0^2=1)

It can be seen that the limited number of sinusoids N_i gives excellent approximation result for $\gamma_{\mu_i\mu_i}(\tau) \approx \tilde{\gamma}_{\mu_i\mu_i}(\tau)$ (ACF reference model and simulation model)) in the interval $0 \leq \tau \leq N_i/\ (2f_{max})$ as show in Figure 3.50 for N = 20. Figure 3.52 shows close relation of the cross-correlation function of simulation model ($\tilde{\gamma}_{\mu_i\mu'_i}(0,\chi)$) and reference model ($\gamma_{\mu_i\mu'_i}(0,\chi)$) for $N_i = 20$. All relative programs for the frequency-hopping channel simulator are listed in Appendix 3.

3.6 CONCLUSION OF THE CHANNEL MODELS

In this chapter, different Mobile fading channels have been treated. Initially, non-stationary land mobile satellite channel by using extended Suzuki process of Type I is implemented. These model parameters were found based on the measured signal. Based on the measured signal system was developed. After that we compared statistical properties of simulated signal with the measured signal. We found that the second order statistical properties were matched with simulated signal and the measured signal. That implies simulated model is best fitted with the real behaviour in the practical situation.

Secondly, the spatial shadowing process and its allied characteristics are described. In this MEA and LPNM methods for parameter computation are used. 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.

Thirdly, the MIMO channels (one-ring geometrical model, two-ring geometrical model and elliptical geometrical model) are introduced. For each model, we presented shortly the fundamental concept and the main features (channel gains, temporal ACF, 2D space-time CCF, impulse response, etc.). Once again, to evaluate the performance of the simulator, we made use of the stochastic properties mentioned earlier.

Forth section dedicates for implementation of uncorrelated Rayleigh channel and in that section some statistical properties of generated waveforms are described. This generated wave forms can be used for the simulation of diversity combine Rayleigh fading channel, frequency selective channel, MIMO channel etc. Here waveforms are generated by using sum of sinusoids channel modelling principle.

Frequency hopping channel simulator was the last simulator we implemented our thesis and it gave nice statistical behaviour compared with reference model as describe in that section. Parameter computation was done based on MEDS. Frequency hopping is highly using in GSM mobile communication.

Above mention channel simulators are important for the design, performance evaluation, optimization, and test of modern wireless communication systems.

The next chapter will be dealt 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

CONCLUSIONS AND RECOMMENDATIONS

Mobile radio Channel simulators are essential component in the development and accurate performance evaluation of wireless systems. Therefore, efficient design and implementation techniques are important in the design and verification of the developing channel simulation model. Wireless communication systems must be designed to operate over radio channels for a wide variety of environmental and weather conditions. Numerous real-time test cases are applied to a new communication system before release the products to the market. Modelling of channel simulators is one of the important tasks during this process.

The mobile radio channel simulators allow the performance evaluation of mobile communication systems under controlled and repeatable conditions that would not normally be possible in actual field testing. It is important that such a channel model represents all of the relevant behaviour and properties of actual propagation environments as accurately as possible. Therefore these developed channel simulators are important for the test, the parameter optimization, and the performance analysis of mobile communication systems.

This thesis considered the modelling and implementation of fading channel simulators based on the research findings listed in [2]. We developed a new toolbox for Matlab software based on the mobile fading channels and to provide some useful function for testing of the performance. User can employ developed channel models to simulate and observe the behaviours. Help file which was created for easy use of the developed system. The help file was developed based on HTML to improve user friendliness. Users have to download relevant m-files from the server and have to run by following the user guide. Users have to select or give appropriate input through the user interface following the help file. Further it is proposed to optimize solution for the channel models to improve its characteristics. The help file add value to developed Matlab toolbox providing user friendliness. In addition to help file each m-files contain help guidelines itself with further explanations. The software package MATLAB provides several toolboxes for signal processing, digital communications, statistics, etc. A toolbox for mobile radio channel which was not in the past system, is introduce in the present study to easy handling the mobile radio channel testing through a simulation.

This research led to start reviewing the reference of [2] and some articles related to channel fading domains to develop the present tool. After study of the relevant article and books we 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 as simple as possible and suitable for channel fading.

Before going to implementation of the channel models we planned to make flow of the work to make sure the consistent of the thesis work. Firstly we concentrated on the parameter computation for the both frequency selective and frequency-non-selective channel separately for later uses. Parameter computation was done for frequency non selective channel under two scenarios called parameter computation for sum-of-sinusoids channel model and sum-of-cisoids channel model. We have carried out parameter computation under different method and compared their performance with the reference model. It was done for both frequency-non-selective and frequency selective channel model. Especially we observed behaviour of the autocorrelation function of the simulation model and concluded each in results and are available in chapter 2.

Third chapter 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. New methods for parameter computation have been introduced with the increase efficiency. Each channel simulator was tested and checked simulation results with the reference model. Second order statistical properties like autocorrelation, cross-correlation, level- crossing rate, and average fade duration were concentrated. At this point, simulation results were compared with the reference model. In this research, an attempt was made to address our research question of "*How can a channel simulator be tested?"* by using above mention testing criteria's. It is precisely described in modelling of non-stationary land mobile satellite channel section in chapter 3. We understood that simulation time, size of the sample, sampling rate, number of scatters, selected parameter computation method etc. were affected for the test and performance. To get the best fitted behaviour of the simulation model compared with the reference model or measurements based model user have to select appropriate input values depending on the selected channel model simulator. The help file provide guidance to the user to select correct inputs.

As we mentioned above parameter computation methods and developed fading channel simulators were integrated as toolbox which supposed to be embed with the Matlab software as a new toolbox. Developed toolbox was manifest with higher performance and universal test functions. It is optimized toolbox and consisting with user friendly interface.

In addition to channel simulator and parameter computation; we programmed to compute and plot of different distributions and its corresponding cumulative distribution. At this point we were focused on uniform, Gaussian, Rayleigh, Rice, Nakagami_m, Nakagami_q, Suzuki and Weibull distributions for the toolbox. Developed m-files are attached to Appendix 5.

Due to the time limit constrains, we limited extension of some simulation models. Those remaining works open for the further work. Following recommendations are proposed for future studies as

- Upgrade developed toolbox with graphical user interface to provide more user-friendliness.
- The uncorrelated Rayleigh channel model was developed based on parameter computation method of $GMEDS_1$ but we can extend it by using parameter computation methods of MCM, RMEDS, MEDS-SP, MEA, $GMEDS_2$ and EMEDS to observe and compare the performance.
- Implemented mobile fading channel simulator was based on COST 207 but we can upgrade it by using standards of COST 273 (new European measurement for 3G mobile systems) as further work.
- Investigation of parameter computation methods for the non-isotropic scattering environment conditions.
- MIMO two-ring model can be extended based on frequency selective model.

REFERENCES

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

[2] M. Pätzold: “Mobile Radio Channels”, 2nd Edition, Chichester: John Wiley & Sons, 2011.

[3] S. Sesia, I. Toufik, and M. Baker, Eds., LTE – The UMTS Long Term Evolution: From Theory to Practice, Chichester: Jhon Wiley & sons,2009

[4] “Matlab documentation,” version 7.0.1.24704 (R14) September 13, 2004.

[5] M. Pätzold, Y. Li and F. Laue, “A study of a land mobile satellite channel model with asymmetrical Doppler power spectrum and lognormal distributed line-of-sight component,” IEEE trans. Veh. Technol., vol. 47, no. 1, pp. 51-54.

[6] 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.

[7] 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.

[8] R. H. Clarke, “A statistical theory of mobile-radio reception,” Bell Syst. Tech. J., pp. 957–1000, July–Aug. 1968

[9] W. C. Jakes, Microwave Mobile Communications: Wiley, 1974. reprinted by IEEE Press in 1994

[10] E. lutz, D. Cygan, M. Dippold, F. Dolainsky, and W. Papke, “ The land mobile satellite communication channel-recording, statistics, and channel model,” IEEE trans. Veh. Technol., vol. 40, no. 3,pp.738-742, Aug. 1994.

[11] M. Pätzold, A. Szczpanski, S. Buonomo, and F. Laue, “Modelling and simulation of non-stationary land mobile satellite channel by using extended Suzuki and handover processes” in proc. IEEE 51st veh. Technol. Conf.,VTC2000-spring, Tkyo, Japan, May 2000, pp. 1787-1792

[12] 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.

[13] 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.

[14] S´ebastien de la Kethulle, 27 September 2004. An Overview of MIMO Systems in

Wireless Communications.Available: http://www.iet.ntnu.no/projects/beats/Documents/mimo.pdf

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

[16] M. Pätzold and B. O. Hogstad : "A Wideband Space-Time MIMO Channel Simulator Based on the Geometrical One-Ring Model," Proc. 64th IEEE Semi-annual Vehicular Technology Conference, IEEE VTC 2006-Fall, Montreal, Canada, Sept. 2006.

[17] A. Abdi and M. Kaveh, "A space-time correlation model for multi-element antenna system in mobile fading channels," IEEE J. Select. Areas Commun., vol. 20, no. 3, pp. 550-560, April 2002.

[18] M. Pätzold, B. O. Hogstad, 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.

[19] M. Pätzold, B. O. Hogstad, N. Youssef, "Modelling, analysis, and simulation of MIMO mobile-to-mobile fading channel," IEICE Trans. Wireless Commun., vol. 7,no. 2, pp.510-520, Feb.2008

[20] M. Pätzold, C.X Wang, and B. O. Hogstad, "Two new sum-of-sinusoids-based method for the efficient generation of multiple uncorrelated Rayleigh fading waveforms," IEEE Trans. Wireless Communication , Vol. 8,no.6,pp. 3122-3131, June 2009.

[21] M. Pätzold and B. O. Hogstad, "design and performance of MIMO channel simulators derived from the two-ring scattering model," in Proc. 14th IST Mobile & Communications Summit, IST 2005, Dresden, Germany, June 2005, Paper no. 121.

[22] M. Pätzold and B. O. Hogstad, and A. Chopra, " A study on the capacity of narrow- and wideband MIMO channel model," in Proc. 15th IST Mobile & Communication Summit, IST 2006, Mycones, Greece, June 2006.

[23] 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.

[24] 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.

[25] Y. Li and X. huang, " The simulation of independent Rayleigh faders," IEEE Trans. Commun., vol. 50, no. 9, pp. 1503-1514, sep.2002.

[26] Y. R. Zheng and C. Xiao, "improved models for the generation of multiple uncorrelated Rayleigh fading waveforms," IEEE Communications Letters, vol.6, no. 6, pp. 256-258, June 2002.

[27] M. Pätzold, F. Laue, and H. Meyr, " A frequency hopping Rayleigh fading channel simulator with given correlation properties," in Proc. IEEE Int. Workshop on intelligent signal processing and communication systems, ISPACS'97, Kuala Lampur, Malaysia, Nov. 1997, pp. S8.1.1-S8.1.

ACRONYMS

ACF	autocorrelation function
ADF	average duration of fades
AoA	angle of arrival
AoD	angle of departure
API	application programming interface
ASK	amplitude shift keying
AWGN	additive white Gaussian noise
BS	base station
BU	Bad Urban
CCF	cross-correlation function
CDF	cumulative distribution function
CF	correlation function
COST	European Cooperation in the Field of Scientific and Technical research
DSP	digital signal processing
EA	elevation angle
EMEDS	Extended method of Exact Doppler Spread
FSK	frequency shift keying
FCF	frequency correlation function
FORTRAN	earlier programming language
GMEA	Generalized method of equal area
GSM	Global system for Mobile Communications
HIPERLAN	High Performance Radio Local Area Network
HT	Hilly Terrain
JM	Jakes method
LAN	local area network
LCR	level-crossing rate
LMS	land mobile satellite
LOS	line-of-sight
LPNM	L_p-norm method
LTE	Long Term Evolution
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
MS	mobile station
MSEM	mean-square-error method
NLOS	non-line-of-sight
OFDM	orthogonal frequency division multiplexing
PAM	phase-amplitude modulation
PDF	probability density function
PSD	Doppler power spectral density function
PSK	phase shift keying
QAM	quadrature amplitude modulation
RA	Rural Area
RMEDS	randomized method of exact Doppler spread
RV	random variable
SISO	single-input single-output
SNR	signal-to-noise ratio
SOC	Sum-of-cisoids
SOSUS	Sum-of-cisoids uncorrelated scattering
SOS	Sum-of-sinusoids
TDMA	time division multiple access
TU	typical urban
WSS	wide-sense stationary

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 $x_1, x_2, ..., x_N$
$\det A$	determinant of the matrix A
$erf\,(\cdot)$	error function
e^x	exponential function
$E\{x\}$	(statistical) mean value or expected 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 $[a,b]$
$< x(t) >$	time average operator

$x \to a$	x tends to a
$\lfloor x \rfloor$	floor function, the greatest integer less than or equal to x
$\sqcup$	distributed according to (statistic) or asymptotically equal (analysis)
$\leq$	less than or equal to
$=$	equal
$a_{m,n}$	the mth row and the nth of the matrix $\left(a_{m,n}\right)$
A^T	transpose matrix of the matrix A
A^{-1}	inverse matrix of the matrix A
α_v	Angle of motion
α_v^T	angle-of-motion of the transmitter
α_n^T	Angle of departure at base station
α_n^R	Angle of arrival at mobile station
$\alpha_{\max}^T$	One half of the maximum angle of departures seen at the base station
β_T	Multi-element antenna tilt angle at base station
β_R	Multi-element antenna tilt angle at mobile station
$\delta(\cdot)$	delta function
δ_T	Antenna element spacing at base station
δ_R	Antenna element spacing at mobile station
$\varsigma(t)$	Rayleigh process
$\Gamma(.)$	gamma function
$f_{\max}$	maximum Doppler frequency
$h(\tau', t)$	time-variant impulse response
σ_L	shadow standard deviation
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 $\tau'_{\ell-1}$
$\tilde{a}_\ell$	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	Doppler frequency
$f_{i,n}$	discrete Doppler frequency of the nth component of $\tilde{\mu}_i(t)$
f_c	cut-off frequency
f_n	Doppler frequency of the nth plane wave
f_s	Sampling frequency
f_ρ	Doppler frequency of the line-of-sight component *m(t)*
f_0	carrier frequency
$f_{i,n,\ell}$	discrete Doppler frequency of the nth component of $\tilde{\mu}_{i,\ell}(t)$
$H_{pq}(t)$	The channel gains matrix
K	number of simulated samples of a discrete deterministic process
κ	angular spread parameter of the von Mises distribution
κ_0	frequency ratio f_{min} over f_{max}
m	number of state
$m(t)$	line-of-sight component
m_L	area mean
m_α	mean direction of angle-of-arrival
m_{μ_i}	mean value of μ_i(t)
M	number of channel state characterizing the M-state Markov model
M_R	number of receiver antenna element
M_T	number of transmitter antenna element
$\mu\ (t)$	zero-mean complex Gaussian random process
$\mu_i(t)$	real Gaussian random process (stochastic reference model)
$\hat{\mu}_i(t)$	real stochastic process (stochastic simulation model)
N	number of propagation path
N_i	number of harmonic functions of $\tilde{\mu}_i(t)$
$N_\varsigma(t)$	level-crossing rate of Rice processes $\varsigma(t)$
$N_\eta(t)$	level-crossing rate of Suzuki processes $\eta(t)$
$N_\lambda(t)$	level-crossing rate of lognormal processes $\lambda(t)$
$N_\lambda(t)$	level-crossing rate of lognormal processes $\lambda(t)$
$N_\xi(t)$	level-crossing rate of Rice processes $\xi(t)$

r	amplitude level
$r_{\mu_i\mu_i}(t)$	autocorrelation function of $\mu_i(t)$
R	Ring radius
$\tilde{r}_{\mu_i\mu_i}(t)$	autocorrelation function of $\tilde{\mu}_i(t)$
$r_{\tau'\tau'}(v')$	frequency correlation function
$\tilde{S}_{\mu_i\mu_i}(f)$	power spectral density of $\tilde{\mu}_i(t)$
$S_{\mu_i\mu_i}(f)$	power spectral density of $\mu_i(t)$
t	time
u_n	random variable, uniformly distributed in the interval $(0,1]$
v	speed of the mobile unit
v_R	speed of the mobile receiver
v_T	speed of the mobile transmitter
$\vec{v}$	mobile speed vector
$\nu(x)$	real spatial Gaussian process
$\hat{\nu}(x)$	real spatial stochastic SOS process
$\Delta_{\min}$	minimum transition time interval
$\Delta_{\max}$	maximum transition time interval
$\theta_{i,n}$	Doppler phase of the nth component of $\tilde{\mu}_i(t)$
$\theta_{i,n,\ell}$	Doppler phase of the nth component of $\tilde{\mu}_{i,\ell}(t)$
θ_n	phase shift due to the interaction with the nth scatterer S_n
θ_ρ	phase of the line-of-sight component *m(t)*
θ_0	constant phase shift
$\tilde{\varsigma}(t)$	continuous-time deterministic
$\tilde{\eta}(t)$	continuous-time deterministic Suzuki process
$\lambda(t)$	lognormal process
$\lambda(x)$	spatial lognormal process
$\tilde{\lambda}(t)$	continuous-time deterministic lognormal process
$\tilde{\lambda}(x)$	spatial deterministic SOS lognormal process
λ_0	wave length
$\xi(t)$	Rice process
ρ	amplitude of the line-of-sight component of *m(t)*
χ	frequency separation variable
Ω	parameter vector of the reference model

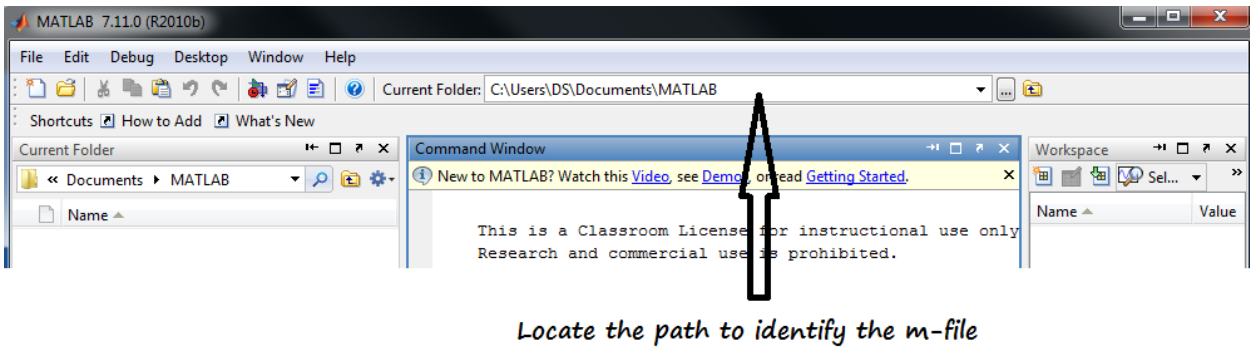
APPENDIX 1

USER INTERFACES OF THE HELP FILE

Attention:

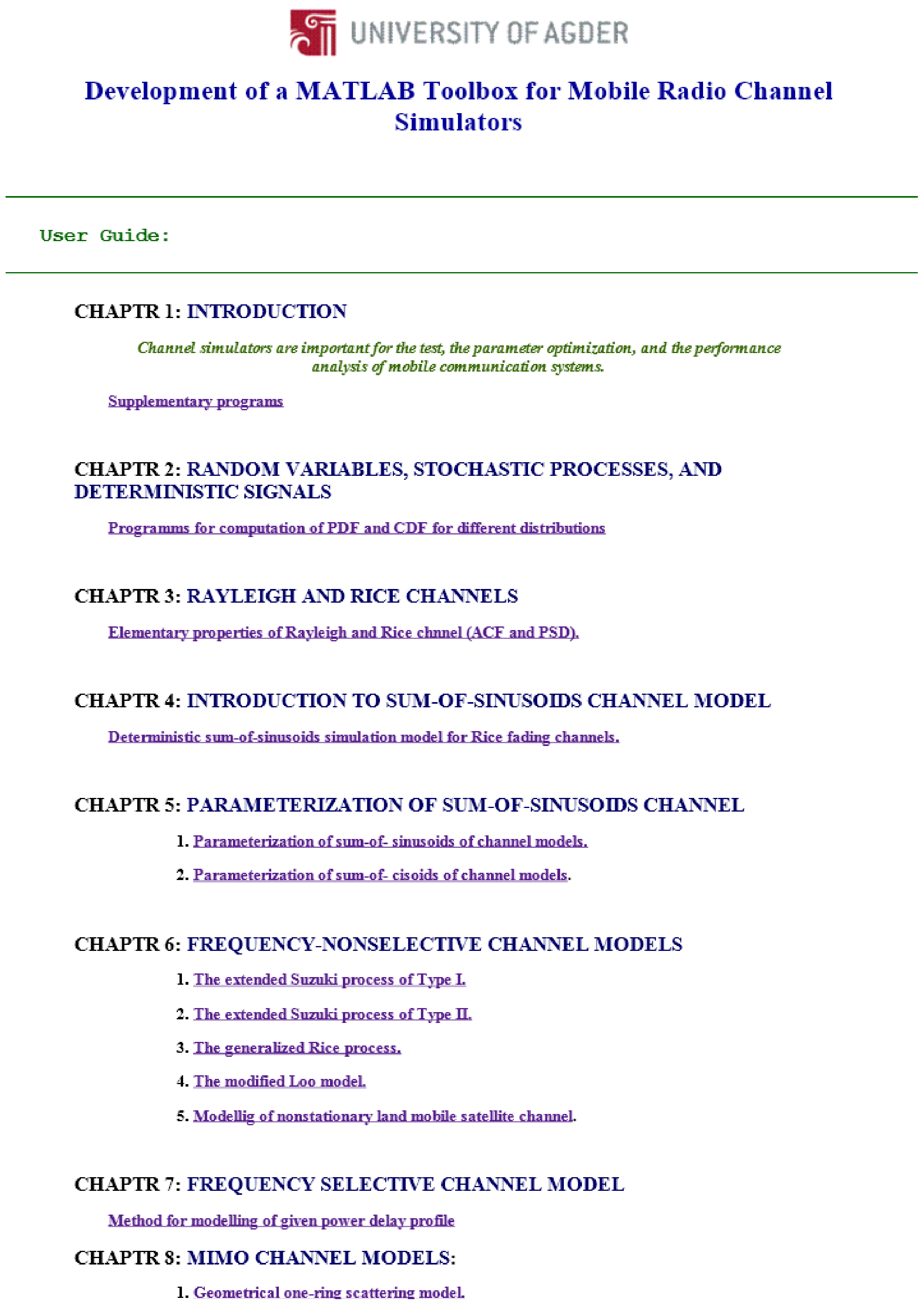
New users can use the web based help file interfaces to getting familiar with the developed toolbox. User can learn about the developed parameter computation methods and the channel simulators by clicking the relevant web link as shown in the interfaces. Help file provide guide lines to users how to select input parameters as well as how to execute the given Matlab programs. Each web interfaces have been linked with the sub web interfaces for the each sub headings. Sub interfaces linked with the relevant m-file program as well as plotted figures. User can compared his results with the given default results to make sure simulated results.

In addition to web base help file each m-file consists with help file itself. By clicking the name of the m-files user can get additional information regarding the relevant m-files which are given in the web interfaces. Flow of the help file was developed according to chapter organization of the reference [2]. Before run the developed simulators user has to download the given m-file to his computer from the server. Matlab software must be installed to user's computer. In addition to that, the path should be set to identify the m-files to Matlab software for the execution.


MATLAB 7.11.0 (R2010b)
File Edit Debug Desktop Window Help
Current Folder: C:\Users\DS\Documents\MATLAB
Shortcuts How to Add What's New
Current Folder
Documents ▸ MATLAB
Name
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Workspace
Sel...
Name
Value
Locate the path to identify the m-file

```
This is a Classroom License for instructional use only.
Research and commercial use is prohibited.
```

Main Interface:



UNIVERSITY OF AGDER

Development of a MATLAB Toolbox for Mobile Radio Channel Simulators

User Guide:

CHAPTR 1: INTRODUCTION

Channel simulators are important for the test, the parameter optimization, and the performance analysis of mobile communication systems.

Supplementary programs

CHAPTR 2: RANDOM VARIABLES, STOCHASTIC PROCESSES, AND DETERMINISTIC SIGNALS

Programms for computation of PDF and CDF for different distributions

CHAPTR 3: RAYLEIGH AND RICE CHANNELS

Elementary properties of Rayleigh and Rice chnnel (ACF and PSD).

CHAPTR 4: INTRODUCTION TO SUM-OF-SINUSOIDS CHANNEL MODEL

Deterministic sum-of-sinusoids simulation model for Rice fading channels.

CHAPTR 5: PARAMETERIZATION OF SUM-OF-SINUSOIDS CHANNEL

1. Parameterization of sum-of- sinusoids of channel models.
2. Parameterization of sum-of- cisoids of channel models.

CHAPTR 6: FREQUENCY-NONSELECTIVE CHANNEL MODELS

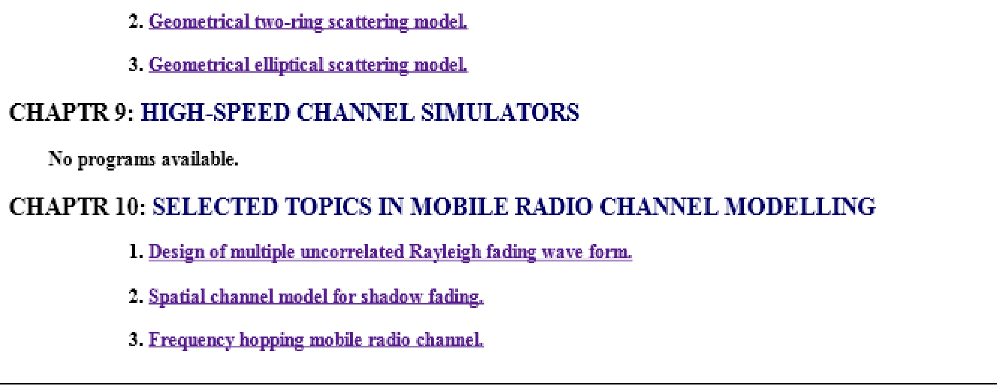
1. The extended Suzuki process of Type I.
2. The extended Suzuki process of Type II.
3. The generalized Rice process.
4. The modified Loo model.
5. Modellig of nonstationary land mobile satellite channel.

CHAPTR 7: FREQUENCY SELECTIVE CHANNEL MODEL

Method for modelling of given power delay profile

CHAPTR 8: MIMO CHANNEL MODELS:

1. Geometrical one-ring scattering model.



2. Geometrical two-ring scattering model.

3. Geometrical elliptical scattering model.

CHAPTR 9: HIGH-SPEED CHANNEL SIMULATORS

No programs available.

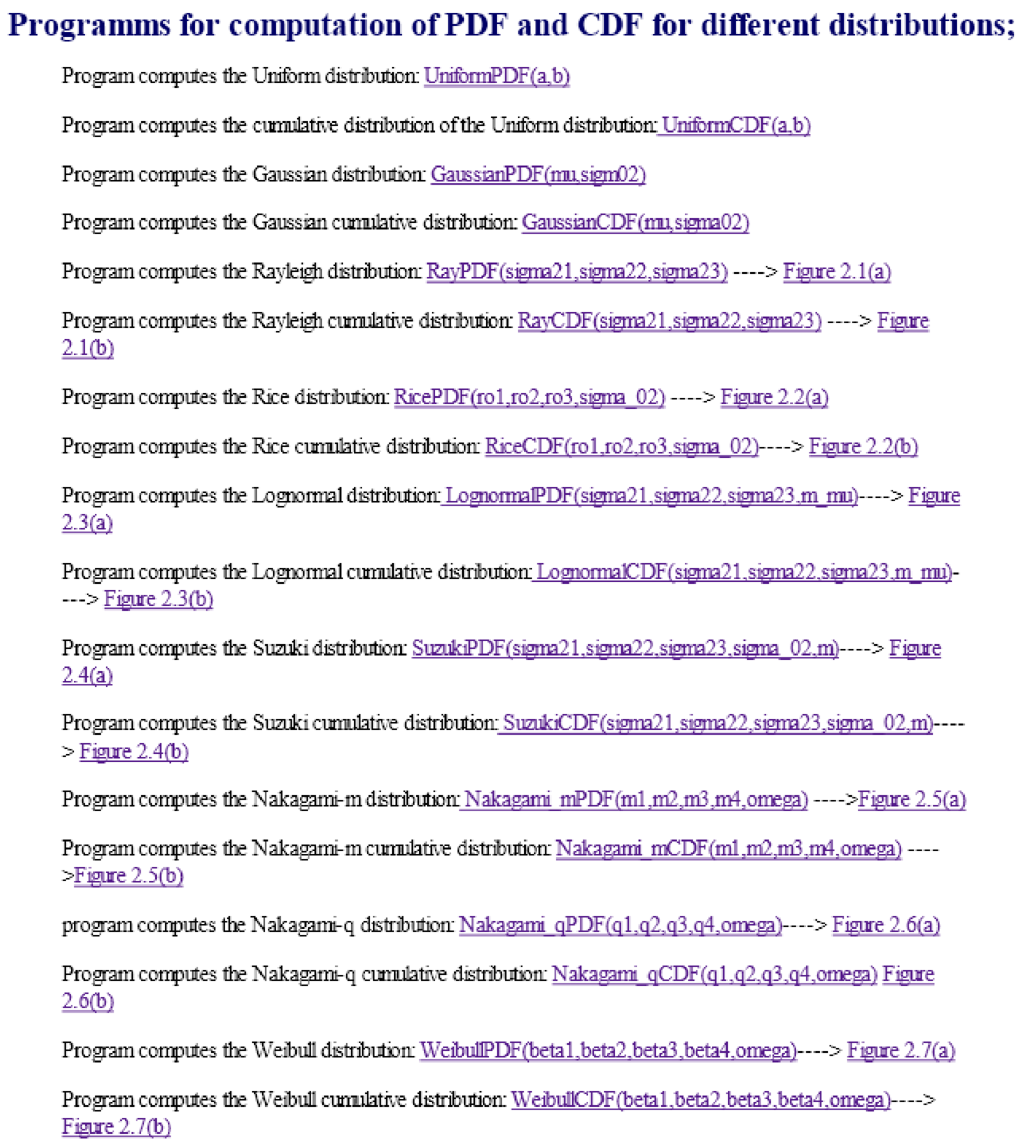
CHAPTR 10: SELECTED TOPICS IN MOBILE RADIO CHANNEL MODELLING

1. Design of multiple uncorrelated Rayleigh fading wave form.

2. Spatial channel model for shadow fading.

3. Frequency hopping mobile radio channel.

Sub Interfaces:



Programms for computation of PDF and CDF for different distributions;

Program computes the Uniform distribution: UniformPDF(a,b)

Program computes the cumulative distribution of the Uniform distribution: UniformCDF(a,b)

Program computes the Gaussian distribution: GaussianPDF(mu,sigm02)

Program computes the Gaussian cumulative distribution: GaussianCDF(mu,sigma02)

Program computes the Rayleigh distribution: RayPDF(sigma21,sigma22,sigma23) ----> Figure 2.1(a)

Program computes the Rayleigh cumulative distribution: RayCDF(sigma21,sigma22,sigma23) ----> Figure 2.1(b)

Program computes the Rice distribution: RicePDF(ro1,ro2,ro3,sigma_02) ----> Figure 2.2(a)

Program computes the Rice cumulative distribution: RiceCDF(ro1,ro2,ro3,sigma_02)----> Figure 2.2(b)

Program computes the Lognormal distribution: LognormalPDF(sigma21,sigma22,sigma23,m_mu)----> Figure 2.3(a)

Program computes the Lognormal cumulative distribution: LognormalCDF(sigma21,sigma22,sigma23,m_mu)----> Figure 2.3(b)

Program computes the Suzuki distribution: SuzukiPDF(sigma21,sigma22,sigma23,sigma_02,m)----> Figure 2.4(a)

Program computes the Suzuki cumulative distribution: SuzukiCDF(sigma21,sigma22,sigma23,sigma_02,m)----> Figure 2.4(b)

Program computes the Nakagami-m distribution: Nakagami_mPDF(m1,m2,m3,m4,omega) ---->Figure 2.5(a)

Program computes the Nakagami-m cumulative distribution: Nakagami_mCDF(m1,m2,m3,m4,omega) ---->Figure 2.5(b)

program computes the Nakagami-q distribution: Nakagami_qPDF(q1,q2,q3,q4,omega)----> Figure 2.6(a)

Program computes the Nakagami-q cumulative distribution: Nakagami_qCDF(q1,q2,q3,q4,omega) Figure 2.6(b)

Program computes the Weibull distribution: WeibullPDF(beta1,beta2,beta3,beta4,omega)----> Figure 2.7(a)

Program computes the Weibull cumulative distribution: WeibullCDF(beta1,beta2,beta3,beta4,omega)----> Figure 2.7(b)

Elementary properties of Rayleigh and Rice chnnel (ACF and PSD)

1. **JakePSD(f_max,sigma02): This program computes the Jakes power spectral density (Figure3.2(a))**

2. **JakeACF(f_max,sigma02): This program computes corresponding autocorrelation function of the Jakes power spectral density (Figure 3.2(b))**

3. **GaussianPSD(f_max,sigma02): This program computes the Gaussian power spectral density (Figure 3.3(a))**

4. **GaussianACF(f_max,sigma02) : This program computes corresponding ACF of the Gaussian power spectral density (Figure 3.3(b))**

5. **RicePDF_SE(rho1,rho2,rho3,sigma02): This program computes the probability density function of the squared envelop of Rice process.(Figure 3.7.(a))**

6. **RayeilyPDF_SE(sigma01,sigma02,sigma03): This program computes the probability density function of the squared envelop of the Rice process given in the equation (3.62). (Figure 3.7.(b))**

Home page

Deterministic sum-of-sinusoids simulation model for Rice fading channels

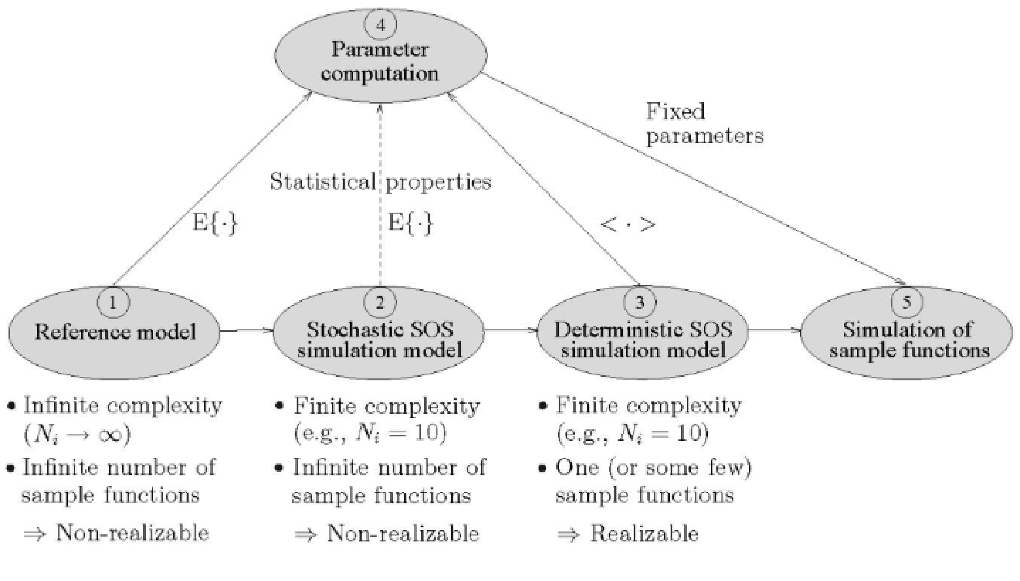

4
Parameter computation
Fixed parameters
Statistical properties
E{·}
E{·}
< · >
1
Reference model
2
Stochastic SOS simulation model
3
Deterministic SOS simulation model
5
Simulation of sample functions
• Infinite complexity ($N_i \to \infty$)
• Infinite number of sample functions
⇒ Non-realizable
• Finite complexity (e.g., $N_i = 10$)
• Infinite number of sample functions
⇒ Non-realizable
• Finite complexity (e.g., $N_i = 10$)
• One (or some few) sample functions
⇒ Realizable

Figure 1: Illustration of the principle of deterministic channel modelling

The steps of deterministic channel modeling:

Step 1: Starting point is the reference model; based on one or several Gaussian processes, each with prescribe autocorrelation function.

Step 2: Derive a stochastic simulation model from the reference model by replacing the Gaussian process by sum-of sinusoid with fixed gain, fixed frequency, and random phases.

Step 3: Determine a deterministic simulation model by fixing all model parameters of the stochastic simulation model including the phases.

Step 4: Compute the model parameters of the simulation model by fitting the relevant statistical properties of the deterministic (or stochastic) simulation model to those of reference model.

Step 5: perform the simulation of one (or some few) sample functions (Deterministic processes).

simulation model for Rice fading channels:

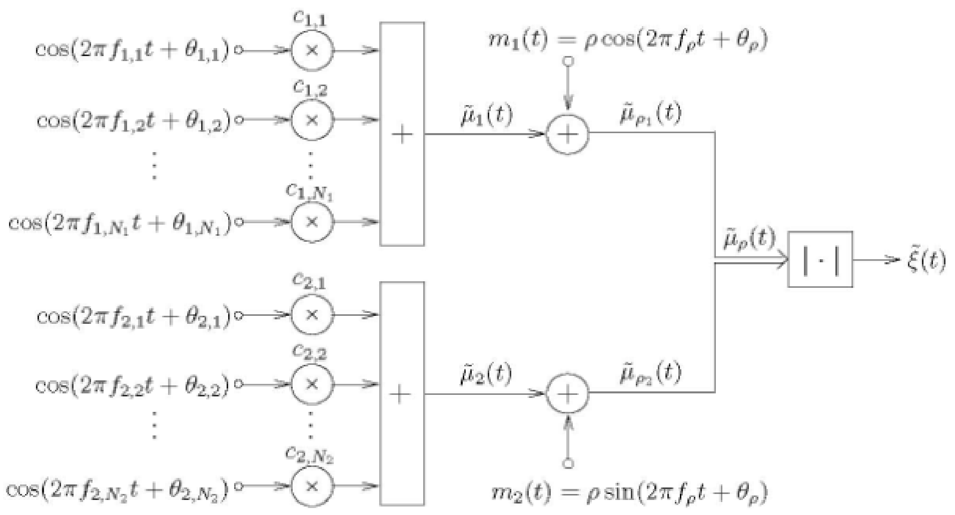

$\cos(2\pi f_{1,1}t+\theta_{1,1})$
$\cos(2\pi f_{1,2}t+\theta_{1,2})$
$\cos(2\pi f_{1,N_1}t+\theta_{1,N_1})$
$c_{1,1}$
$c_{1,2}$
c_{1,N_1}
$m_1(t)=\rho\cos(2\pi f_\rho t+\theta_\rho)$
$\tilde{\mu}_1(t)$
$\tilde{\mu}_{\rho_1}(t)$
$\tilde{\mu}_\rho(t)$
$|\cdot|$
$\tilde{\xi}(t)$
$\cos(2\pi f_{2,1}t+\theta_{2,1})$
$\cos(2\pi f_{2,2}t+\theta_{2,2})$
$\cos(2\pi f_{2,N_2}t+\theta_{2,N_2})$
$c_{2,1}$
$c_{2,2}$
c_{2,N_2}
$\tilde{\mu}_2(t)$
$\tilde{\mu}_{\rho_2}(t)$
$m_2(t)=\rho\sin(2\pi f_\rho t+\theta_\rho)$

Figure 2: Deterministic sum-of-sinusoids simulation model for Rice fading channels

Program for the simulation of deterministic Rice processes zeta(t) :

Rice_proc.m

Rice_Proc_Jakes.m

gen_Rice_proc.m

Program for the simulation of real deterministic Gaussian processes mu_i(t):

Mu_i_t.m

Test and perfomance:

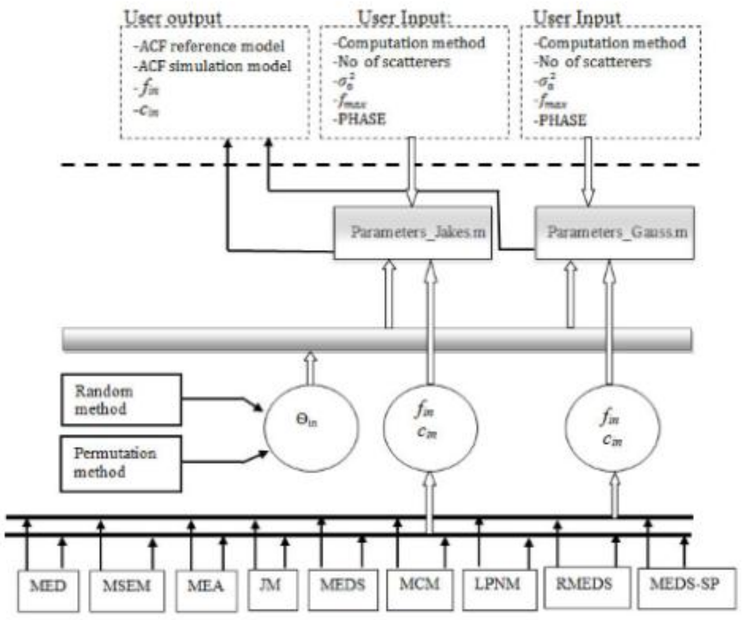
Program computes the PDF of p_mu_i(x): mu_i_PDF.m

Program computes the PDF of zeta_i(x) (Figure 4.13, Figure 4.14) of the simulation model: pdf_sim.m

Program for the computation of probability density functions (Figure 4.15) of p(theta): pdf_phase.m

Home page

PARAMETERIZATION OF SUM-OF-SINUSOIDS CHANNEL



The implementation diagram of the parameters computation methods for the SOS channel

Implimentation:

- Conditions on the computation methods: MEA, MSEM, MED, MCM, LPNM, JM, MEDS, RMEDS, 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).

  - Model of Jakes power spectral density

    parameter_Jakes.m

    LPNM_opt_Jakes.m

    grad_Jakes.m

    acf_mue.m

    fun_Jakes.m

- Model of Gauss power spectral density

  `parameter_Gauss.m`

  `LPNM_opt_Gauss.m`

  `fun_Gauss.m`

  `grad_Gauss.m`

  `acf_mue.m`

Go to home page

PARAMETERIZATION OF SUM-OF-CISOIDS CHANNEL

sum-of cisoids process generally represent as form of

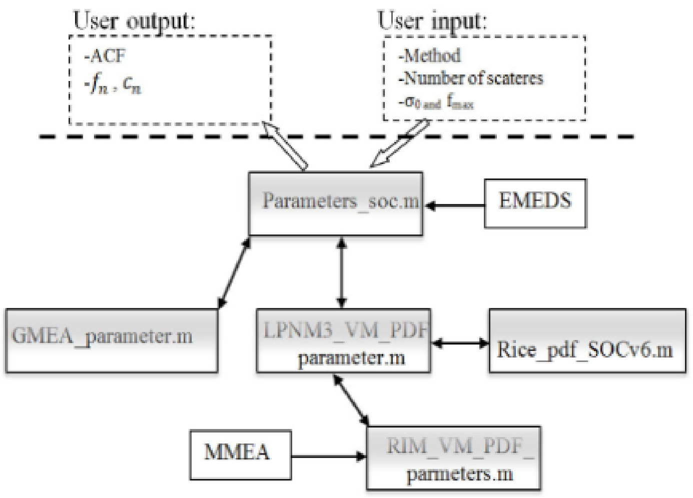
$$\hat{\mu}(t) = \sum_{n=1}^{N} c_n \, e^{j(2\pi f_n t + \theta_n)},$$

Where c_n, f_n and θ_n are the path gains, Doppler frequencies, and phase of the nth propagation path. Here we assume that θ_n is i.i.d random variable which uniformly distributed over $(0, 2\pi)$.

The gain c_n and f_n are determined by using three methods namely the extended method of exact Doppler spread (EMEDS) , the Lp-norm (LPNM), and the generalized method of equal area (GMEA).

Implimentation:

The user has to first pick the appropriate computation method for the Doppler frequencies, and Doppler coefficients that can be, for instance GMEA, LPNM or EMEDS. Secondly, he sets the number of scatters, the variance , and the frequency Doppler shift.



The implementation diagram of the parameters computation methods for SOC channel models

- The LPNM method needs two functionsof RIM_VM_PDF_parameter.m and rice_pdf_SOCv6)

Related m-files :

parameter_soc.m

GMEA_parameter.m (Figure 50.70)

LPNM3_VM_PDF_parameter.m

rice_pdf_SOCv6.m

RIM_VM_PDF_parameter.m

Go to home page

The extended Suzuki process of Type I.

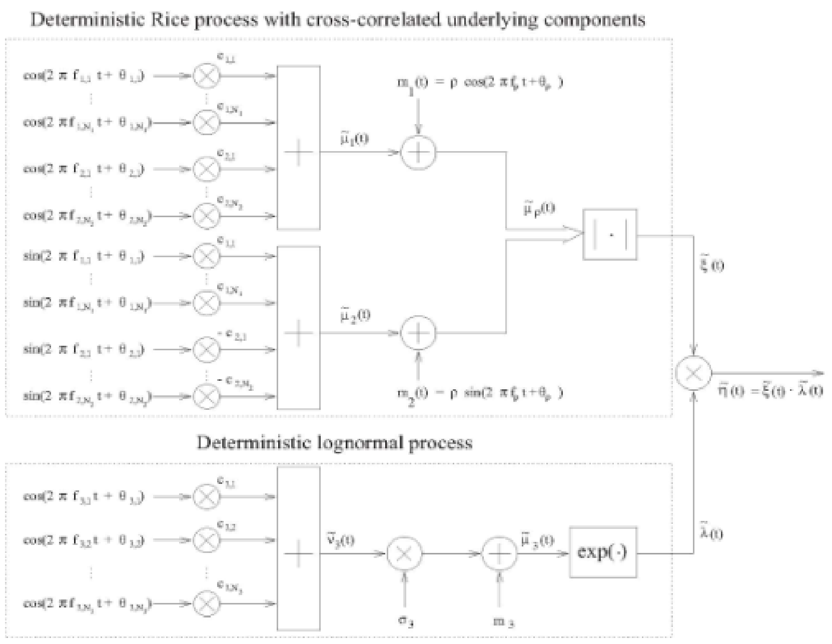

Deterministic Rice process with cross-correlated underlying components
Deterministic lognormal process
$m_1(t) = \rho\cos(2\pi f_\rho t + \theta_\rho)$
$m_2(t) = \rho\sin(2\pi f_\rho t + \theta_\rho)$
$\tilde{\mu}_1(t)$
$\tilde{\mu}_2(t)$
$\tilde{\mu}_\rho(t)$
$\tilde{\xi}(t)$
$\tilde{\lambda}(t)$
$\tilde{\eta}(t) = \tilde{\xi}(t)\cdot\tilde{\lambda}(t)$
$\tilde{\nu}_3(t)$
$\tilde{\mu}_3(t)$
exp(·)
σ_3
m_3

Figure 6.9: Deterministic simulation model for extended Suzuki process (Type 1)

Simulation programm for the extended Suzuki process of Type I:

Program for the simulation of deterministic extended Suzuki processes of Type I (see Fig. 6.9).

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)

Go to home page

The extended Suzuki process of Type II.

Modeling and analysis of short term fading:

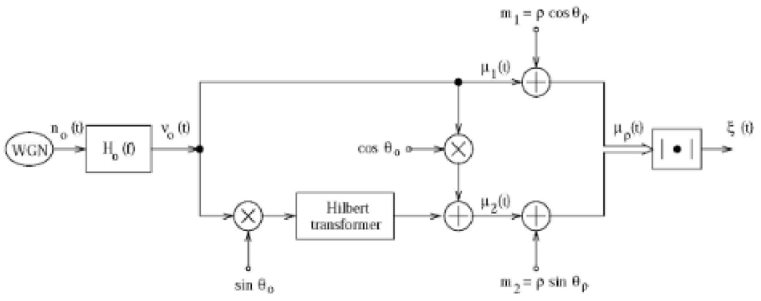


Figure 6.14: Reference model for stochastic process zeta(t) with cross-correlated Gaussian random process mu_1(t) and mu_2(t)

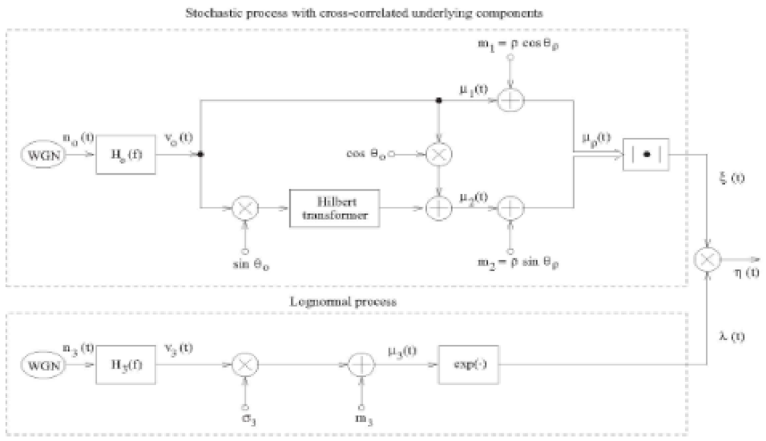
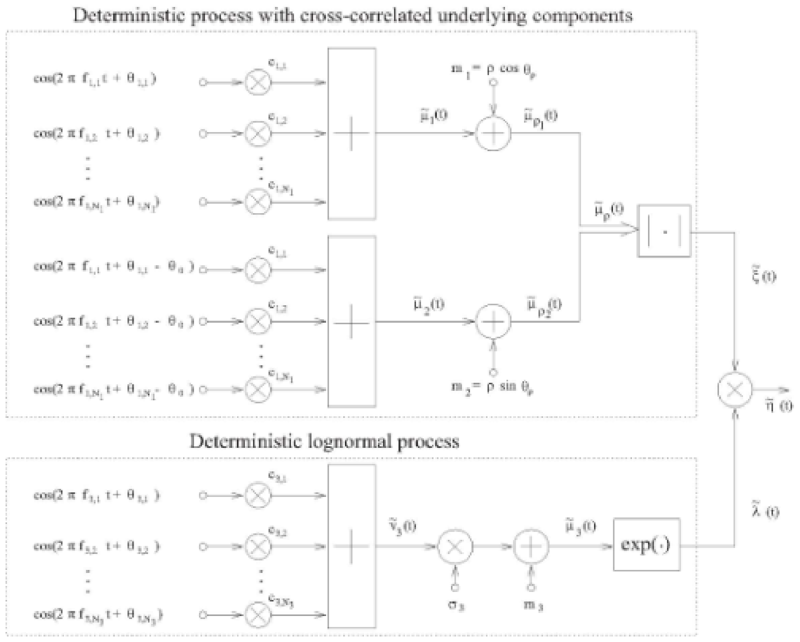


Figure 6.20: Reference model for extended Suzuki process (Type II)



Deterministic process with cross-correlated underlying components
$\cos(2\pi f_{1,1} t + \theta_{1,1})$
$\cos(2\pi f_{1,2} t + \theta_{1,2})$
$\cos(2\pi f_{1,N_1} t + \theta_{1,N_1})$
$c_{1,1}$
$c_{1,2}$
c_{1,N_1}
$m_1 = \rho \cos\theta_\rho$
$\tilde{\mu}_1(t)$
$\tilde{\mu}_{\rho_1}(t)$
$\tilde{\mu}_\rho(t)$
$\cos(2\pi f_{1,1} t + \theta_{1,1} - \theta_0)$
$\cos(2\pi f_{1,2} t + \theta_{1,2} - \theta_0)$
$\cos(2\pi f_{1,N_1} t + \theta_{1,N_1} - \theta_0)$
$\tilde{\mu}_2(t)$
$\tilde{\mu}_{\rho_2}(t)$
$m_2 = \rho \sin\theta_\rho$
$\tilde{\xi}(t)$
$\tilde{\eta}(t)$
Deterministic lognormal process
$\cos(2\pi f_{3,1} t + \theta_{3,1})$
$\cos(2\pi f_{3,2} t + \theta_{3,2})$
$\cos(2\pi f_{3,N_3} t + \theta_{3,N_3})$
$c_{3,1}$
$c_{3,2}$
c_{3,N_3}
$\tilde{\nu}_3(t)$
$\tilde{\mu}_3(t)$
exp(·)
$\tilde{\lambda}(t)$
σ_3
m_3

Figure 6.23: Deterministic simulation model for extended Suzuki processes (Type II)

Simulation programm for the extended Suzuki process of Type II:

Program for the simulation of deterministic extended Suzuki processes of Type II (see Fig. 6.23).

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)

Go to home page

The generalized Rice process.

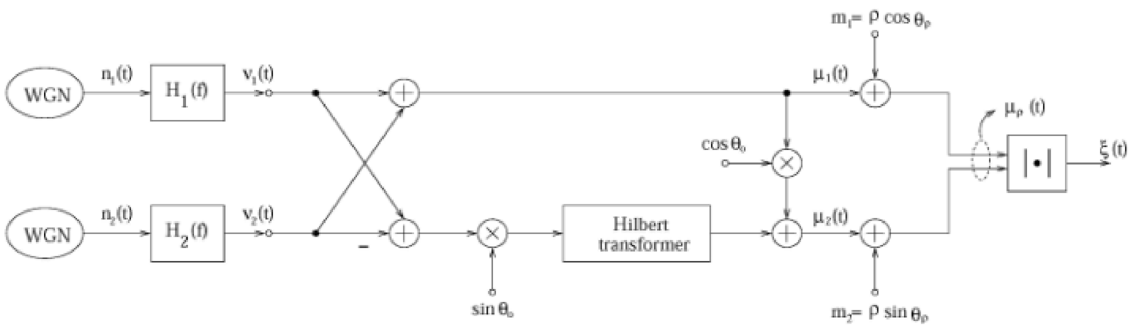
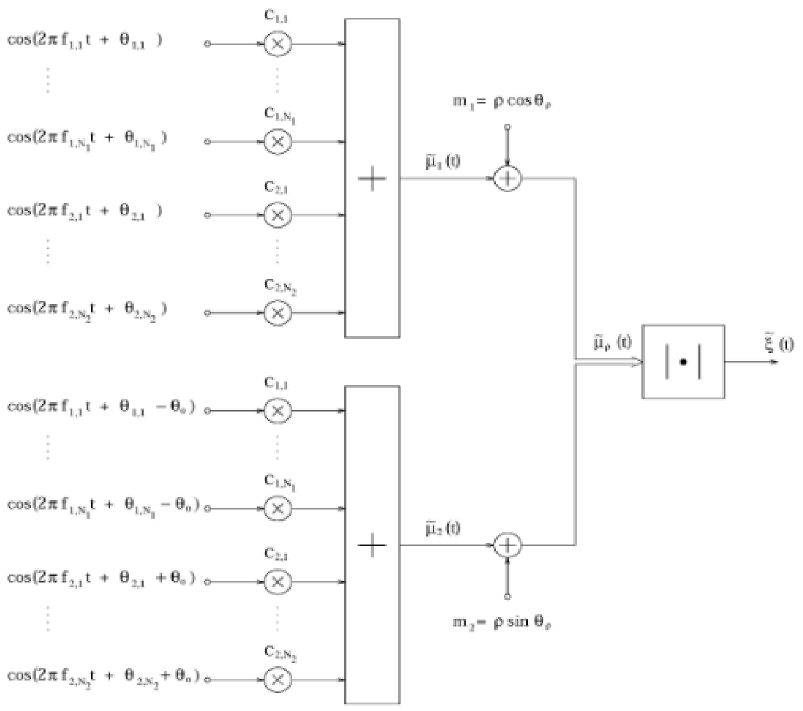


Figure 6.27: Reference model for generalized Rice process



$\cos(2\pi f_{1,1}t + \theta_{1,1})$
$c_{1,1}$
$\cos(2\pi f_{1,N_1}t + \theta_{1,N_1})$
c_{1,N_1}
$\cos(2\pi f_{2,1}t + \theta_{2,1})$
$c_{2,1}$
$\cos(2\pi f_{2,N_2}t + \theta_{2,N_2})$
c_{2,N_2}
$\cos(2\pi f_{1,1}t + \theta_{1,1} - \theta_0)$
$\cos(2\pi f_{1,N_1}t + \theta_{1,N_1} - \theta_0)$
$\cos(2\pi f_{2,1}t + \theta_{2,1} + \theta_0)$
$\cos(2\pi f_{2,N_2}t + \theta_{2,N_2} + \theta_0)$
$m_1 = \rho\cos\theta_\rho$
$\tilde{\mu}_1(t)$
$\tilde{\mu}_2(t)$
$m_2 = \rho\sin\theta_\rho$
$\tilde{\mu}_\rho(t)$
$|\bullet|$
$\tilde{\xi}(t)$

Figure 6.29: Deterministic simulation model for generalized Rice processes

Simulation programm for the generalized Rice processes:

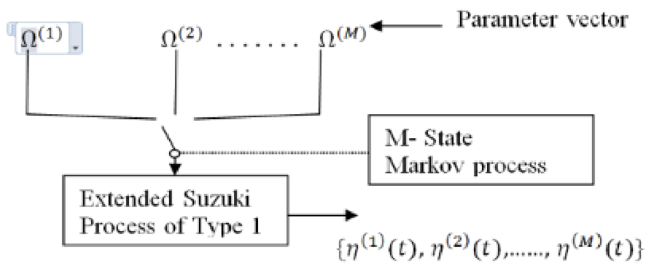
Program for the simulation of deterministic generalized Rice processes (see Fig. 6.29)

gen_Rice_proc.m

Go to home page

NON-STATIONARY LAND MOBILE SATELLITE CHANNEL

In order to provide global coverage with a high signal quality to diverse users, seamless integration of terrestrial and satellites networks are expected to play a vital role in the upcoming era of mobile communications. We show how the statistical properties of the proposed dynamic channel model can be adapted to the statistics of real-world LMS channels in different environments. From the dynamic m-state model, which is considered as reference model, an efficient simulation model is derived by replacing all colored Gaussian noise processes by finite sums of sinusoidal functions. It is demonstrated by several theoretical and simulation results that the performance of the reference model and the there from derived simulation model are excellent with respect to the probability density function (PDF) of the fading envelope, level-crossing rate (LCR), and average duration of fades (ADF).

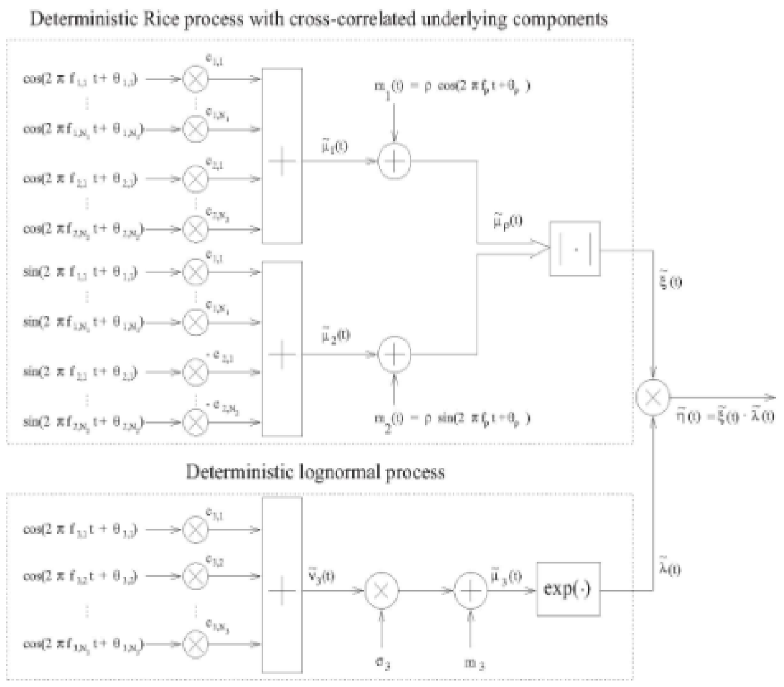

$\Omega^{(1)}$
$\Omega^{(2)}$ ....
$\Omega^{(M)}$
Parameter vector
M- State Markov process
Extended Suzuki Process of Type 1
$\{\eta^{(1)}(t), \eta^{(2)}(t), ..., \eta^{(M)}(t)\}$

Dynamic model with M states for modelling of non-stationary LMS channel

parameter vector define as;

$$\Omega = (\ \sigma_0,\ \kappa_0,\ \sigma_3, m_3, \rho\ \ , f_\rho,\ f_{\max})$$

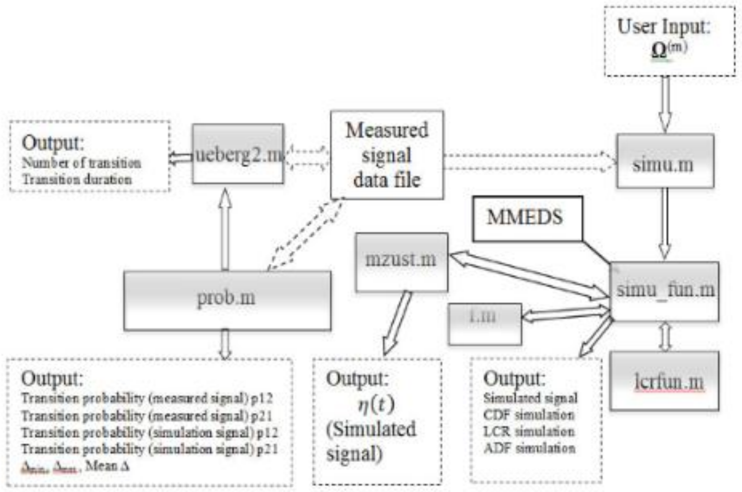
Simulation model:


Deterministic Rice process with cross-correlated underlying components
$\cos(2\pi f_{1,1} t + \theta_{1,1})$
$\cos(2\pi f_{1,N_1} t + \theta_{1,N_1})$
$\cos(2\pi f_{2,1} t + \theta_{2,1})$
$\cos(2\pi f_{2,N_2} t + \theta_{2,N_2})$
$\sin(2\pi f_{1,1} t + \theta_{1,1})$
$\sin(2\pi f_{1,N_1} t + \theta_{1,N_1})$
$\sin(2\pi f_{2,1} t + \theta_{2,1})$
$\sin(2\pi f_{2,N_2} t + \theta_{2,N_2})$
$c_{1,1}$
c_{1,N_1}
$c_{2,1}$
c_{2,N_2}
$-c_{2,1}$
$-c_{2,N_2}$
$\tilde{\mu}_1(t)$
$\tilde{\mu}_2(t)$
$m_1(t) = \rho\cos(2\pi f_\rho t + \theta_\rho)$
$m_2(t) = \rho\sin(2\pi f_\rho t + \theta_\rho)$
$\tilde{\mu}_\rho(t)$
$|\cdot|$
$\tilde{\xi}(t)$
$\tilde{\eta}(t) = \tilde{\xi}(t)\cdot\tilde{\lambda}(t)$
Deterministic lognormal process
$\cos(2\pi f_{3,1} t + \theta_{3,1})$
$\cos(2\pi f_{3,2} t + \theta_{3,2})$
$\cos(2\pi f_{3,N_3} t + \theta_{3,N_3})$
$c_{3,1}$
$c_{3,2}$
c_{3,N_3}
$\tilde{\nu}_3(t)$
$\tilde{\mu}_3(t)$
σ_3
m_3
$\exp(\cdot)$
$\tilde{\lambda}(t)$

Deterministic simulation model for extended Suzuki process (Type I)

Implementation:

- Computation of the parameters: ueberg2.m and prob.m m-file use to determine reference model parameters such as number of transition of the process from the measured data file, The prob.m, computation of the transition probabilities, Δmin , Δmax , Mean Δ etc. from the measured signal data file.
- Determination of the processes: The simu_fun.m file generate the simulated signal by using simu.m file which based on the extended Suzuki type I process. The mzust.m program use to switch both the extended Suzuki processes based on transition probabilities and time sharing factor to generate equivalent satellite signal of and
- 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 complimentary CDF, LCR and ADF, each in a single matlab file.


User Input:
$\Omega^{(m)}$
Output:
Number of transition
Transition duration
ueberg2.m
Measured signal data file
simu.m
MMEDS
mzust.m
simu_fun.m
prob.m
f.m
lcrfun.m
Output:
Transition probability (measured signal) p12
Transition probability (measured signal) p21
Transition probability (simulation signal) p12
Transition probability (simulation signal) p21
Δ_{min}, Δ_{max}, Mean Δ
Output:
$\eta(t)$
(Simulated signal)
Output:
Simulated signal
CDF simulation
LCR simulation
ADF simulation

Block diagram for implementation of the 2 states non-stationary LMS simulation channel

Simulation m_file:

```
x: Input parameter vector of the extended Suzuki processes of Type 1.
sigma_o=x(1);
rho    =x(2);
m3     =x(3);
sigma3 =x(4);
kappa0 =x(5);
frho   =x(6);
fmax   =x(7);
```

simu.m :

simu_fun.m:

mzust.m:

f.m:

lcrfun.m:

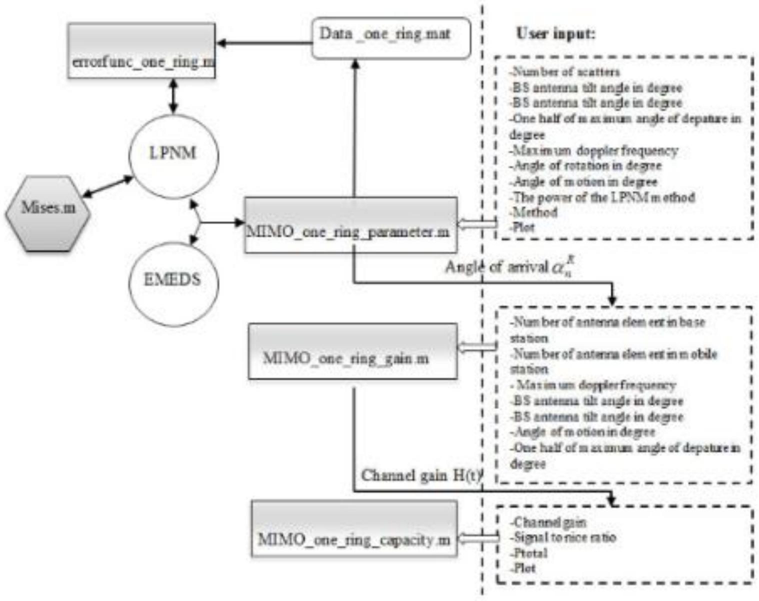
ueberg2.m:

prob.m:

GEOMETRICAL ONE-RING SCATTERING MODEL

Implementation:

- First, we determine the angles of arrivals using two fundamental methods. 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.


errorfunc_one_ring.m
Data _one_ring.mat
User input:
-Number of scatters
-BS antenna tilt angle in degree
-BS antenna tilt angle in degree
-One half of maximum angle of depature in degree
-Maximum doppler frequency
-Angle of rotation in degree
-Angle of motion in degree
-The power of the LPNM method
-Method
-Plot
LPNM
Mises.m
MIMO_one_ring_parameter.m
EMEDS
Angle of arrival α_n^R
-Number of antenna element in base station
-Number of antenna element in mobile station
- Maximum doppler frequency
-BS antenna tilt angle in degree
-BS antenna tilt angle in degree
-Angle of motion in degree
-One half of maximum angle of depature in degree
MIMO_one_ring_gain.m
Channel gain H(t)
MIMO_one_ring_capacity.m
-Channel gain
-Signal to rice ratio
-Ptotal
-Plot

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

Simulated program m-files:

MIMO_one_ring_parameter.m

MIMO_one_ring_gain.m

errorfun_one_ring.m

MIMO_one_ring_capacity.m

fun_phi.m

Mises.m

Data_one_ring.mat

Simulation results:

Isotropic scattering:

RUN ---> [alpha]=MIMO_one_ring_parameters(25,90,90,2,0,91,0,180,1,'emed',1)

Non-isotropic scattering:

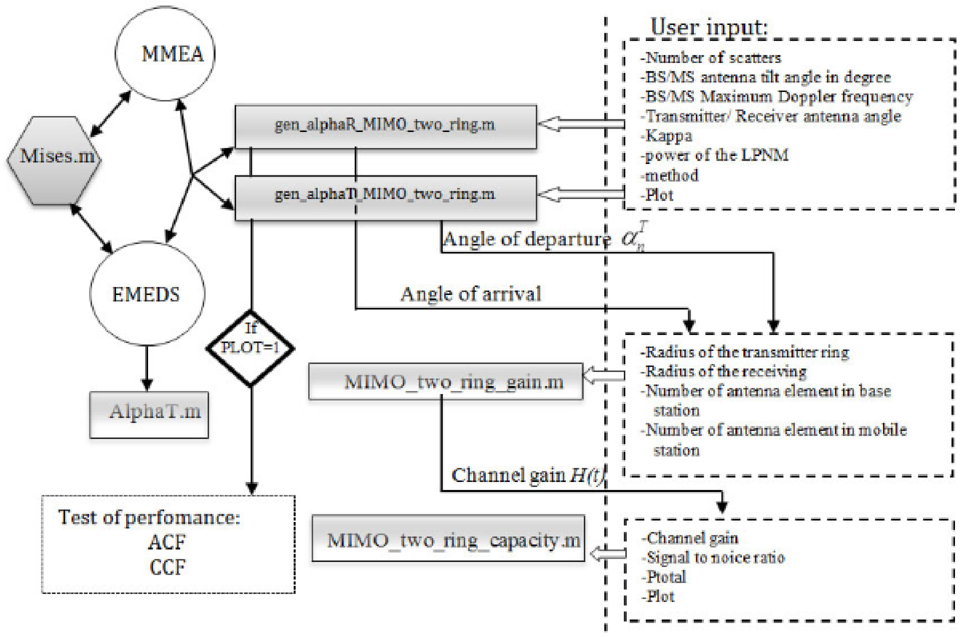
RUN ---> [alpha]=MIMO_one_ring_parameters(25,90,90,2,10,91,0,180,2,'lpnm',1)

Go to home page

THE TWO-RING MIMO CHANNEL MODEL

Implimentation:

- The functions gen_alphaT_MIMO_two_ring.m and gen_alphaR_MIMO_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_ring_gains.m. To make sure that the angles are different in order to de-correlate the channels, we use the EMEDS method.
- The option PLOT is provided to sketch the performance criteria, such as temporal ACF and space-time CCF.


MMEA
Mises.m
EMEDS
AlphaT.m
gen_alphaR_MIMO_two_ring.m
gen_alphaT_MIMO_two_ring.m
Angle of departure α_n^T
Angle of arrival
If PLOT=1
MIMO_two_ring_gain.m
Channel gain $H(t)$
MIMO_two_ring_capacity.m
Test of perfomance: ACF CCF
User input:
-Number of scatters
-BS/MS antenna tilt angle in degree
-BS/MS Maximum Doppler frequency
-Transmitter/ Receiver antenna angle
-Kappa
-power of the LPNM
-method
-Plot
-Radius of the transmitter ring
-Radius of the receiving
-Number of antenna element in base station
-Number of antenna element in mobile station
-Channel gain
-Signal to noice ratio
-Ptotal
-Plot

The diagram for the implementation of a MIMO channel simulator based on two-ring geometrical model

Simulation program m-files:

MIMO_two_ring_gain.m

gen_alphaR_MIMO_tow_ring.m

gen_alphaT_MIMO_tow_ring.m

fun_alphaT.m

Mises.m

MIMO_two_ring_capacity.m

Simulation results:

Isotropic scattering:

RUN ---> [alphaT]=gen_alphaT_MIMO_tow_ring(40,180,91,90,0,2,'emed',1)

Non-isotropic scattering:

RUN ---> [alphaT]=gen_alphaT_MIMO_tow_ring(40,180,91,90,40,2,'mmea',1)

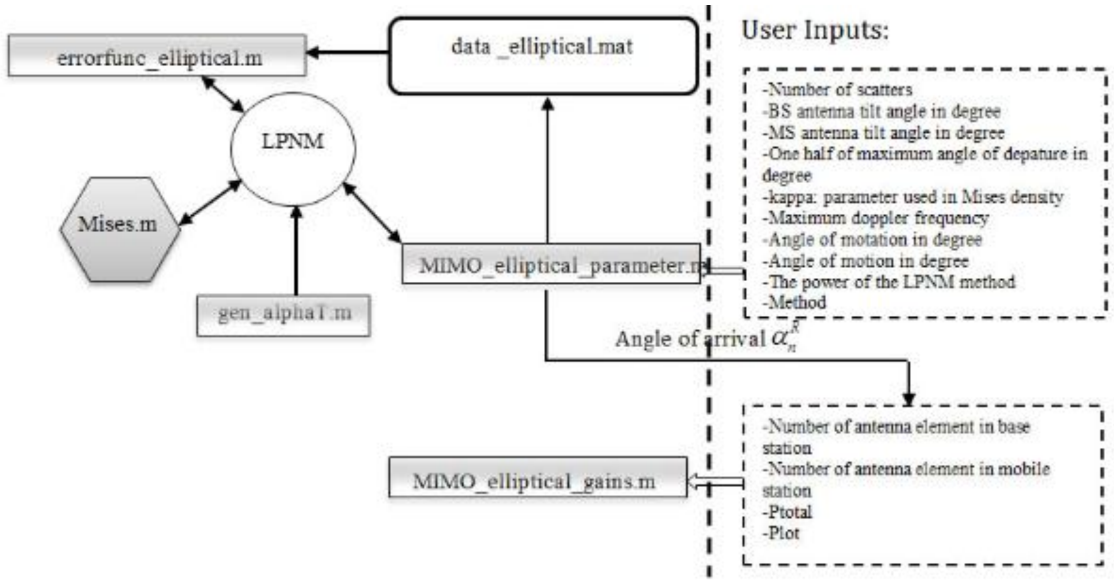
Go to home page

MIMO CHANNEL SIMULATOR BASED ON ELLIPTICAL MODEL

Implementation:

The figure illustrates the block diagram of our simulator under the Matlab environment. Here follows a simple guidance on how the user can simulate the elliptical model in below:

- To compute the angles of arrival, the user has to run the program called MIMO_elliptical_parameters.m. This program is related to gen_alphaT.m which generates the angles of departure starting from the angles of arrival with using the LPNM method.
- 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.


errorfunc_elliptical.m
data _elliptical.mat
User Inputs:
LPNM
Mises.m
gen_alphaT.m
MIMO_elliptical_parameter.m
-Number of scatters
-BS antenna tilt angle in degree
-MS antenna tilt angle in degree
-One half of maximum angle of depature in degree
-kappa: parameter used in Mises density
-Maximum doppler frequency
-Angle of motation in degree
-Angle of motion in degree
-The power of the LPNM method
-Method
Angle of arrival α_n^R
MIMO_elliptical_gains.m
-Number of antenna element in base station
-Number of antenna element in mobile station
-Ptotal
-Plot

Block diagram for the implementation of a MIMO channel simulator based on elliptical geometrical model

Simulation m-files:

MIMO_elliptical_parameters

MIMO_elliptical_gains.m

errorfunc_elliptical.m

gen_alphaT.m

data _elliptical.mat

Mises.m

Simulation results:

Isotropic scattering: ($K = 0$)

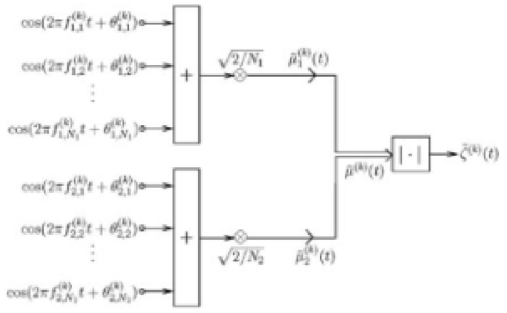
```
RUN ---> [phi]=MIMO elliptical parameters(25,90,90,0,91,180,0,0,'lpnm',1)
```

Non-isotropic scattering: ($K = 10$)

```
RUN --->[phi]=MIMO_elliptical_parameters(25,90,90,10,91,180,0,0,'lpnm',1)
```

UNCORRRLATED RAYLEIGH CHANELS

Generation of a set of multiple uncorrelated Rayleigh fading waveforms is important for the development and analysis of diversity scheme and MIMO channel with uncorrelated sub channel. improved sum-of-sinusoids simulation model is proposed for uncorrelated Rayleigh fading channels.


$\cos(2\pi f_{1,1}^{(k)}t + \theta_{1,1}^{(k)})$
$\cos(2\pi f_{1,2}^{(k)}t + \theta_{1,2}^{(k)})$
$\cos(2\pi f_{1,N_1}^{(k)}t + \theta_{1,N_1}^{(k)})$
$\cos(2\pi f_{2,1}^{(k)}t + \theta_{2,1}^{(k)})$
$\cos(2\pi f_{2,2}^{(k)}t + \theta_{2,2}^{(k)})$
$\cos(2\pi f_{2,N_2}^{(k)}t + \theta_{2,N_2}^{(k)})$
+
+
$\sqrt{2/N_1}$
$\sqrt{2/N_2}$
$\tilde{\mu}_1^{(k)}(t)$
$\tilde{\mu}_2^{(k)}(t)$
$\tilde{\mu}^{(k)}(t)$
$|\cdot|$
$\tilde{\zeta}^{(k)}(t)$

Structure of the SOS channel simulator generating Rayleigh fading waveforms

Implementation

- The program errorfunE2.m shows the effective of changing the angle-of-rotation with using the GMEDS1 for different value for Ni. In this case we use Ni = 20, 25,30.
- Program file ACF_Uncorr.m computes the autocorrelation function of the reference model in comparison to autocorrelation function of simulation model by using the MEDS with and the GMEDS1 with , the number of sinusoid Ni =20, and k = K= 4.
- Program file ACF_GAU_Uncorr.m computes the autocorrelation function of the complex Gaussian process of the reference model in comparison with corresponding autocorrelation function of simulated model designed by using the MEDS with and the GMEDS1 with and . the number of sinusoid $N1$ = $N2$ = N = 20 with K = 4.
- Program f_uncorr.m shows the simulation of uncorrelated Rayleigh fading waveform for several sample function which is selected K = 4, and $N1$ = $N2$=20, the phases are considered as outcomes of independent and identically distributed (i,i,d) random variable each having a uniform distribution over the interval $(0,2\pi]$.The discrete Doppler frequencies is defined by using GMEDS1.

Simulation m-files:

Usere can get Figure 10.2 to 10.6 by giving following inputs values to the perticular m-file.

errorfunE2.m RUN ------> errorfunE2([20;25;30],91,1,2,1)

ACF_Uncorr.m RUN-----> ACF_Uncorr(20,0,pi/120,-pi/120)

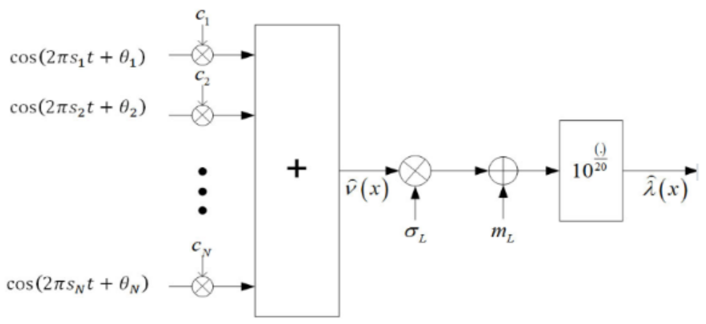
ACF_GAU_Uncorr.m RUN------> ACF_GAU_Uncorr(20,0,pi/160,pi/480)

f_uncorr.m RUN------> f_uncorr.m(20,20,4,91,1)

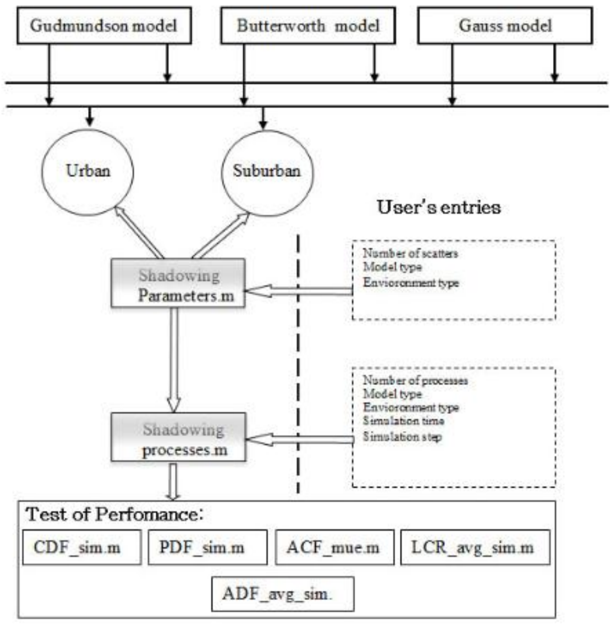
Go to home page

Spatial channel model for shadow fading

The 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.


$\cos(2\pi s_1 t + \theta_1)$
$\cos(2\pi s_2 t + \theta_2)$
$\cos(2\pi s_N t + \theta_N)$
c_1
c_2
c_N
+
$\hat{\nu}(x)$
σ_L
m_L
$10^{\frac{(\cdot)}{20}}$
$\hat{\lambda}(x)$

structure of a shadowing processes simulator



The diagram for the implementation of the spatial shadowing process

Simulation programs based on MEA:

Shadowing_parameters.m : Program for the computation of the gains and spatial frequencies of a spatial shadowing simulation using the method of equal areas (MEA).

Shadowing_processes.m: program for the simulation of deterministic log-normal process.

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

cdf_sim.m : Program for the computation of cumulative distribution functions F(r)

pdf_sim.m: Program for the computation of probability density functions p(z).

acf_mue.m

lcr_sim.m

lcr_avg_sim.m

adf_avg_sim

FREQUENCY HOPPING MOBILE RADIO CHANNEL

Frequency hopping techniques is used in the Global System for Mobile Communication (GSM) system. Frequency hopping combined with error protection techniques and interleaving is a very effective means in combating fading in mobile radio communication. Principle of frequency hopping in GSM is that the carrier frequency changes with every time division multiple accesses.

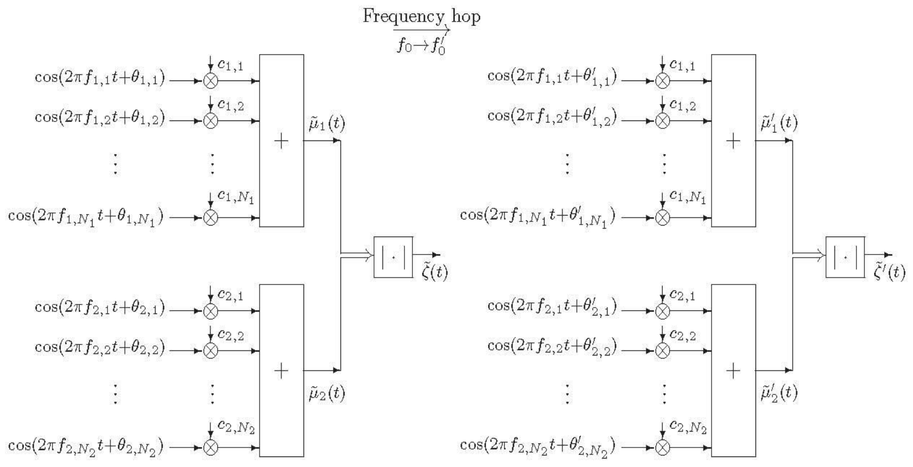

Frequency hop
$f_0 \rightarrow f_0'$
$\cos(2\pi f_{1,1}t+\theta_{1,1})$
$c_{1,1}$
$\cos(2\pi f_{1,2}t+\theta_{1,2})$
$c_{1,2}$
$\cos(2\pi f_{1,N_1}t+\theta_{1,N_1})$
c_{1,N_1}
$\tilde{\mu}_1(t)$
$|\cdot|$
$\tilde{\zeta}(t)$
$\cos(2\pi f_{2,1}t+\theta_{2,1})$
$c_{2,1}$
$\cos(2\pi f_{2,2}t+\theta_{2,2})$
$c_{2,2}$
$\cos(2\pi f_{2,N_2}t+\theta_{2,N_2})$
c_{2,N_2}
$\tilde{\mu}_2(t)$
$\cos(2\pi f_{1,1}t+\theta'_{1,1})$
$\cos(2\pi f_{1,2}t+\theta'_{1,2})$
$\cos(2\pi f_{1,N_1}t+\theta'_{1,N_1})$
$\tilde{\mu}'_1(t)$
$\tilde{\zeta}'(t)$
$\cos(2\pi f_{2,1}t+\theta'_{2,1})$
$\cos(2\pi f_{2,2}t+\theta'_{2,2})$
$\cos(2\pi f_{2,N_2}t+\theta'_{2,N_2})$
$\tilde{\mu}'_2(t)$

Figure 3.47: Frequency hopping Rayleigh fading channel simulator.

Implementation

- The function FH_channel_simulator.m is used to compute the simulated envelope $\hat{\zeta}(t)$ for both with and without frequency hopping to show its effect on the channel envelope $\hat{\zeta}(t)$.
- The function FH_ch_model_test.m is used to test the behaviour of channel with applying frequency hopping and compare between reference model and simulation model by using ACF and CCF, also show the effective of different values of $\chi = f_0' - f_0$.
- The option PLOT is provided to sketch the performance criteria, such as temporal ACF and space-time CCF as shown in Figure 3.50 to 3.53.

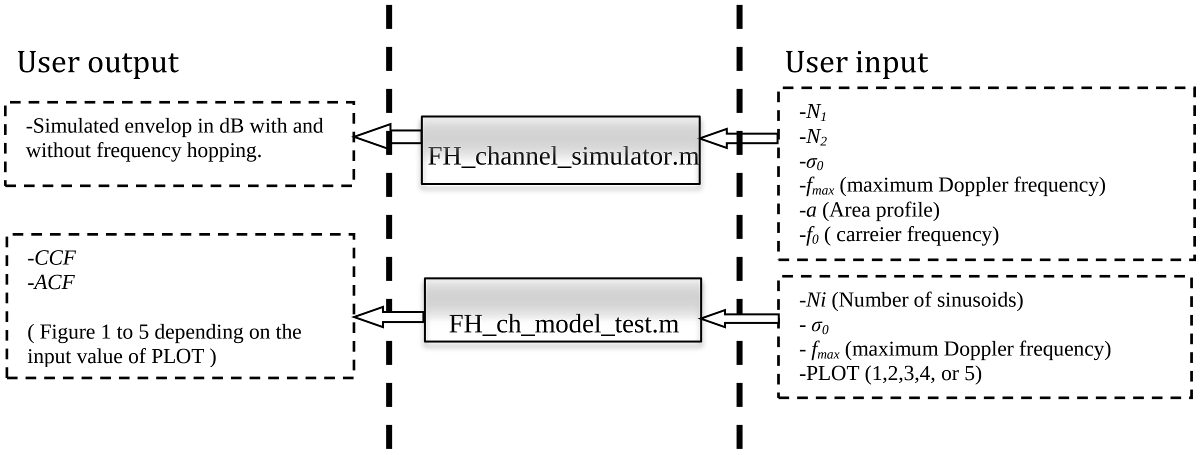
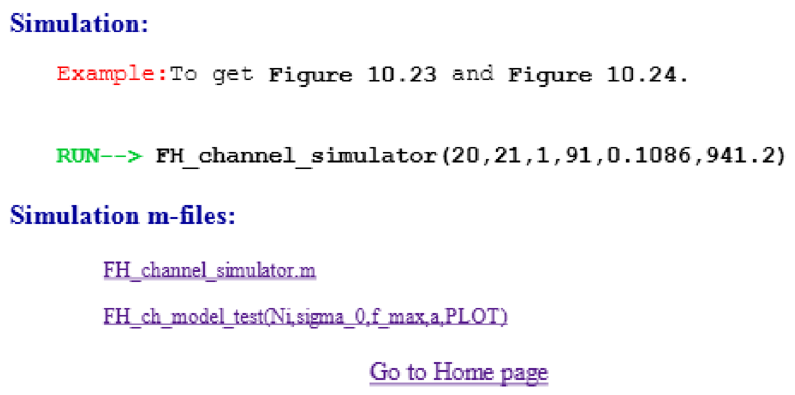

User output
-Simulated envelop in dB with and without frequency hopping.
-CCF
-ACF
(Figure 1 to 5 depending on the input value of PLOT)
FH_channel_simulator.m
FH_ch_model_test.m
User input
$-N_1$
$-N_2$
$-\sigma_0$
$-f_{max}$ (maximum Doppler frequency)
-a (Area profile)
$-f_0$ (carreier frequency)
-Ni (Number of sinusoids)
$- \sigma_0$
$- f_{max}$ (maximum Doppler frequency)
-PLOT (1,2,3,4, or 5)

Figure 3.48: The diagram for the implementation of a Frequency hopping channel simulator.



Simulation:

Example:To get **Figure 10.23** and **Figure 10.24.**

```
RUN--> FH_channel_simulator(20,21,1,91,0.1086,941.2)
```

Simulation m-files:

FH_channel_simulator.m

FH_ch_model_test(Ni,sigma_0,f_max,a,PLOT)

Go to Home page

APPENDIX 2

MATLAB PROGRAMS FOR PARAMETERS COMPUTATION METHODS

MATLAB-PROGRAMS of Parameters Computation Methods for Frequency-Non-Selective Channels

Parameter Computation Methods for SOS Channel Simulators

```
%--------------------------------------------------------------------
% 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*cos(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)
        set(0,'DefaultAxesFontSize',18);
       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',18);
       title('i=1')
       xlabel('  Frequency, f (Hz)','HorizontalAlignment',...
       'center','FontName','Helvetica','FontSize',...
           24,'Interpreter','latex')

       ylabel('PSD,  $ \rm{S_{\mu_i\mu_i}\mbox(f)}$','FontName',...
           'Helvetica','FontSize',24,'Interpreter','latex')
       subplot(2,3,2)
        set(0,'DefaultAxesFontSize',18);
       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',18);
       title('i=2')
       xlabel('  Frequency, f (Hz)','HorizontalAlignment',...
       'center','FontName','Helvetica','FontSize',...
           24,'Interpreter','latex')

       ylabel('PSD,  $\rm{ S_{\mu_i\mu_i}\mbox(f)}$','FontName',...
           'Helvetica','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,r_mm,'k-',tau,r_mm_tilde1,'k-.')
       xlim([0 tau_max])
       set(0,'DefaultAxesFontSize',18);
       title('i=1')
       xlabel('Time difference, $\rm{\tau}$ (s)','HorizontalAlignment',...
       'center','FontName','Helvetica','FontSize',...
           24,'Interpreter','latex')
       ylabel({'ACF, $\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$'},'FontName',...
           'Helvetica','FontSize',24,'Interpreter','latex')
       r_mm_tilde2=acf_mue(f_i_n(:,2),c_i_n(:,2),tau);
       subplot(2,3,5)
        %set(0,'DefaultAxesFontSize',18);
       plot(tau,r_mm,'k-',tau,r_mm_tilde2,'k-.')
       xlim([0 tau_max])
       set(0,'DefaultAxesFontSize',18);
       title('i=2')
      xlabel('Time difference, $\rm{\tau}$ (s)','HorizontalAlignment',...
       'center','FontName','Helvetica','FontSize',...
           24,'Interpreter','latex')
        ylabel({'ACF, $\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$'},'FontName',...
           'Helvetica','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',18);
       title('i=1,2')
       xlabel('  Frequency, f (Hz)','HorizontalAlignment',...
       'center','FontName','Helvetica','FontSize',...
           24,'Interpreter','latex')
       ylabel('PSD, $ \rm{S_{\mu_i\mu_i}\mbox(f)}$','FontName',...
           'Helvetica','FontSize',24,'Interpreter','latex')
       subplot(2,3,6)
        set(0,'DefaultAxesFontSize',18);
       plot(tau,2*r_mm,'k-')
       plot(tau,r_mm_tilde1+r_mm_tilde2,'k-.')
       xlim([0 tau_max])
       set(0,'DefaultAxesFontSize',18);
       title('i=1,2')
        xlabel('Time difference, $\rm{\tau}$ (s)','HorizontalAlignment',...
       'center','FontName','Helvetica','FontSize',...
           24,'Interpreter','latex')
        ylabel({'ACF, $\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$'},'FontName',...
           'Helvetica','FontSize',24,'Interpreter','latex')
   elseif METHOD=='sp_j'|METHOD=='rm_j'
       subplot(1,2,1)
       %figure(1)
        set(0,'DefaultAxesFontSize',18);
       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',18);
          xlabel('Frequency, f (Hz)','HorizontalAlignment',...
       'center','FontName','Helvetica','FontSize',...
           24,'Interpreter','latex')
              ylabel('PSD, $\rm{ S_{\mu_i\mu_i}\mbox(f)}$','FontName',...
           'Helvetica','FontSize',24,'Interpreter','latex')
       tau_max=2*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)
      % figure (2)
       plot(tau,r_mm_tilde,'k-.',tau,r_mm,'k-')
       h=legend('$\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$ (Simulation model)',...
           '$r_{\mu_i\mu_i}(\tau)$ (Reference model)');
       set(h,'FontSize',22,'Interpreter','latex');
       xlim([0 tau_max])
       set(0,'DefaultAxesFontSize',18);
       xlabel('Time difference, $\rm{\tau}$ (s)','HorizontalAlignment',...
       'center','FontName','Helvetica','FontSize',...
           24,'Interpreter','latex')
       ylabel({'ACF, $\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$'},'FontName',...
           'Helvetica','FontSize',24,'Interpreter','latex')
   else
       subplot(1,2,1)
       set(0,'DefaultAxesFontSize',18);
       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',18);
       %set(0,'DefaultAxesFontSize',18);
       xlabel(' Frequency,  f (Hz)','HorizontalAlignment',...
       'center','FontName','Helvetica','FontSize',...
           24,'Interpreter','latex')
    ylabel('PSD, $\rm{ S_{\mu_i\mu_i}\mbox(f)}$','FontName',...
           'Helvetica','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,r_mm_tilde,'k-.',tau,r_mm,'k-')
       h=legend('$\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$ (Simulation model)',...
           '$\rm{r_{\mu_i\mu_i}(\tau)}$ (Reference model)');
       set(h,'FontSize',22,'Interpreter','latex');
       %     plot(tau*f_max,r_mm,'k-',tau*f_max,r_mm_tilde,'k--')
       xlim([0 tau_max])
       set(0,'DefaultAxesFontSize',18);
       xlabel('Time difference, $\rm{\tau}$ (s)','HorizontalAlignment',...
       'center','FontSize',...
           24,'Interpreter','latex')
       ylabel({'ACF, $\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$'},'FontSize',...
           24,'Interpreter','latex')
   end
end
```

```
%----------------------------------------------------------------------
% fun_Jakes.m
%----------------------------------------------------------------------
% Computation of the error function according to Eq.(5.61) 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 Jo;
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',18);
     xlabel(' Frequency,  f (Hz)','HorizontalAlignment',...
        'center','FontName','Helvetica','FontSize',...
            24,'Interpreter','latex')
     ylabel('PSD, $\rm{ S_{\mu_i\mu_i}\mbox(f)}$','FontName',...
            'Helvetica','FontSize',24,'Interpreter','latex')
    tau_max=N_i/(K*kappa_c*f_c);
    tau=linspace(0,tau_max,80);
    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,r_mm_tilde,'k--',tau,r_mm,'k-')
    h=legend('$\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$ (Simulation model)',...
            '$\rm{r_{\mu_i\mu_i}(\tau)}$ (Reference model)');
        set(h,'FontSize',22,'Interpreter','latex');
    xlim([0 tau_max])
    xlabel('Time difference, $\rm{\tau}$ (s)','HorizontalAlignment',...
        'center','FontName','Helvetica','FontSize',...
            24,'Interpreter','latex')
        ylabel({'ACF, $\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$'},'FontName',...
            'Helvetica','FontSize',24,'Interpreter','latex')
 end

 %----------------------------------------------------------------------
 % fun_Gauss.m
 %----------------------------------------------------------------------
 % Computation of the error function according to Eq.(5.61) 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));
```

Parameter Computation Methods for SOC channel Simulators

```
%----------------------------------------------------------------------
% parameter_soc.m
%----------------------------------------------------------------------
% Program for the computation of the discrete Doppler frequencies,
% Doppler coefficients and Doppler phases for the Sum of Cisoids models
%
%
% Used m-files: GMEA_parameter.m, LPNM3_VM_PDF_parameter.m
%----------------------------------------------------------------------
% [fn,Cn,An]=parameter_soc(METHOD,N_i,sigma_0,f_max)
%----------------------------------------------------------------------
% Explanation of the input parameters:
%
% METHOD:
% |----------------------------------------------|--------------------|
% | Methods for the computation of the discrete  |      Input         |
% | Doppler frequencies and Doppler coefficients |                    |
% |----------------------------------------------|--------------------|
% |----------------------------------------------|--------------------|
% | Extended method of Exact Doppler Spread      |     'EMED'         |
% |----------------------------------------------|--------------------|
% | Lp-norm method (LPNM)                        |     'LPNM'         |
% |----------------------------------------------|--------------------|
% | Generalized method of Equal Areac            |     'GMEA'         |
% |----------------------------------------------|--------------------|
%
% N_i: number of harmonic functions
% sigma_0: Square root value of average power of the real deterministic
% Gaussian process mu_i(t)
% f_max: maximum Doppler frequency


function [fn,Cn,An]=parameter_soc(METHOD,N_i,sigma_0,f_max)

if nargin<4,
   error('Not enough input parameters')
end
N=N_i;
fmax=f_max;

% Extended method of Exact Doppler Spread (EMEDS)
if      METHOD == 'EMED',
        n=(1:N)';
        fn=fmax*sin(2*pi/N*(n-1/4));
        cn=sigma_0*sqrt(2/(N))*ones(size(fn));
        figure (1)
        tau_max=N/(1*fmax);
        tau=linspace(0,tau_max,500);
        r_mm=sigma_0^2*besselj(0,2*pi*fmax*tau);
        r_mm_tilde=acf_mue(fn,cn,tau);
         set(0,'DefaultAxesFontSize',18);
                plot(tau,r_mm_tilde,'k-.',tau,r_mm,'k-')
        h=legend('$\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$ (Simulation model)',...
            '$\rm{r_{\mu_i\mu_i}(\tau)}$ (Reference model)');
```

```
       set(h,'FontSize',22,'Interpreter','latex');
       %     plot(tau*f_max,r_mm,'k-',tau*f_max,r_mm_tilde,'k--')
       xlim([0 tau_max])
             xlabel('Time difference, $\rm{\tau}$
(s)','HorizontalAlignment',...
       'center','FontSize',...
           24,'Interpreter','latex')
       ylabel({'ACF, $\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$'},'FontSize',...
           24,'Interpreter','latex')


 % Generalized method of Equal Areac(GMEA)
elseif METHOD=='GMEA'
        N=N_i;
        fmax=f_max;
        SAVE_data = 0;
        vec_alpha_0 = [0]; vec_kappa = [0];
       [fn, Cn, An] = GMEA_parameter( N, fmax, sigma_0,...
        SAVE_data, vec_alpha_0, vec_kappa);
           figure (1)
       tau_max=N/(1*fmax);
       tau=linspace(0,tau_max,500);
       r_mm=sigma_0^2*besselj(0,2*pi*fmax*tau);
       r_mm_tilde=acf_mue(fn,Cn,tau);
               plot(tau,r_mm_tilde*2,'k-.',tau,r_mm,'k-')
       h=legend('$\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$ (Simulation model)',...
           '$\rm{r_{\mu_i\mu_i}(\tau)}$ (Reference model)');
       set(h,'FontSize',22,'Interpreter','latex');
       %     plot(tau*f_max,r_mm,'k-',tau*f_max,r_mm_tilde,'k--')
       xlim([0 tau_max])
       set(0,'DefaultAxesFontSize',18);
       xlabel('Time difference, $\rm{\tau}$ (s)','HorizontalAlignment',...
       'center','FontSize',...
           24,'Interpreter','latex')
       ylabel({'ACF, $\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$'},'FontSize',...
           24,'Interpreter','latex')

% Lp-Norm Methd (LPNM)
elseif METHOD=='LPNM',
        N=N_i;
        fmax=f_max;
        p = 2;
        SAVE_data = 0;
        vec_alpha_0 = [0]; vec_kappa = [0];
        [fn, Cn]=LPNM3_VM_PDF_parameter(N, fmax, p, sigma_0, SAVE_data,...
             vec_alpha_0, vec_kappa);
                    figure (1)
       tau_max=N/(1*fmax);
       tau=linspace(0,tau_max,500);
       r_mm=sigma_0^2*besselj(0,2*pi*fmax*tau);
       r_mm_tilde=acf_mue(fn,Cn,tau);
               plot(tau,r_mm_tilde*2,'k-.',tau,r_mm,'k-')
       h=legend('$\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$ (Simulation model)',...
           '$\rm{r_{\mu_i\mu_i}(\tau)}$ (Reference model)');
       set(h,'FontSize',22,'Interpreter','latex');
       %     plot(tau*f_max,r_mm,'k-',tau*f_max,r_mm_tilde,'k--')
       xlim([0 tau_max])
       set(0,'DefaultAxesFontSize',18);
       xlabel('Time difference, $\rm{\tau}$ (s)','HorizontalAlignment',...
       'center','FontSize',...
           24,'Interpreter','latex')
       ylabel({'ACF, $\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$'},'FontSize',...
           24,'Interpreter','latex')
else
    error('method is unknown')
end;

end



%----------------------------------------------------------------------
% GMEA_parameter
%----------------------------------------------------------------------
% Program for the computation of the discrete Doppler frequencies,
% Doppler coefficients and Doppler phases by using the GMEA method
%
% Used m-files: parameter_soc.m
%--------------------------------------------------------------------
%function [fn, Cn, An] = GMEA_parameter( N, fmax, sigma_0, SAVE_data,
vec_alpha_0, vec_kappa)
%--------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: number of harmonic functions
% f_max: maximum Doppler frequency
% sigma_0_2: average power of the real deterministic Gaussian
%            process mu_i(t)
% SAVE_data: data will be save if SAVE_data==1
% vec_alpha_0:[0 0 0 0 30 90]
% vec_kappa:[0 5 20 10 10 10]
%----------------------------------------------------------------------
function [fn, Cn, An] = GMEA_parameter( N, fmax, sigma_0, SAVE_data,
vec_alpha_0, vec_kappa)

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
tic

if nargin == 2
   sigma_0 = 1;
   SAVE_data = 0;
   vec_alpha_0 = [0 0 0 0 30 90]; vec_kappa = [0 5 20 10 10 10];
end
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
if nargin == 3
   SAVE_data = 0;
   vec_alpha_0 = [0 0 0 0 30 90]; vec_kappa = [0 5 20 10 10 10];
end
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
if nargin == 4
   vec_alpha_0 = [0 0 0 0 30 90]; vec_kappa = [0 5 20 10 10 10];
end
%%%%% Definition of variables and functions of the program
P = length(vec_alpha_0);
```

```
%------------------------- Variables to evaluate the numerical integral -----
resol = pi/1600;
angle_axis =resol:resol:pi;
Int_VM = zeros(1,length( angle_axis ));    %Define the numerical integral for
pdf of Von Mises
%------------------------ Variables of simulation model ----------------------
----------------
fn = zeros(P,N);  %Defines a matrix containing the Doppler freq.
Cn = zeros(P,N);  %Defines a matrix containing the Doppler coeff.
An = zeros(P,N);  %Defines a matrix containing the deterministic AOAs.
%%%%%%%%%% Section to evaluate the numerical integral for dif. alpha_n values
options = optimset('TolFun',1*10^(-16));
for p = 1:P
    alpha_0 = vec_alpha_0(p)*pi/180;
    kappa = vec_kappa(p);
    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    an = zeros(1,N);
    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    for n = 1:N                                 %Cycle to calculate the
alphas for each sinusoid..
        alpha_nN = (1/(2*N))*(n-0.5);
        [an(n),fval,exitflag] = fzero( @(alpha_n) NumInt_VM_EvenPDF( alpha_0,
kappa, alpha_nN, alpha_n), [pi/1e8 pi], options );
    end
    fi = fmax*cos(an);
    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    fn(p,:) = sort(fi,'descend');
    Cn(p,:) = sqrt(sigma_0/N)*ones(1,N);
    An(p,:) = an;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if SAVE_data == 1
    parameters_VM_PDF = [vec_alpha_0; vec_kappa];
    save( sprintf('E:/work/RIM-
Jrnl/DataFiles/Parameters_M=%d_VM_N=%d_fmax=%d',1,N,fmax), ...
          'fn', 'Cn', 'An', 'parameters_VM_PDF');
    display(['       * ' sprintf('Data have been saved -> RIM-
Jrnl/DataFiles/Parameters_M=%d_VM_N=%d_fmax=%d',1,N,fmax) ])
end
%%%%% NOTE: "M" denotes the method code for other programs (1 -> MMEA)%%
t1 = toc;
disp(['  * Simulation time for the program "MMEA_VM_PDF_parameter" = '
num2str(t1) ' secs '])

function Int_VM = NumInt_VM_EvenPDF(alpha_0 , kappa, alphanN, alpha_n)
    %------------------------------------------------------------------------
    Kappa = num2str(kappa);                %Variable Kappa format to
evaluate the integral character.
    Alpha_0 = num2str(alpha_0);      %Variable alpha_0 format to evaluate the
integral character.
    %------------------------------------------------------------------------
    pdfVM = [ '( exp(' Kappa '*( cos(alpha - ' Alpha_0 '))) + exp(' Kappa '*(
cos(alpha + ' Alpha_0 ')))) ./(4*pi*besseli(0,' Kappa '))' ]; %even part of
the VonMises pdf format character.
    %------------------------------------------------------------------------
    Int_VM = alphanN - quadl( inline( pdfVM ), 0, alpha_n);
return
```

```
%-----------------------------------------------------------------------
% LPNM3_VM_PDF_parameter.m
%
% Program for the computation of the discrete Doppler frequencies,
% Doppler coefficients and Doppler phases by using the LPNM III.
%
% Used m-files: acf_mue.m, RIM_VM_PDF_parameter.m, rice_pdf_SOCv6.m
%----------------------------------------------------------------------
% [fn, Cn] = LPNM3_VM_PDF_parameter(N, fmax, p, sigma_0, SAVE_data,
% vec_alpha_0, vec_kappa)
%----------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: number of harmonic functions
% f_max: maximum Doppler frequency
% p: p value of the LPNM method
% sigma_0: average power of the real deterministic Gaussian
%            process mu_i(t)
% SAVE_data: save the data if SAVE_data==1
% vec_alpha_0:If user not define the vec_alpha_0 then program will use
%             vec_alpha_0 input as [0 0 0 0 30 90]
% vec_kappa:If user not define the vec_kappa then program will use
%           vec_kappa input as [0 5 20 10 10 10]
%----------------------------------------------------------------------

function [fn, Cn] = LPNM3_VM_PDF_parameter(N, fmax, p, sigma_0,...
    SAVE_data, vec_alpha_0, vec_kappa)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if nargin < 2
       N = 10; fmax = 91; p = 2; sigma_0 = 1; SAVE_data = 0; ...
           vec_alpha_0 = [0 0 0 0 30 90]; vec_kappa = [0 5 20 10 10 10];
    elseif nargin == 2
       p = 2; sigma_0 = 1; SAVE_data = 0; vec_alpha_0 = [0 0 0 0 30 90];...
           vec_kappa = [0 5 20 10 10 10];
    elseif nargin == 3
       sigma_0 = 1; SAVE_data = 0; vec_alpha_0 = [0 0 0 0 30 90];
           vec_kappa = [0 5 20 10 10 10];
    elseif nargin == 4
       SAVE_data = 0; vec_alpha_0 = [0 0 0 0 30 90];
       vec_kappa = [0 5 20 10 10 10];
    elseif nargin == 5
       vec_alpha_0 = [0 0 0 0 30 90]; vec_kappa = [0 5 20 10 10 10];
    end
    %----------------------------------------------------------------------
    if exist('fminunc')~=2
       disp([' =====> This method requires the Optimization Toolbox !!'])
    end
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    HayInitialCond = 0;
%Computation of the initial Doppler frequencies (by using the MMEA)%
if HayInitialCond == 1
load( sprintf('E:/work/RIM-
Jrnl/DataFiles/Parameters_M=%d_VM_N=%d_fmax=%d',...
    3,N,fmax) );
        An = An;
        Cn = Cn;
    else
```

```
[~, Cn An] = RIM_VM_PDF_parameter(N, fmax, sigma_0, 0, vec_alpha_0,...
    vec_kappa);
%Calcula las frec. Doppler de acuedo al MMEA (parametros iniciales).
    end
%%% Computation of the simulator's parameters by using the LPNM  %%%%%%%
tic              %Reinitializes the counter to measure the simulation time.
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
P = length(vec_alpha_0);
%~~~~~~~~~ Parameters to evaluate the standard ACF Lp~~~~~~~~~~~~~~~~~~-
Tau_max = N/(fmax*4);          % Maximum value to minimize the standard LP.
tau=linspace(0,Tau_max,1000);      % Time axis to assess the standard Lp.
tau_axis = linspace(0,N/(fmax*2),2000);
% Time axis to plot the ACFs.

%~~~~~~~~~~ Parameters to evaluate the standard PDF Lp~~~~~~~~~~~~~~~~~-
z_axis = linspace(0,4,100);
%~~~~~~~~~~ Standard weights to assess the joint Lp~~~~~~~~~~~~~~~~~~~~~
W1 = 1/4;                    % Weight error of the ACF
W2 = 3/4;                    % Weight PDF error
%---------------------------------------------------------------------------
p_ref = (2*z_axis/(sigma_0^2)).*exp( -(z_axis/sigma_0).^2 );
%---------------------------------------------------------------------------
for caso = 1:P
        %-------------------------------------------------------------------
        alpha_0 = vec_alpha_0(caso);
        kappa = vec_kappa(caso);
        %-------------------------------------------------------------------
        r_ref = sigma_0^2 * besseli(0, sqrt( kappa^2 - (2*pi*fmax*tau).^2 +
...
        j*4*pi*kappa*fmax*tau*cos(alpha_0*pi/180) ) ) / (besseli(0,kappa));
        %-------------------------------------------------------------------
        Pinit = [An(caso,:), Cn(caso,:)];
        %-------------------------------------------------------------------
        save dataLPNM N p fmax r_ref tau p_ref z_axis W1 W2
        % Save data to be used in the function.
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        Popt = fminsearch( @(Popt) Function_LpNorm(Popt), Pinit );
        Aopt = Popt(1:N);
        Copt = Popt(N+1:end);
        fi = fmax*cos(Aopt);
        %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~%
        fn(caso,:) = (fi);
        Cn(caso,:) = (Copt);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if SAVE_data == 1
    parameters_VM_PDF = [vec_alpha_0; vec_kappa];
    save( sprintf('E:/work/RIM-
Jrnl/DataFiles/Parameters_M=%d_VM_N=%d_fmax=%d_p=%d',6,N,fmax,p), ...
          'fn', 'Cn', 'parameters_VM_PDF');
    display(['*' sprintf('Data have been saved -> RIM-
Jrnl/DataFiles/Parameters_M=%d_VM_N=%d_fmax=%d_p=%d',6,N,fmax,p) ])
end
%%%%% NOTE: "M" denotes the method code for other programs (1 -> MMEA)
x = toc;
disp(['  * Simulation time for the program "LPNM3_VM_PDF_parameter" = ' ...
    num2str( x ) ' secs, ' num2str( x/60 ) 'min' ])
```

```
%~~~~~~~~~~~~~~~~~~~ PLOTS the RESULTS ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~%
        for caso = 1:P
        %------------------------------------------------------------------
        alpha_0 = vec_alpha_0(caso);
        kappa = vec_kappa(caso);
        %------------------------------------------------------------------
        r_ref = besseli(0, sqrt( kappa^2 - (2*pi*fmax*tau_axis).^2 + ...
        j*4*pi*kappa*fmax*tau_axis*cos(alpha_0*pi/180) ) ) / ...
        (besseli(0,kappa));
        r_mm_tilde=acf_mue( fn(caso,:), Cn(caso,:), tau_axis);
        %-------------
        figure(ceil(2*caso/P))
        hold on
        plot(tau_axis*fmax,abs(r_ref),'k')
        hold on
        plot(tau_axis*fmax,abs(r_mm_tilde),'b')
        end
        figure(1); axis([0 (1+N/4) 0 1.2]); hold off;
        figure(2); axis([0 (1+N/4) 0 1.2]); hold off;
        figure(3)
        hold on
        plot(z_axis, p_ref,'k')
        figure(4)
        hold on
        plot(z_axis, p_ref,'k')
        %------------------------------------------------------------------
        for caso = 1:P
        %------------------------------------------------------------------
           p_emp = rice_pdf_SOCv6(0, N, fmax, Cn(caso,:), z_axis);
           % pdf of the simulation model (theory)
           figure(2+ceil(2*caso/P))
           plot(z_axis, p_emp,'b')
        end
        figure(3); axis([0 z_axis(end) 0 1.4]); hold off;
        figure(4); axis([0 z_axis(end) 0 1.4]); hold off;
return
function F = Function_LpNorm(P_ancn)
    %----------------------------------------------------------------------
    load dataLPNM
    Ai = P_ancn(1:N);
    cn = P_ancn(N+1:end);
    %----------------------------------------------------------------------
    r_emp = zeros(1,length(tau));
    for n = 1:N
        r_emp = r_emp + (cn(n)^2)*exp( j*2*pi*fmax*cos(Ai(n))*tau );
    end
    %----------------------------------------------------------------------
    p_emp = rice_pdf_SOCv6(0, N, fmax, cn, z_axis);
    %pdf of the simulation model (theory) % 1 FUNCTION REQUIRED %
    %----------------------------------------------------------------------
    F =    W1 *( (1/tau(end)) * trapz(tau, abs( r_ref - r_emp).^p ) )^(1/p)
...
        +  W2 *( trapz(z_axis, abs( p_ref - p_emp).^p ) )^(1/p);
return
```

MATLAB-PROGRAMS of Parameters Computation Methods for Frequency Selective Channel Simulators

```
%-----------------------------------------------------------------------
% 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,
    figure;
    text(2,0.4,{['COST 207, ',...
       upper(Area),', ',upper(method)]},...
    'interpreter','latex','FontSize',22,'Margin',100000);
    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),'k--');

    set(0,'DefaultAxesFontSize',18)
%     title(['COST 207, ',...
%        upper(Area),', ',upper(method),',
L=',num2str(L)],'Interpreter','latex','FontSize',24);
    legend({['Reference model',],['Simulation model, with
L=',num2str(L)]},...

'FontName','Arial','Location','NorthEast','FontSize',22,'Interpreter','latex'
);
    xlabel('Frquency seperation, $\upsilon$\
(MHz)','Interpreter','latex','FontSize',24);
    ylabel('Absolute value of the 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 3

MATLAB PROGRAM 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)
% (see Fig. 4.3)[2].
%
% 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)+...
         1j*(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


%-------------------------------------------------------------------
% Rice_Proc_Jakes.m
%-------------------------------------------------------------------
% Program for the simulation of deterministic Rice
% processes as shown in Figure 4.4[2].parameters calculate based on
% parameter_Jakes.m
%
% Used m-files: parameter_Jakes.m, Mu_i_t.m
```

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

% xi_t=Rice_Proc_Jakes(N_1,N_2,sigma_1_2,sigma_2_2,kappa_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)
% 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'        |
% |-----------------------------------------------|-------------------|
% rho: amplitude of the LOS component m(t)
% f_rho: Dopplerfrequency of the LOS componaent 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 Rice process xi(t),
%       if PLOT==1

function
xi_t=Rice_Proc_Jakes(N_1,N_2,sigma_1_2,sigma_2_2,METHOD,rho,f_rho,...
    theta_rho,f_max,T_s,T_sim,K,PLOT)
if nargin==11,
   PLOT=0;
end
[f1,c1,th1]=parameter_Jakes(METHOD,N_1,sigma_1_2,f_max,'rand',K,0);
%c1=c1/sqrt(2);
```

```
[f2,c2,th2]=parameter_Jakes(METHOD,N_2,sigma_2_2,f_max,'rand',K,0);
%c2=c2/sqrt(2);
save f1;
save c1;

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)+...
         1j*(Mu_i_t(c2,f2,th2,T_s,T_sim)+...
         rho*sin(theta_rho)));
pha_t=angle(Mu_i_t(c1,f1,th1,T_s,T_sim)+...
         +rho*cos(theta_rho)+...
         1j*(Mu_i_t(c2,f2,th2,T_s,T_sim)+...
         rho.*sin(arg)));
save pha_t;
save xi_t;

if PLOT==1,
    figure(1)
   plot(t,20*log10(xi_t),'b-')
    xlabel({'t (s)'},'FontName','Arial','FontSize',16)
   ylabel({'20 log(x_i(t))'},'FontName','Arial','FontSize',16)
   figure(2)
   plot(t,pha_t,'k-')
   xlabel({'t (s)'},'FontName','Arial','FontSize',16)
   ylabel({'phase of the (x_i(t))'},'FontName','Arial','FontSize',16)
end

%--------------------------------------------------------------------
% gen_Rice_proc.m ---------------------------------------------------
%
% Program for the simulation of deterministic generalized Rice
% processes (see Fig. 6.29)[2].
%
% 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)
% f_rho: frequency 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),'k-')
  set(0,'DefaultAxesFontSize',18);
   xlabel({'t (s)'},'FontName','Arial','FontSize',24)
   ylabel({'20 log(x_i(t))'},'FontName','Arial','FontSize',24)
end

%--------------------------------------------------------------------
% Suzuki_Type_I.m
%--------------------------------------------------------------------
% Program for the simulation of deterministic extended Suzuki
% processes of Type I (see Fig. 6.9[2].
%
% 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',18)
   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 (see Fig. 6.23)[2].
%
% 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 frequency
% 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',18)
   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[2].
%
% 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',18)
   xlabel('t (s)')
   ylabel('20 log rho(t)')
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
```

MATHLAB –PROGRAMS of Non-Stationary Land Mobile Satellite Channel Simulator

```
%--------------------------------------------------------------------------
% simu.m
%--------------------------------------------------------------------------
% program to initiate the parameters for the extended Suzuki processes and
% generate both simulation signal for both state (due to M=2) under
% relevant reference model parameters. User can use this program to
% simulate under Urban or suburban area condition by changing the 'fname'.
% fname consist with reference model parameters.
%
% Used m-files:simu_fun.m,
%--------------------------------------------------------------------------
```

```
clear all;
fname='city25';
load(fname)

fabtast=500;
%x:Parameter vector of the extended Suzuki processes of Type 1.
eta1=simu_fun(x(1:7),fabtast);
eta2=simu_fun(x(8:14),fabtast);
save(fname,'eta1','eta2','-append')


%-------------------------------------------------------------------------
% simu_fun.m
%-------------------------------------------------------------------------
% Program for the NRWLMS channel simulator.
% simulated fading envelop obtained by using the dynamic (2+2)-state model
% with embedded extended Suzuki processes of Type 1.
%
% Used m-files: lcrfun.m,f2.m,simu.m
%-------------------------------------------------------------------------
% eta=simu(x,fabtast)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
% x: Input parameter vector of the extended Suzuki processes of Type 1.
    % psi_o  =x(1);
    % rho    =x(2);
    % m3     =x(3);
    % sigma3 =x(4);
    % kappa0 =x(5);
    % frho   =x(6);
    % fmax   =x(7);
%
% fabtast:level

function eta=simu_fun(x,fabtast)
tic
iter=1e6;   %Number of iteration point
berechn=0;  % LCR, CDF and ADF calculated? (1 = yes)

if exist('N1')==0
   N1=15;
end
if exist('N2')==0
   N2=15;  % or 16
end
if exist('N3')==0
   N3=15;
end

kappa_grenz=1e-3; %Minimum for kappa (kappa if <kappa_grenz => kappa = 0)

psi_o=x(1);
rho=x(2);
m3=x(3);
sigma3=x(4);
kappa0=x(5);
```

```
frho=x(6);
fmax=x(7);

sigma0=sqrt(2*x(1)/(1+2/pi*asin(x(5))));

%fabtast=40*fmax; % Sampling
thetarho=0;
kappac=20;
sigmac=fmax/kappac/sqrt(2*log(2));

N1mod=N1;
N2mod=ceil(N2*pi/2/asin(kappa0));

n1=[1:N1];
n2=[1:N2];

f1=fmax*sin(pi/(2*N1mod)*(n1-0.5));
f2=fmax*sin(pi/(2*N2mod)*(n2-0.5));

c1=sigma0*sqrt(1/N1mod);
c2=sigma0*sqrt(1/N2mod);

n3=[1:N3];

% Calculation of f3
% f3=erfinv((n3(1:N3-1)-0.5)/N3);
f3=erfinv((n3(1:N3-1)-0.5)/N3)*(sqrt(2)*sigmac);
f3(N3)=sqrt(N3*sigmac^2-sum(f3(1:N3-1).^2));
c3=sqrt(2/N3);

% theta: uniformly distributed over the interval [0,2 * pi)
theta1=rand(1,N1)*2*pi;theta2=rand(1,N2)*2*pi;theta3=rand(1,N3)*2*pi;
% theta1=(n1-1)*2*pi/N1;theta2=(n2-1)*2*pi/N2;theta3=(n3-1)*2*pi/N3;

if kappa0<=kappa_grenz,
   c2=0;
   f2=0;
   theta2=0;
end;

%t=[0:iter-1]/fabtast;

% Calculation of My1 My2 My3
clear myrho my3 eta
myrho=zeros(1,iter);
my3=zeros(1,iter);
klaenge=10000;
for k=1:iter/klaenge
   t=([0:klaenge-1]+(k-1)*klaenge)/fabtast;
   kk=[1:klaenge]+(k-1)*klaenge;
   myrho(kk)=sqrt((sum(c1*cos((2*pi*f1')*t + theta1'*ones(1,klaenge))) +...
      sum(c2*cos((2*pi*f2')*t + theta2'*ones(1,klaenge)))+...
      rho*cos(2*pi*frho*t+thetarho)).^2+...
      (sum(c1*sin(2*pi*(f1')*t + theta1'*ones(1,klaenge))) -...
      sum(c2*sin(2*pi*(f2')*t + theta2'*ones(1,klaenge)))+...
      rho*sin(2*pi*frho*t+thetarho)).^2);
end
for k=1:iter/klaenge
   t=([0:klaenge-1]+(k-1)*klaenge)/fabtast;
   kk=[1:klaenge]+(k-1)*klaenge;
   my3(kk)=sigma3*sum(c3*cos(2*pi*(f3')*t+theta3'*ones(1,klaenge)))+m3;
end

 %  Calculation of MyRho
 %  myrho1=my1+rho*cos(2*pi*frho*t+thetarho); %Realteil von MyRho
 %  myrho2=my2+rho*sin(2*pi*frho*t+thetarho); % Imaginaerteil von MyRho

% Amplitude of Eta(t)
eta=myrho.*exp(my3);
clear my3 myrho

% Storing the data
save sim_ant1 eta fabtast iter kappac

clear N1 N2 N3 c1 c2 c3 theta1 theta2 theta3
clear f1 f2 f3
clear N1mod N2mod n1 n2 n3 k kk klaenge iter thetarho frho
clear sigmac myrho1 myrho2 my1 my2 my3
clear kappa0 sigma0 sigma3 m3 rho psi_o t
clear kappa_grenz fmax


if berechn==1
% Calculation of LCR, F+ und ADF
[rsim,lcr_sim]=lcrfun(eta,length(R),fabtast,R);
[rsim,cdf_sim]=f2(eta,length(R),fabtast,R);
adf_sim=(1-cdf_sim)./lcr_sim;

% Storing the data (LCR, F+, ADF)
   save ant1_sim rsim lcr_sim cdf_sim adf_sim fabtast
end

toc
 %---------------------------------------------------------------------------
% mzust.m
%---------------------------------------------------------------------------
% Program for switch both embedded Suzuki processes in between two states
% with referred to time to generate simulation signal.
% {η^(1)(t) and η^(2)(t) -->  η(t)}
%
% Used m-files:mplot.m, simu_fun.m,
%---------------------------------------------------------------------------

ind1=1;
ind2=1;
tic

laenge=1e6;
eta=zeros(1,laenge);
```

```
if rand(1,1)>p1
   zus=2;
else
   zus=1;
end

for k=1:laenge
   cc=rand(1,1);
   if zus==1&cc<=p12
      zus=2;
   elseif zus==2&cc<=p21
      zus=1;
   end

   if zus==1
      eta(k)=eta1(ind1);
      ind1=ind1+1;
   else
      eta(k)=eta2(ind2);
      ind2=ind2+1;
   end
end
toc

mplot % calling of mplot.m file

%clear ind1 ind2
clear k cc zus laenge


% ---------------------------------------------------------------------------
% f2.m
% ---------------------------------------------------------------------------
% Used m-files:,simu_fun.m,
% ---------------------------------------------------------------------------
% [r,y]=f2(x,stufen,freq,r)
%----------------------------------------------------------------------------
%
% Explanation of the input parameters:
% x: probability that the signal x bigger as a level r.
% stufen:(levels) the number of stages
% freq: the frequency of the signal x
% r: is the vector level
% y: is the rate vector
function [r,y]=f2(x,stufen,freq,r);
% function [r, y] = f2 (x, step, freq)


if nargin==1
   stufen=80;
   freq=500;
end
if nargin==2
   freq=500;
end
if size(x,1)>1
```

```
   x=x';
end

%r=linspace(log10(min(x)),log10(max(x)),stufen);
%r=10.^r;
if nargin < 4
   anfang=2;
   r=(min(x):(max(x)-min(x))/(stufen-2):max(x));
   r2=(r(2)-r(1))/anfang+r(1);
   r3=(r(2)-r2)/2+r2;
   r=[r(1) r2 r3 r(2:length(r))];
   clear r2 r3 anfang
end

y=0*r;
for k=1:length(r)
   x1=x>r(k);
   y(k)=sum(x1);
end

y=y/length(x);
if nargout==0
   plot(r,y);
end
if nargout==1
   r=y;
end




% -------------------------------------------------------------------------
% lcrfun.m
% -------------------------------------------------------------------------
% Program for the computation of the level-crossing rate.
% Used m-files:simu_fun.m
% -------------------------------------------------------------------------
%[r,y]=lcrfun(x,stufen,freq,r);
% -------------------------------------------------------------------------
%
% Explanation of the input parameters:
% x: LCR (level crossing rate) calculated the levels below rate of the
% signal (x vector) .
% stufen: the number of stages
% freq: the frequency of the signal x
% r: level vector
% y: the rate vector

function [r,y]=lcrfun(x,stufen,freq,r);

% function [r, y] = lcrfun (x, step, freq)


if nargin==1
   stufen=80;
   freq=500;
end
```

```
if nargin==2
   freq=500;
end

if nargin < 4
   anfang=2;
   r=(min(x):(max(x)-min(x))/(stufen-2):max(x));
   r2=(r(2)-r(1))/anfang+r(1);
   r3=(r(2)-r2)/2+r2;
   r=[r(1) r2 r3 r(2:length(r))];
   clear r2 r3 anfang
end

y=0*r;
if size(x,1)>1
   x=x';
end
for k=1:length(r)
   x1=x<r(k);
   x1=x1-[x1(1) x1(1:length(x1)-1)];
   x1=x1>0;
   y(k)=sum(x1);
end

y=y*abs(freq)/length(x);
if nargout==0
   plot(r,y);
end
if nargout==1
   r=y;
end
  plot(r,y);


% --------------------------------------------------------------------------
% mplot.m
% --------------------------------------------------------------------------
% Program to plot "measured fading envelope" and "simulated fading
% envelope"
% Used m-files:
% --------------------------------------------------------------------------
% mplot.m
%---------------------------------------------------------------------------
%
font_size1=20;
font_size2=24;
font='Times';

figure(1)
clf

t1=[0:length(amp_mess)-1]/500;
semilogy(t1,amp_mess,'k')
set(0,'DefaultAxesFontSize',18);
axis([0 t1(length(t1)) 1e-4 10])
set(gca,'FontSize',font_size1,'FontName',font)
```

```
xlabel('Time, {\itt}  (s)','FontSize',font_size2,'FontName',font)
ylabel('Measured fading envelope','FontSize',font_size2,'FontName',font)

figure(2)
clf

t2=[0:length(eta)-1]/fabtast;
t2=t2(t2<=t1(length(t1)));

clear t1

%anf=62e4; % first value for time variable

zwi=1; % step width
set(0,'DefaultAxesFontSize',18);
semilogy(t2(1:zwi:length(t2)),eta(1+anf:zwi:length(t2)+anf),'k')
axis([0 t2(length(t2)) 1e-4 10])
%axis 'auto y'
set(gca,'FontSize',font_size1,'FontName',font)
xlabel('Time, {\itt}  (s)','FontSize',font_size2,'FontName',font)
ylabel('Simulated fading envelope','FontSize',font_size2,'FontName',font)

clear t2 zwi



% ------------------------------------------------------------------------
% prob.m
% ------------------------------------------------------------------------
% Program to calculate and display the transition probabilities of p12, p21,
% Delta minimum, Delta maximum and Mean Delta.
% Used m-files:ueberg2.m,
% ------------------------------------------------------------------------

tic
[Aij,delta]=ueberg2(amp_mess,1.25,0.25);

p12=Aij/x(15)/length(amp_mess)*500/fabtast;
p21=Aij/(1-x(15))/length(amp_mess)*500/fabtast;

disp(' ')
disp(['Total number of transitions = ',num2str(Aij)])
disp(['Transition probability (measured signal) p12  = ',...
    num2str(p12*fabtast/500)])
disp(['Transition probability (measured signal) p21  = ',...
    num2str(p21*fabtast/500)])
disp(['Transition probability (sim. signal)     p12  = ',...
    num2str(p12)])
disp(['Transition probability (sim. signal)     p21  = ',...
    num2str(p21)])
disp(['State probability P1        = ',num2str(x(15))])
disp(' ')
disp(['Delta minimum = ',num2str(min(delta)/500)])
disp(['Delta maximum = ',num2str(max(delta)/500)])
disp(['Mean Delta    = ',num2str(mean(delta)/500)])
disp(' ')

toc

% ------------------------------------------------------------------------
% ueberg2.m
%
% program to determine the number of transitions between the states of the
% amplitude of the measured signal.
% Used m-files:prob.m,
% ------------------------------------------------------------------------
%[Aij,delta] = ueberg(mess,r1,r2,width)
%-------------------------------------------------------------------------
%
% Explanation of the input parameters:
% Aij: number of transitions.
% delta: the duration of transitions (in sample intervals)
% meass: vector of the time course of the amplitude of the measured signal.
% r1: high level (for example: r1 = 0.8).
% r2: low level (for example: r2 = 0.1).
% ------------------------------------------------------------------------

function [Aij,delta] = ueberg2(mess,r1,r2)
%
% vec_r1: vector of all times, for which the amplitude of the
% Measurement signal is greater than the high level r1.

vec_r1 = find( mess > r1 );
%
% vec_r2: Vector of all times, for which the amplitude of the
% Measured signal is smaller than the small level r2.
vec_r2 = find( mess < r2 );
%
% Initialization of Aij:
if vec_r1(1) > vec_r2(1)
   Aij = 1;
else
   Aij = 0;
end
delta=[];%
% Calculation of Aij and delta:
for k = 1 : length(vec_r1)-1
   a = find( vec_r2 > vec_r1(k)  &  vec_r2 < vec_r1(k+1) );
   if length(a) > 0
      delta = [delta, vec_r2(a(1))-vec_r1(k), vec_r1(k+1)-
vec_r2(a(length(a)))];
      Aij = Aij + sign( length(a) );
   end
end
```

MATHLAB –PROGRAMS of Spatial Shadowing Channel 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,PLOT)
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(-abs(delta)/D);
        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',18);
    xlabel('Spatial separation, $\Delta x$ (m)',
'FontSize',24,'Interpreter','latex');
    ylabel('Spatial ACF, ${r}_{\nu\nu}$ $(\Delta x)$',
'FontSize',24,'Interpreter','latex');
    legend({['Reference model'],['Simulation for (MEA) N = ',num2str(N)]},...

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


%-----------------------------------------------------------------------
% 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,P
LOT)
[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

%-------------------------------------------------------------------------
% 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);

% ------------------------------------------------------------------------
% 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));
```

MATHLAB –PROGRAMS of MIMO Channel Simulator Based on One-Ring Model

```
%--------------------------------------------------------------------------
% MIMO_one_ring_parameters.m
%--------------------------------------------------------------------------
% Program for the computation of angle of arrivals alpha of MIMO one-ring-
% model by using LPNM method or Extended method of exact Doppler spread.
%
% Used files: data_one_ring.mat, errorfunc_one_ring.m
%--------------------------------------------------------------------------
% [alpha]=MIMO_one_ring_parameters(N,BetaBS,BetaMS,alphaTmax,kappa,f_max,
% mu_0,alphav,p,method,PLOT)
%--------------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: number of scatterers
% BetaBS: BS antenna tilt angle in degree
% BetaMS: MS antenna tilt angle in degree
% alphaT_max: one half of the maximum angle of departure in degree
% kappa: parameter used in Mises density
% f_max: maximum Doppler frequency
% mu_0: the angle of rotation in degree
% alphav: the angle of motion in degree
% p: the power of LPNM method
% method: parameters computation method that can be:
%       1) EMEDS:           'EMED'
%       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 [alpha]=MIMO_one_ring_parameters(N,BetaBS,BetaMS,alphaT_max...
,kappa,f_max,mu_0,alphav,p,method,PLOT)
if nargin<11
    PLOT=0;
end
% converting parameters from degre to radium
BetaBS=2*pi*BetaBS/360;
BetaMS=2*pi*BetaMS/360;
alphaT_max=2*pi*alphaT_max/360;
alphav=2*pi*alphav/360;
mu_0=2*pi*mu_0/360;
save data_one_ring.mat N BetaBS BetaMS alphaT_max kappa f_max mu_0 alphav p;
 % computation of the parameters phi using EMEDS
  if all(upper(method)=='EMED')
   alpha=zeros(N,1);
    for n=1:N;
    alpha_0_r=.5*pi/N;
    alpha(n)=(1/N)*2*pi*(n-0.5)+alpha_0_r;
    end

  elseif all(upper(method)=='LPNM')
 % computation of initial values for the parameter alphaR using LPNM
  alpha=zeros(N,1);
    for n=1:N;
     alpha(n)=fzero(@(x) fun_phi(n,N,kappa,x,mu_0),pi);
    end
if kappa==0;
    alpha=alpha;
else
% determination of the phases alpha through LPNM method
    options = optimset('Display','iter','MaxIter',100,'TypicalX',alpha);
    x=fminsearch('errorfunc_one_ring',alpha,options);
      alpha=x;
end
else
    disp('enter a valid input method belonging to{EMEDS,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);
alphaMS=linspace(-pi,pi,4*N);
for k=1:length(tau)
    ACF(k)=trapz(alphaMS,exp(-sqrt(-1)*2*pi*f_max*cos(alphaMS-alphav)...
        *tau(k)).*Mises(alphaMS,kappa,mu_0));
    ACF_teld(k)=sum(exp(-sqrt(-1).*2.*pi.*f_max.*cos(alpha-alphav)...
        .*tau(k)))./length(alpha);
```

```
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(0,1.25*N,4*N);
deltaMS=linspace(0,.125*N,4*N);
alphaMS=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(BetaBS)...
        +alphaT_max.*sin(BetaBS).*sin(alphaMS)));
        b=exp(sqrt(-1).*pi.*deltaMS(g).*cos(alphaMS-BetaMS));
        CCF(q,g)=trapz(alphaMS,a.^2.*b.^2.*Mises(alphaMS,kappa,mu_0));
        aa=exp(sqrt(-1).*pi.*deltaBS(q).*(cos(BetaBS)+...
            alphaT_max.*sin(BetaBS).*sin(alpha)));
        bb=exp(sqrt(-1).*pi.*deltaMS(g).*cos(alpha-BetaMS));
        CCF_teld(q,g)=sum(aa.^2.*bb.^2)/length(alpha);
    end
end
Err=CCF-CCF_teld;
if plott==1;

    figure(1)

    grid on;
    plot(tau*f_max,real(ACF),'k');
    hold on;
    plot(tau*f_max,real(ACF_teld),'k*')
        plot(tau*f_max,real(ACF_teld),'k-.')
    xlim([0 N/2])
    ylim([-1 1])
    set(0,'DefaultAxesFontSize',18);
    legend({['Reference model'],['Simulation model with LPNM, N = ',...
        num2str(N)],['Simulation model with (EMEDS N = ',num2str(N)]},...
        'FontName','times','Location','NorthEast',...
        'FontSize',22,'Interpreter','latex');
    xlabel({'Normalized time lag, $\tau$ . $f_{max}$'},'FontName',...
        'Arial','FontSize',24,'Interpreter','latex');
    ylabel({'Time ACF, $\rm{{r}_{h_{kl}}(\tau)}$ '},'FontName',...
        'times','FontSize',24,'Interpreter'...
        ,'latex');

    figure(2)
    surf(deltaBS,deltaMS,(real(CCF))');
    zlim([-.5 1])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',18);
    title({'Reference model'},'FontName','times',...
        'FontSize',20,'Interpreter','latex');
    zlabel({'2D space CCF,
$\rm{\rho_{}\mbox(\delta_T,\delta_R)}$'},'FontName','times',...
        'FontSize',20,'Interpreter','latex');
    xlabel({'$\delta_{T}$ / $\lambda_0$'},'FontName','times',...
    'FontSize',20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{R}$ / $\lambda_0$'},'FontName','times',...
        'FontSize',20,'Rotation',-20,'Interpreter','latex');
    figure(3)
    surf(deltaBS,deltaMS,real(CCF_teld)');
    zlim([-.5 1])
      set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',18);
    title({'Simulation model'},'FontName','times',...
        'FontSize',20,'Interpreter','latex');
     zlabel({'2D space CCF,
$\rm{\tilde{\rho}\mbox(\delta_T,\tau)}$'},'FontName','times',...
        'FontSize',20,'Interpreter','latex');
    xlabel({'$\delta_{T}$ / $\lambda_0$'},'FontName','times',...
    'FontSize',20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{R}$ / $\lambda_0$'},'FontName','times',...
        'FontSize',20,'Rotation',-20,'Interpreter','latex');
end

%---------------------------------------------------------------------------
% MIMO_one_ring_gains.m
%---------------------------------------------------------------------------
% Program for the determination of the MIMO one-ring channel gains.
%
% Used file: data_one_ring.mat
%---------------------------------------------------------------------------
%[H]=MIMO_one_ring_gains(Mbs,Mms,N,BetaBS,BetaMS,alpha_max,f_max,alphav)
%---------------------------------------------------------------------------
% Explanation of the input parameters:
%
% Mbs: number of antennas elements in base station
% Mms: number of antennas elements in mobile station
% N: number of scatterers around mobile station
% BetaBS: BS antenna tilt angle in degree
% BetaMS: MS antenna tilt angle in degree
% alpha_max: one half of the maximum angle of departure in degree
% kappa: parameter used in Mises density
% f_max: maximum Doppler frequency
% mu_0: the angle of rotation in degree
% alphav: the angle of motion in degree
%-------------------------------------------------------------------------
function[H]=MIMO_one_ring_gains(Mbs,Mms,N,BetaBS,BetaMS,alpha_max,f_max,alpha
v)
nbr=10000;
BetaBS=2*pi*BetaBS/360;
BetaMS=2*pi*BetaMS/360;
alpha_max=2*pi*alpha_max/360;
alphav=2*pi*alphav/360;
for i=1:N;
    alpha_0_r=.5*pi/N;
    alpha(i)=(1/N)*2*pi*(i-0.5)+alpha_0_r;
end
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 l=1:nbr
            sum1=0;
            for n=1:length(alpha)
                a(q)=exp(sqrt(-1)*pi*(Mbs-2*q+1)*(deltaBS/lamda)*...

(cos(BetaBS)+alpha_max*sin(BetaBS)*sin(alpha(n))));%equation number (8.11)
                b(P)=exp(sqrt(-1)*pi*(Mms-
2*P+1)*(deltaMS/lamda)*cos(alpha(n)-BetaMS));%equation (8.12)
                f=f_max.*cos(alpha(n)-alphav);
                sum1=sum1+a(q)*b(P)*exp(sqrt(-
1)*(2*pi*f.*t(l)+teta(l,n)));%equation (8.14)
            end
            H(q,P,l)=sum1/sqrt(length(alpha));
        end
    end
end
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(alpha)
%------------------------------------------------------------------------
% Explanation of the input parameters:
%
% phi: angle of arrival alpha

function [E]=errorfunc_one_ring(alpha)
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);
alphaMS=linspace(-pi,pi,6*N);
for k=1:length(tau)
    ACF(k)=trapz(alphaMS,exp(-j*2*pi*f_max*cos(alphaMS-
alphav)*tau(k)).*Mises(alphaMS,kappa,mu_0));
    ACF_teld(k)=sum(exp(-j*2*pi*f_max*cos(alpha-
alphav)*tau(k)))/length(alpha);
end
% The error function between space CCF reference and space CCF simulation
deltaBS=linspace(0,N/4,N);
deltaMS=linspace(0,N/4,N);
alphaMS=linspace(-pi,pi,4*N);
for q=1:length(deltaBS)
    for g=1:length(deltaMS)
        a=exp(sqrt(-
1).*pi.*deltaBS(q).*(cos(BetaBS)+alphaT_max.*sin(BetaBS).*sin(alphaMS)));
        b=exp(sqrt(-1).*pi.*deltaMS(g).*cos(alphaMS-BetaMS));
        CCF(q,g)=trapz(alphaMS,a.^2.*b.^2.*Mises(alphaMS,kappa,mu_0));
        aa=exp(sqrt(-
1).*pi.*deltaBS(q).*(cos(BetaBS)+alphaT_max.*sin(BetaBS).*sin(alpha)));
        bb=exp(sqrt(-1).*pi.*deltaMS(g).*cos(alpha-BetaMS));
        CCF_teld(q,g)=sum(aa.^2.*bb.^2)/length(alpha);
    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

%--------------------------------------------------------------------------
% 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,mu_0)
%--------------------------------------------------------------------------
% Explanation of the input parameters:
%
% n: iteration index
% N: number of the sinusoids
% x: arbitrary vector which should be as longer as N
% mu_0: angle of rotation

function [y]=fun_phi(n,N,kappa,x,mu_0)

phi_n=linspace(-pi,x,10000);
y=trapz(phi_n,Mises(phi_n,kappa,mu_0))-((1/N)*(n-1/4));


%--------------------------------------------------------------------------
% MIMO_one_ring_capacity.m
%--------------------------------------------------------------------------
% Program for the determination and the plotting of the MIMO capacity
%used file : MIMO_one_ring_gain.m
%--------------------------------------------------------------------------
%[C]=MIMO_one_ring_capacity(H,SNR,Ptotal,PLOT)
%--------------------------------------------------------------------------
% Explanation of the input parameters:
%
% H: channel gains Hij
% SNR: signal to noise ratio in dB
% Ptotal: total transmitted power in dB
% PLOT: if PLOT==1, plot of the capacity of the channel bits/sec/Hz over the
variable time
function [C]=MIMO_one_ring_capacity(H,SNR,Ptotal,PLOT)
if nargin<4
    PLOT=0;

end
```

```
t=0:0.001:1-0.001;
nbr=length(t);
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(t,C);
    hold on
    plot(t,Cmean,'r--');
    set(0,'DefaultAxesFontSize',16);
    legend({'Capacity C','mean of C'},...

'FontName','Arial','Location','NorthEast','FontSize',20,'Interpreter','latex'
);
    ylabel({'Simulated channel capacity,C(t)(bit/s/Hz)
'},'FontName','Arial','FontSize',18,'Interpreter','latex');
    xlabel({'Time, t
(s)'},'FontName','Arial','FontSize',20,'Interpreter','latex');
    xlim([0 .5])
    ylim([0 11])
end
```

MATHLAB –PROGRAMS of MIMO Channel Simulator Based on Two-Ring Model

```
%---------------------------------------------------------------------------
% MIMO_tow_ring_gains.m
%---------------------------------------------------------------------------
% Program for the determination of the MIMO tow-ring channel gains
%
% Used file: data_tow_ring.mat
%---------------------------------------------------------------------------
%[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 around the BS
% M: number of scatterers around the MS
% Rt: radius of scatterers around the BS ring
% Rr: radius of scatterers around the MS ring
% Mt: number of antennas element in base station
% Mr: number of antennas element in mobile station
% kappa
% alphaT: BS antenna tilt angle in degree
% fT_max: Doppler frequency at the transmitter side
% betaT: transmit antenna tilt angle
% alphaR: MS antenna tilt angle in degree
% 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,alphavT,fT_max,BetaT,alphavR,fR
_max,BetaR)

%MIMO tow-ring parameters
alphavT=2*pi*alphavT/360;
BetaT=2*pi*BetaT/360;
alphavR=2*pi*alphavR/360;
BetaR=2*pi*BetaR/360;
nbr=10000;
p=2;
[alphaR]=gen_alphaR_MIMO_tow_ring(M,alphavR,fR_max,BetaR,kappa,p,'emed',0);
[alphaT]=gen_alphaT_MIMO_tow_ring(N,alphavT,fT_max,BetaT,kappa,p,'emed',0);
lamda=0.15;
deltaT=lamda/2;
deltaR=lamda/2;
tetan=2*pi*rand(1,length(alphaR));
tetam=2*pi*rand(1,length(alphaT));
tetamn=[];
for a=1:length(alphaR)
    tetamn=[tetamn;mod(tetam+tetan(a),2*pi)];
end
fT=fT_max*cos(alphaT-alphavT);
fR=fR_max*cos(alphaR-alphavR);
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(alphaT)
            sum1=0;
            for n=1:length(alphaR)
                an=exp((Mt-2*q+1).*sqrt(-1)*pi*deltaT*(cos(alphaT(m)-
BetaT)));
                bn=exp((Mr-2*p+1).*sqrt(-1)*pi*deltaR*cos(alphaR(n)-BetaR));
                cmn=exp(sqrt(-1)*2*pi*(Rt*cos(alphaT(m))-
Rr*cos(alphaR(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(alphaR))*sqrt(length(alphaT)));
        end
    end
end


%-------------------------------------------------------------------------
% gen_alphaR_MIMO_tow_ring.m
%-------------------------------------------------------------------------
%
% Program for the computation for the angle of arrival alphaT for
% MIMO tow-ring model using EMEDS and Modified MEA.
```

```
%
% Used files: data_tow_ring.mat, errorfunc.m
%------------------------------------------------------------------------
%
[alphaR]=gen_alphaR_MIMO_tow_ring(N,alphavT,fT_max,BetaT,,kappa,p,method,PLOT
)
%------------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: number of scatterers
% alphavT: MS motivation angle in degree.
% fT_max: Doppler frequency at the transmitter side
% BetaT: receiver antenna tilt angle
% p: the power of LPNM method
% method: parameters computation method that can be:
%       1) Modified MEA:    'MMEA'
%       2) Extended MEDS :  'EMED'
% PLOT: if PLOT==1, plot of the Time ACF and 2D-space CCF of the reference
% model as well as the simulation model

function
[alphaR]=gen_alphaR_MIMO_tow_ring(N,alphavR,fR_max,BetaR,kappa,p,method,PLOT)

% converting parameters from degree to radian
alphavR=2*pi*alphavR/360;
BetaR=2*pi*BetaR/360;
save data_tow_ring.mat N alphavR fR_max kappa BetaR p;
% computation of the parameters alpha using Extended MEDS

if all(upper(method)=='EMED')
     alphaR=zeros(N,1);
     for i=1:N
    alphaR0=.5*pi/N;
    alphaR(i)=(1/N)*2*pi*(i-0.5)+alphaR0;

     end


else all(upper(method)=='MMEA')
    % computation of initial values for the parameter phi using MMEA

    for n=1:N
        alphaR(n)=fzero(@(x) fun_phi(n,N,kappa,x,BetaT),BetaT);
    end
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 {EMED,MMEA}');
% end

tau=linspace(0,N/(2*fR_max),8*N);
alphaMS=linspace(-pi,pi,4*N);
```

```
for k=1:length(tau)
    f=fR_max*cos(alphaMS-alphavR);
    ACF(k)=trapz(alphaMS,exp(-2*sqrt(-
1)*pi*f*tau(k)).*Mises(alphaMS,kappa,BetaR));
    ff=fR_max*cos(alphaR-alphavR);
    ACF_teld(k)=sum(exp(-2*sqrt(-1)*pi*ff*tau(k)))/length(alphaR);
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/8,2*N);
tau1=linspace(0,N/(8*fR_max),2*N);
alphaMS=linspace(-pi,pi,4*N);%the angle with uniform distribution in
reference model
for q=1:length(deltaR)
    for g=1:length(tau1)
        a=exp(sqrt(-1)*pi*deltaR(q).*cos(alphaMS-BetaR));
        f=fR_max*cos(alphaMS-alphavR);
        ro12(q,g)=trapz(alphaMS,a.^2.*exp(-2*sqrt(-
1)*pi*f*tau1(g)).*Mises(alphaMS,kappa,BetaR));
        aa=exp(sqrt(-1)*pi*deltaR(q).*cos(alphaR-BetaR));
        ff=fR_max*cos(alphaR-alphavR);
        ro12_teld(q,g)=sum(aa.^2.*exp(-2*sqrt(-
1)*pi*ff*tau1(g)))/length(alphaR);
    end
end
Err=abs(ro12-ro12_teld);
if PLOT==1
   figure(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',18);
    legend({['Reference model'],['Simulation model with N =
',num2str(N)]},...

'FontName','times','Location','NorthEast','FontSize',24,'Interpreter','latex'
);
    xlabel({'$\tau$ .
$f_{max}$'},'FontName','times','FontSize',24,'Interpreter','latex');
    ylabel({'Time
ACF'},'FontName','Arial','FontSize',24,'Interpreter','latex');
    figure(2);
    surf(tau1*fR_max,deltaR,real(ro12));
    zlim([-.5 1])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',18);
    title({'Reference
model'},'FontName','times','FontSize',24,'Interpreter','latex');
    zlabel({'Reciver CF,$\rm{{\rho_R}\mbox(\delta_R,\tau)}$
'},'FontName','Arial','FontSize',24,'Interpreter','latex');
    xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',24,...
        'Rotation',20,'Interpreter','latex');
    ylabel({'$\delta_R$ / $\lambda_0$'},'FontName','times','FontSize',24,...
        'Rotation',-20,'Interpreter','latex');
    zlim([-.5 1])
    xlim([0 5])
    ylim([0 5])

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    figure(3);
    surf(tau1*fR_max,deltaR,real(ro12_teld));
    set(gca,'YDir','reverse');
    zlim([-.5 1])
    xlim([0 5])
    ylim([0 5])

    set(0,'DefaultAxesFontSize',18);
    title({'Simulation
model'},'FontName','Arial','FontSize',24,'Interpreter','latex');
    zlabel({'Reciver CF,$\rm{\tilde{\rho_R}\mbox(\delta_R,\tau)}$
'},'FontName','Arial','FontSize',24,'Interpreter','latex');
    xlabel({'$\tau$ . $f^R_{max}$'},'FontName','times','FontSize',24,...
        'Rotation',20,'Interpreter','latex');
    ylabel({'$\delta_R$ / $\lambda_0$'},'FontName','times','FontSize',24,...
        'Rotation',-20,'Interpreter','latex');
    figure(4);
    %zlabel({'2D space CCF,
$\rm{\tilde{\rho}\mbox(\delta_T,\delta_R)}$'},'FontName','times',...
        %'FontSize',20,'Interpreter','latex');
    surf(tau1*fR_max,deltaR,real(Err));
    set(gca,'YDir','reverse');
    zlim([0 .4])
    xlim([0 5])
    ylim([0 5])
    set(0,'DefaultAxesFontSize',18);
    zlabel({'Absolute error,$\rm{{e_R}\mbox(\delta_R,\tau)}$
'},'FontName','times','FontSize',24,'Interpreter','latex');
    xlabel({'$\tau$ . $f^R_{max}$'},'FontName','times','FontSize',24,...
        'Rotation',20,'Interpreter','latex');
    ylabel({'$\delta_R$ / $\lambda_0$'},'FontName','times','FontSize',24,...
        'Rotation',-20,'Interpreter','latex');
else
    z=1;
end


%-------------------------------------------------------------------------
% gen_alphaT_MIMO_tow_ring.m
%-------------------------------------------------------------------------
% Program for the computation of the angle-of-departure alphaT of
% MIMO tow-ring model.   Extended MEDS method was used for isotropic and
% modified MEA used for non-isotropic to test the model performance
%
%
% Used files: data_tow_ring.mat,
%-------------------------------------------------------------------------
%
[alphaT]=gen_alphaT_MIMO_tow_ring(M,alphavT,fT_max,BetaT,kappa,p,method,PLOT)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
%
```

```
% M: number of scatterers at transmitter side
% alphavT: BS motion angle in degree
% fT_max: Doppler frequency at the transmitter side
% BetaT: transmit antenna tilt angle
% p: the power of LPNM method
% method: parameters computation method that can be:
%       1) Modified MEA:    'MMEA'
%       2) EMEDS:           'EMED'
% PLOT: if PLOT==1, plot of the Time ACF and Transmitter CF of the reference
% model as well as the simulation model

function
[alphaT]=gen_alphaT_MIMO_tow_ring(N,alphavT,fT_max,BetaT,kappa,p,method,PLOT)
if nargin<8
    PLOT=0;
end
% converting parameters from degree to radian
alphavT=2*pi*alphavT/360;
mu_0 = pi/3;
BetaT=2*pi*BetaT/360;
save data_tow_ring.mat N alphavT fT_max kappa BetaT p mu_0;
% computation of the parameters alphaT using MMEA
 if all(upper(method)=='MMEA')
  alphaT=zeros(N,1);
   for n=1:N
        alphaT(n)=fzero(@(x) fun_alphaT(n,N,kappa,x,BetaT),BetaT);
     end

 elseif all(upper(method)=='EMED')% computation of the parameters alphaT
using EMEDS
     alphaT=zeros(N,1);
     for i=1:N;
    alphaT0=.5*pi/N;
    alphaT(i)=(1/N)*2*pi*(i-0.5)+alphaT0;
     end
else
    disp('enter a valid input method belonging to {MMEA,EMED}');
 end
tau=linspace(0,N/(2*fT_max),8*N);
alphaBS=linspace(-pi,pi,4*N);
for k=1:length(tau)
    f=fT_max*cos(alphaBS-alphavT);
    ACF(k)=trapz(alphaBS,exp(-2*sqrt(-
1)*pi*f*tau(k)).*Mises(alphaBS,kappa,BetaT));
    ff=fT_max*cos(alphaT-alphavT);
    ACF_teld(k)=sum(exp(-2*sqrt(-1)*pi*ff*tau(k)))/length(alphaT);
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(0,5,2*N);
tau1=linspace(0,5/(fT_max),2*N);
alphaBS=linspace(-pi,pi,4*N);
for q=1:length(deltaT)
    for g=1:length(tau1)
        a=exp(sqrt(-1)*pi*deltaT(q).*cos(alphaBS-BetaT));
        f=fT_max*cos(alphaBS-alphavT);
        ro12(q,g)=trapz(alphaBS,a.^2.*exp(-2*sqrt(-1)*pi*f*tau1(g)).*...
            Mises(alphaBS,kappa,BetaT));
        aa=exp(sqrt(-1)*pi*deltaT(q).*cos(alphaT-BetaT));
        ff=fT_max*cos(alphaT-alphavT);
        ro12_teld(q,g)=sum(aa.^2.*exp(-2*sqrt(-1)*pi*...
            ff*tau1(g)))/length(alphaT);
    end
end
Err=abs(ro12-ro12_teld);
if PLOT==1
    figure(1)
    plot(tau*fT_max,real(ACF));
    hold on;
    plot(tau*fT_max,real(ACF_teld),'r*')
    xlim([0 N/4])
    ylim([-1 1])
    set(0,'DefaultAxesFontSize',16);
    legend({['Reference model'],['Simulation model with N =
',num2str(N)]},...
        'FontName','times','Location','NorthEast','FontSize',16,...
        'Interpreter','latex');
    xlabel({'$\tau$ . $f_{max}$'},'FontName','times','FontSize',20,...
        'Interpreter','latex');
    ylabel({'Time ACF'},'FontName','times','FontSize',20,'Interpreter',...
        'latex');
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    figure(2)
    if kappa==0
    surf(tau1*fT_max,deltaT,real(ro12));
    zlim([-.5 1])
    xlim([0 5])
    ylim([0 5])
    zlabel({'Transmitter CF, $\rm{\rho_{T}\mbox(\delta_T,\tau)}$'},...
        'FontName','times','FontSize',20,'Interpreter','latex');
    else
    surf(tau1*fT_max,deltaT,abs(ro12));
    zlim([0 1])
    xlim([0 5])
    ylim([0 5])
    zlabel({'Transmitter CF,
$\rm\mid{\rho_{T}\mbox(\delta_T,\tau)}\mid$'},...
        'FontName','times','FontSize',20,'Interpreter','latex');
    end
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',18);
    title({'Reference model'},'FontName','times',...
        'FontSize',20,'Interpreter','latex');

    xlabel({'$\tau$ . $f^T_{max}$'},'FontName','times',...
    'FontSize',20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{T}$ / $\lambda_0$'},'FontName','times',...
        'FontSize',20,'Rotation',-20,'Interpreter','latex');

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    figure(3)
     if kappa==0
    surf(tau1*fT_max,deltaT,real(ro12_teld));
    zlim([-.5 1])
    xlim([0 5])
    ylim([0 5])
    zlabel({'Transmitter CF, $\rm{\tilde{\rho_T}\mbox(\delta_T,\tau)}$'},...
         'FontName','times','FontSize',20,'Interpreter','latex');
     else
      surf(tau1*fT_max,deltaT,abs(ro12_teld));
     zlim([0 1])
     xlim([0 5])
    ylim([0 5])
      zlabel({'Transmitter CF,
$\rm\mid{\tilde{\rho_T}\mbox(\delta_T,\tau)}\mid$'},...
         'FontName','times','FontSize',20,'Interpreter','latex');
     end
      set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',18);
    title({'Simulation model'},'FontName','times',...
        'FontSize',20,'Interpreter','latex');

    xlabel({'$\tau$ . $f^T_{max}$'},'FontName','times',...
    'FontSize',20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{T}$ / $\lambda_0$'},'FontName','times',...
        'FontSize',20,'Rotation',-20,'Interpreter','latex');
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    figure(4)
    surf(tau1*fT_max,deltaT,abs(Err));
    xlim([0 5])
    ylim([0 5])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',16);
   zlabel({'Absolute error, $\rm{e_T\mbox(\delta_T,\tau)}$'},'FontName',...
       'times', 'FontSize',20,'Interpreter','latex');

    xlabel({'$\tau$ . $f_{max}$'},'FontName','Arial','FontSize',20,...
        'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_T$ / $\lambda_0$'},'FontName','Arial','FontSize',20,...
        'Rotation',-20,'Interpreter','latex');
end


%--------------------------------------------------------------------------
% fun_alphaT.m
%--------------------------------------------------------------------------
% Computation of the equation relatively to MMEA method that should be
% resolved in order to find the AoA alphaT
%
%--------------------------------------------------------------------------
% [y]=fun_alphaT(n,N,kappa,x,mu_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_alphaT(n,N,kappa,x,mu_0)
```

```
alphaB=linspace(mu_0-pi,x,10000);
y=-1*trapz(alphaB,Mises(alphaB,kappa,mu_0))+((1/N)*(n-1/4));


%--------------------------------------------------------------------------
% MIMO_two_ring_capacity.m
%--------------------------------------------------------------------------
% Program for the determination and the plotting of the MIMO capacity
% used file : MIMO_tow_ring_gain.m
%--------------------------------------------------------------------------
%[C]=MIMO_two_ring_capacity(H,SNR,Ptotal,PLOT)
%--------------------------------------------------------------------------
% Explanation of the input parameters:
%
% H: channel gains Hij
% SNR: signal to noise ratio in dB
% Ptotal: total transmitted power in dB
% PLOT: if PLOT==1, plot of the capacity of the channel bits/sec/Hz over the
variable time
function [C]=MIMO_two_ring_capacity(H,SNR,Ptotal,PLOT)
if nargin<4
    PLOT=0;
end
t=0:0.001:1-0.001;
nbr=length(t);
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(t,C);
    hold on
    plot(t,Cmean,'r--');
    set(0,'DefaultAxesFontSize',16);
    legend({'Capacity C','mean of C'},...

'FontName','Arial','Location','NorthEast','FontSize',20,'Interpreter','latex'
);
    ylabel({'Simulated channel capacity,C(t)(bit/s/Hz)
'},'FontName','Arial','FontSize',18,'Interpreter','latex');
    xlabel({'Time, t
(s)'},'FontName','Arial','FontSize',20,'Interpreter','latex');
    xlim([0 .5])

end
```

MATHLAB –PROGRAMS of MIMO Channel Simulator Based on Elliptical Model

```
%-------------------------------------------------------------------------
% MIMO_elliptical_gains.m
%-------------------------------------------------------------------------
% Program for the determination of the MIMO elliptical model channel gains
%
%-------------------------------------------------------------------------
%[H]=MIMO_elliptical_gains(Mt,Mr,N,BetaT,BetaR,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
% N: number of scatterers
% BetaT: BS antenna tilt angle in degree
% BetaR: MS antenna tilt angle in degree
% kappa: parameter used in Mises density
% f_max: maximum Doppler frequency
% mu_0: the angle of rotation in degree
% alphav: the angle of motion in degree

function
[H]=MIMO_elliptical_gains(Mt,Mr,N,BetaT,BetaR,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,BetaT,BetaR,kappa,f_max,phi_0,alphav,p,'lp
nm');
        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)-BetaT)));
                b=exp(sqrt(-1)*pi*(Mr-2*k+1)*(deltaR/lamda)*cos(phi(n)-
BetaR));
                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_parameters.m
%------------------------------------------------------------------------
% Program for the computation of the angle of arrivals alpha of MIMO
% elliptical model using LPNM method.
%
% Used files: data_elliptical.mat, errorfunc_elliptical.m, gen_alphaT.
%------------------------------------------------------------------------
% [alpha]=MIMO_elliptical_parameters(N,BetaBS,BetaMS,kappa,f_max,alphav,
%       alpha_0,p,method,PLOT)
%------------------------------------------------------------------------
% Explanation of the input parameters:
%
% N: number of scatterers
% BetaT: BS antenna tilt angle in degree
% BetaR: MS antenna tilt angle in degree
% kappa: parameter used in Mises density
% f_max: maximum Doppler frequency
% alpha_0: the angle of rotation in degree
% alphav: the angle of motion in degree
% p: the power of LPNM method
% method: parameters computation method that can be:
%       1) 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 [alpha]=MIMO_elliptical_parameters...
    (N,BetaT,BetaR,kappa,f_max,alpha_0,alphav,p,method,PLOT)
if nargin<10
    PLOT=0;
end
% converting parameters from degre to radian
BetaT=2*pi*BetaT/360;
BetaR=2*pi*BetaR/360;
alphav=2*pi*alphav/360;
alpha_0=2*pi*alpha_0/360;
save data_elliptical.mat N BetaT BetaR kappa f_max p alpha_0 alphav;

% computation of initial values for the parameter alpha using EMEDS
% if all(upper(method)=='EMED')
%     alpha=zeros(N,1);
%     for n=1:N
%     alpha_0_r=.5*pi/N;
%     alpha(n)=(1/N)*2*pi*(n-0.5)+alpha_0_r;



if all(upper(method)=='LPNM')
    alpha=zeros(N,1);
    for n=1:N
        alpha(n)=fzero(@(x) fun_alpha(n,N,kappa,x,alpha_0),alpha_0);
    end
%     % determination of the phases alpha through LPNM method
%     options = optimset('Display','iter','TolFun',1*10^(-12),'MaxIter',...
%         1000,'MaxFunEvals',100000000,'TypicalX',alpha,'TolX',1*10^(-12));
%     x=fminsearch('errorfunc_elliptical',alpha,options);
%     alpha=x;
else
```

```
    disp('enter a valid input method belonging to{EMEDS,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);
alphaMS=linspace(-pi,pi,4*N);
for k=1:length(tau)
    ACF(k)=trapz(alphaMS,exp(-sqrt(-1)*2*pi*f_max*cos(alphaMS-alphav)*...
        tau(k)).*Mises(alphaMS,kappa,alpha_0));
    ACF_teld(k)=sum(exp(-sqrt(-1).*2.*pi.*f_max.*...
        cos(alpha-alphav).*tau(k)))./length(alpha);
end

% testing the correctness of the simulation model by determining the
%2D-space CCF for the simulation model as well as the reference model
if kappa==0
deltaT=linspace(0,N/5,2*N);
deltaR=linspace(0,N/5,2*N);
alphaR=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_alphaT(alphaR)-BetaT));
        dn=exp(-sqrt(-1).*2*pi.*deltaR(g).*cos(alphaR-BetaR));
        CCF(q,g)=(trapz(alphaR,cn.*dn.*Mises(alphaR,kappa,alpha_0)));
        cn_sim=exp(-sqrt(-1).*2.*pi.*deltaT(q).*...
            cos(gen_alphaT(alpha)'-BetaT));
        dn_sim=exp(-sqrt(-1).*2.*pi.*deltaR(g).*cos(alpha-BetaR));
        CCF_teld(q,g)=(sum(cn_sim.*dn_sim)/length(alpha));
    end
end
Err=CCF-CCF_teld;
if PLOT==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');
     figure(2)
    surf(deltaT,deltaR,(real(CCF))');
    zlim([-.5 1])
     ylim([0 5])
    xlim([0 5])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',18);
    title({'Reference model'},'FontName','times',...
        'FontSize',20,'Interpreter','latex');
    zlabel({'2D space CCF, $\rm{\rho_{11,22}\mbox(\delta_T,\delta_R)}$'},...
        'FontName','times',...
        'FontSize',20,'Interpreter','latex');
    xlabel({'$\delta_{R}$ / $\lambda_0$'},'FontName','times',...
    'FontSize',20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{T}$ / $\lambda_0$'},'FontName','times',...
        'FontSize',20,'Rotation',-20,'Interpreter','latex');
    figure(3)
    surf(deltaT,deltaR,real(CCF_teld)');
    zlim([-.5 1])
     ylim([0 5])
    xlim([0 5])
      set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',18);
    title({'Simulation model'},'FontName','times',...
        'FontSize',20,'Interpreter','latex');
     zlabel({'2D space
CCF,$\rm{\tilde{\rho}_{11,22}\mbox(\delta_T,\delta_R)}$'},'FontName','times',
...
        'FontSize',20,'Interpreter','latex');
    xlabel({'$\delta_{R}$ / $\lambda_0$'},'FontName','times',...
    'FontSize',20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{T}$ / $\lambda_0$'},'FontName','times',...
        'FontSize',20,'Rotation',-20,'Interpreter','latex');
    figure(4)
    surf(deltaR,deltaT,abs(Err)');
    zlim([0 1])
    ylim([0 5])
    xlim([0 5])
      set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',18);
    title({'Simulation model'},'FontName','times',...
        'FontSize',20,'Interpreter','latex');
      zlabel({'Absolute error, $\rm{e_T\mbox(\delta_T,\delta_R)}$'},...
          'FontName','times', 'FontSize',20,'Interpreter','latex');
    xlabel({'$\delta_{R}$ / $\lambda_0$'},'FontName','times',...
    'FontSize',20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{T}$ / $\lambda_0$'},'FontName','times',...
        'FontSize',20,'Rotation',-20,'Interpreter','latex');
end
else
 deltaT=linspace(0,N/3,2*N);
deltaR=linspace(0,N/16,2*N);
alphaR=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_alphaT(alphaR)-BetaT));
        dn=exp(-sqrt(-1).*2*pi.*deltaR(g).*cos(alphaR-BetaR));
        CCF(q,g)=(trapz(alphaR,cn.*dn.*Mises(alphaR,kappa,alpha_0)));
        cn_sim=exp(-sqrt(-1).*2.*pi.*deltaT(q).*...
            cos(gen_alphaT(alpha)'-BetaT));
        dn_sim=exp(-sqrt(-1).*2.*pi.*deltaR(g).*cos(alpha-BetaR));
        CCF_teld(q,g)=(sum(cn_sim.*dn_sim)/length(alpha));
    end
```

```
end
if PLOT==1
Err=CCF-CCF_teld;
     plot(tau*f_max,abs(ACF));
    hold on;
    plot(tau*f_max,abs(ACF_teld),'r*')
    xlim([0 N/2])
    ylim([0 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');
     figure(2)
    surf(deltaR,deltaT,(abs(CCF))');
    zlim([0 1])
    xlim([0 1.6])
    ylim([0 8.5])
    set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',18);
    title({'Reference model'},'FontName','times',...
        'FontSize',20,'Interpreter','latex');
     zlabel({'2D space CCF,
$\rm\mid{{\rho_{11,22}}\mbox(\delta_T,\delta_R)}\mid$'},...
         'FontName','times','FontSize',20,'Interpreter','latex');
    xlabel({'$\delta_{R}$ / $\lambda_0$'},'FontName','times',...
    'FontSize',20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{T}$ / $\lambda_0$'},'FontName','times',...
        'FontSize',20,'Rotation',-20,'Interpreter','latex');

 figure(3)
    surf(deltaR,deltaT,abs(CCF_teld)');
    zlim([0 1])
     xlim([0 1.6])
    ylim([0 8.5])
      set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',18);
    title({'Simulation model'},'FontName','times',...
        'FontSize',20,'Interpreter','latex');
     zlabel({'2D space CCF,
$\rm\mid{{\tilde\rho_{11,22}}\mbox(\delta_T,\delta_R)}\mid$'},...
         'FontName','times','FontSize',20,'Interpreter','latex');
    xlabel({'$\delta_{R}$ / $\lambda_0$'},'FontName','times',...
    'FontSize',20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{T}$ / $\lambda_0$'},'FontName','times',...
        'FontSize',20,'Rotation',-20,'Interpreter','latex');

     figure(4)
    surf(deltaR,deltaT,abs(Err)');
    zlim([0 1])
    ylim([0 8.6])
    xlim([0 1.6])
      set(gca,'YDir','reverse');
    set(0,'DefaultAxesFontSize',18);
    title({'Simulation model'},'FontName','times',...
        'FontSize',20,'Interpreter','latex');
      zlabel({'Absolute error, $\rm{e_T\mbox(\delta_T,\delta_R)}$'},...
          'FontName', 'times', 'FontSize',20,'Interpreter','latex');
    xlabel({'$\delta_{R}$ / $\lambda_0$'},'FontName','times',...
    'FontSize',20,'Rotation',12,'Interpreter','latex');
    ylabel({'$\delta_{T}$ / $\lambda_0$'},'FontName','times',...
        'FontSize',20,'Rotation',-20,'Interpreter','latex');

    end
end
end


%---------------------------------------------------------------------------
% gen_alphaT.m
%---------------------------------------------------------------------------
% Program for the computation of angles of departure according to given
% angles of arrivals
%
%---------------------------------------------------------------------------
% [alphaT]=gen_alphaT(alphaR)
%---------------------------------------------------------------------------
% Explanation of the input parameters:
%
% alphaR: angles of arrivals

function [alphaT]=gen_alphaT(alphaR)
v=1.5;
alphaT=zeros(1,length(alphaR));
alphav=atan((v^2-1)/(2*v));
for i=1:length(alphaR)
  if (alphaR(i)>-pi)&(alphaR(i)<=-alphav)
             alphaT(i)=atan(((v^2-1)*sin(alphaR(i)))/((2*v)-
(v^2+1)*cos(alphaR(i))));
  elseif (alphaR(i)>-alphav)&(alphaR(i)<=0)
             alphaT(i)=atan(((v^2-1)*sin(alphaR(i)))/((2*v)-
(v^2+1)*cos(alphaR(i))))-pi;
  elseif (alphaR(i)>0)&(alphaR<=alphav)
             alphaT(i)=atan(((v^2-1)*sin(alphaR(i)))/((2*v)-
(v^2+1)*cos(alphaR(i))))+pi;
  elseif (alphaR(i)>alphav)&(alphaR<=pi)
             alphaT(i)=atan(((v^2-1)*sin(alphaR(i)))/((2*v)-
(v^2+1)*cos(alphaR(i))));
  end
end

%---------------------------------------------------
% von Mises PDF
%---------------------------------------------------
function [v]=Mises(phi,kappa,phi_0)

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

MATHLAB –PROGRAMS of Uncorrelated Rayleigh Channel Simulator

```
%----------------------------------------------------------------------------
% errorfunE.m
%----------------------------------------------------------------------------
% Program to compute the error function for the angel-of rotation by using
% GMEDS1 for various values of Ni.
% errorfunE2(Ni,fmax,q,p,PLOT)
%----------------------------------------------------------------------------
% Exeplanation of inputs:
% Ni : are the number of sinusoids its will in as [N1;N2;N3]
% fmax : Maximum Doppler frequency
% q : is the index for GEMEDSq method for Example q=1 (GEMEDS1)
% p : the LPNM power.
%----------------------------------------------------------------------------
function errorfunE2(Ni,fmax,q,p,PLOT)
alpha0=linspace(-pi/2,pi/2,450);
for i=1:length(Ni)
    taumax=Ni(i)/(2*q*fmax);
    tau=linspace(0,taumax,100);
    c=sqrt(2/Ni(i));
    for j=1:length(alpha0)
        alpha=q*pi/(2*Ni(i))*((1:Ni(i))-1/2)+alpha0(j);
        f=fmax.*cos(alpha);
        R=besselj(0,2*pi*fmax.*tau);
        r=zeros(1,length(tau));
        for n=1:Ni(i)
            r=r+c^2/2*cos(2*pi*f(n).*tau);
        end
        E(i,j)=(1/taumax*trapz(tau,(abs(R-r)).^p))^(1/p);
    end
end
if PLOT==1
set(0,'DefaultAxesFontSize',18);
plot(alpha0,E(1,:),'k-','LineWidth',1.5);
hold on
plot(alpha0,E(2,:),'k-.','LineWidth',1.5);
plot(alpha0,E(3,:),'k--','LineWidth',1.5);

legend({'$N_i=20$','$N_i=25$','$N_i=30$'},...
    'Interpreter','Latex','Fontsize',25);
xlim([-pi/2 pi/2])
%ylim([0.5 1])

set(gca,'XMinorTick','off')
set(gca,'XTick',[-pi/2 0 pi/2])
set(gca,'XTickLabel',{'','',''})
xlabel('Angle of rotation $\alpha_{i,0}^{(k)}$ (rad)',...
    'Interpreter','Latex','Fontsize',25);
ylabel('Error function, $E_2(\alpha_{i,0}^{(k)})$',...
    'Interpreter','Latex','Fontsize',25);
 annotation('textbox',...
  'Position',[0.16065221 0.0017 0.07031 0.09141],...
  'LineStyle','none',...
  'FontName','Arial',...
  'FontSize',25,...
  'String',{'$-\frac{\pi}{2}$'},...
  'Interpreter','latex',...
  'FitHeightToText','on');
 annotation('textbox',...
  'Position',[0.5222 0.035 0.04296 0.07352],...
  'LineStyle','none',...
  'FontName','Arial',...
  'FontSize',25,...
  'String',{'$0$'},...
  'Interpreter','latex',...
  'FitHeightToText','on');
annotation('textbox',...
  'Position',[0.886841277 0.0026374 0.07226 0.09141],...
  'LineStyle','none',...
  'FontName','Arial',...
  'FontSize',25,...
  'String',{'$\frac{\pi}{2}$'},...
  'Interpreter','latex',...
  'FitHeightToText','on');
else
    X=0
end
end


%--------------------------------------------------------------------------
% ACF_Uncorr(N,alpha01,alpha02,alpha03)
%--------------------------------------------------------------------------
% This program to compute the Autocorrelation function of the reference
% in comparison to the autocorrelation functions of the simulation model.
%
% ACF_Uncorr(N,alpha01,alpha02,alpha03)
%--------------------------------------------------------------------------
% Explanation of the input parameters
% N : number of sinusoids
% alpha01 :angle-of-rotation for MEDS method.
% alpha02 :angle-of-rotation for GMEDS1 method positive value.
% alpha03 :angle-of-rotation for GMEDS1 method negative value.
%--------------------------------------------------------------------------
% To get the same result as fig 10.3.
% user can can use the angle-of-rotation = 0 for MEDS method and angle-of-
rotation
% alpha0=+pi/120 and -pi/120.
%--------------------------------------------------------------------------
function ACF_Uncorr(N,alpha01,alpha02,alpha03)


alpha0=[alpha01,alpha02,alpha03];
c=sqrt(2/N);
tau_fmax=linspace(0,15,600);
r=zeros(length(alpha0),length(tau_fmax));
for i=1:length(alpha0)
    alpha=1*pi/(2*N)*((1:N)-1/2)+alpha0(i);
    for n=1:N
```

```
        r(i,:)=r(i,:)+c^2/2.*cos(2*pi.*tau_fmax.*cos(alpha(n)));
    end
end
r_ref=besselj(0,2*pi.*tau_fmax);
plot(tau_fmax,r_ref,'k-','LineWidth',1.5);
hold on
plot(tau_fmax,r(1,:),'k-.','LineWidth',1.5);
plot(tau_fmax,r(2,:),'k:','LineWidth',2.5);
plot(tau_fmax,r(3,:),'k--','LineWidth',1.5);
set(0,'DefaultAxesFontSize',18);
legend({'Reference model',...
    'Simulation model, MEDS $(\alpha_{i,0}^{(k)}=0)$',...
    'Simulation model, GMEDS1 $(\alpha_{i,0}^{(k)}=\pi/120)$',...
    'Simulation model, GMEDS1 $(\alpha_{i,0}^{(k)}=-\pi/120)$'},...
    'Interpreter','Latex','Fontsize',22)

xlabel('Normalized time lag, $\tau\cdot f_{\max}$',...
    'Interpreter','Latex','Fontsize',24);
ylabel('Autocorrelation function, $\tilde{r}_{\mu_i\mu_i}^{(k)}(\tau)$',...
    'Interpreter','Latex','Fontsize',24);
xlim([0 15])
ylim([-0.5 1])

end


%----------------------------------------------------------------------
% f_uncorr.m --------------------------------------------------
% Program for simulates uncorrelated Rayleigh fading waveforms.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Explanation of the input parameters:
% N1: Number of sinusoids for in phase component.
% N2: Number of sinusoids for quadrature component.
% f_max: Maximum Doppler-frequency.
% K: Number of waveforms.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The user will get Figure 10.6. if given parameter is as follows.
% |---------------------------------|-----------------------------------|
% |   N1                            | 20                                |
% |---------------------------------|-----------------------------------|
% |   N2                            | 21                                |
% |---------------------------------|-----------------------------------|
% | sigma_0                         |  1                                |
% |---------------------------------|-----------------------------------|
% |   K                             |  4                                |
% |---------------------------------|-----------------------------------|
% |   fmax                          |  91                               |
% |---------------------------------|-----------------------------------|
% |   PLOT                          |  1                                |
% |---------------------------------|-----------------------------------|

function f_uncorr(N1,N2,K,f_max,PLOT)
f_1_n=zeros(K,N1);
f_2_n=zeros(K,N2);
t=linspace(0,0.1,1000);
for k=1:K
```

```
alpha_1_0(k)=-1*(pi/(4*N1))*(k/(K+2));
alpha_2_0(k)=(pi/(4*N1))*(k/(K+2));

for n=1:N1
f_1_n(k,n)=f_max*cos((pi/(2*N1)).*(n-0.5)+(alpha_1_0(k)));
f_2_n(k,n)=f_max*cos((pi/(2*N2)).*(n-0.5)+(alpha_2_0(k)));
theta_1_n=randn(1,N1);
theta_2_n=randn(1,N2);
end
for q=1:length(t)
mu_1_k(k,q)=sum(cos(2*pi*f_1_n(k,:).*t(q)+theta_1_n));
mu_2_k(k,q)=sum(cos(2*pi*f_2_n(k,:).*t(q)+theta_2_n));
end
mu_k(k,:)=10*log10((abs(sqrt(2/N1)*mu_1_k(k,:)+1j*sqrt(2/N1)*mu_2_k(k,:))));


end

if PLOT==1
plot(t,mu_k(1,:),'k')
hold on
plot(t,mu_k(2,:),'k.')
 plot(t,mu_k(3,:),'k-.')
plot(t,mu_k(4,:),'k--')
set(0,'DefaultAxesFontSize',18);
 H=legend('$\rm{\tilde{\zeta^1}(t)}(dB)$',...
     ' $\rm{\tilde{\zeta^2}(t)}(dB)$','$\rm{\tilde{\zeta^3}(t)}(dB)$',...
     '$\rm{\tilde{\zeta^4}(t)}(dB)$');
set(H,'FontSize',22,'Interpreter','latex');


   ylabel({'Simulated envelope,
$\rm{\tilde{\zeta^k}(t)}(dB)$'},'FontName','times','FontSize',24,'Interpreter
','latex');
   xlabel({'Time,
$t(s)$'},'FontName','times','FontSize',24,'Interpreter','latex');
else
    X=0,
end
end

%--------------------------------------------------------------------------
% ACF_GAU_Uncorr
%--------------------------------------------------------------------------
% The programme computes the autocorrelation function of complex Gaussian
% process of the reference model in comparison with the corresponding
% autocorrelation function of the simulation model.
%
% ACF_GAU_Uncorr(N,alpha01,alpha02,alpha03)
%--------------------------------------------------------------------------
% Explanation of the input parameters:
% N: Number of sinusoids.
% alpha01: The angle-of-rotation in MEDS.
% alpha02: selected value of angle-of-rotation in GMEDS1.
% alpha03: selected value of angle-of-rotation in GMEDS1.
%--------------------------------------------------------------------------
```

```
% To get same result as Fig(10.5)the function used
% ACF_GAU_Uncorr(20,0,pi/160,pi/480)
%---------------------------------------------------------------------------

function ACF_GAU_Uncorr(N,alpha01,alpha02,alpha03)
c=sqrt(2/N);
tau_fmax=linspace(0,15,600);
alpha0=[alpha01;alpha02;alpha03;-alpha02;-alpha03];
r1=zeros(length(alpha0),length(tau_fmax));
r2=zeros(length(alpha0),length(tau_fmax));
for i=1:length(alpha0)
    alpha=pi/(2*N)*((1:N)-1/2)+alpha0(i);
    for n=1:N
        r1(i,:)=r1(i,:)+c^2/2.*cos(2*pi.*tau_fmax.*cos(alpha(n)));
    end
    alpha=pi/(2*N)*((1:N)-1/2)-alpha0(i);
    for n=1:N
        r2(i,:)=r2(i,:)+c^2/2.*cos(2*pi.*tau_fmax.*cos(alpha(n)));
    end
end
r=r1+r2;
r_ref=2*besselj(0,2*pi.*tau_fmax);
plot(tau_fmax,r_ref,'k-','LineWidth',1.5);
hold on
set(0,'DefaultAxesFontSize',18);
plot(tau_fmax,r(1,:),'k-.','LineWidth',1.5);
plot(tau_fmax,r(2,:),'k:','LineWidth',2.5);
plot(tau_fmax,r(3,:),'k--','LineWidth',1.5);
plot(tau_fmax,r(4,:),'k:','LineWidth',2.5);
plot(tau_fmax,r(5,:),'k--','LineWidth',1.5);


legend({'Reference model','Simulation model, MEDS
$(\alpha_{i,0}^{(k)}=0)$',...
    'Simulation model, GMEDS1 $(\alpha_{i,0}^{(k)}=\pm\pi/160)$',...
    'Simulation model, GMEDS1 $(\alpha_{i,0}^{(k)}=\pm\pi/480)$'},...
    'Interpreter','Latex','Fontsize',24)

xlabel('Normalized time lag, $\tau\cdot f_{\max}$','Interpreter','Latex',...
    'Fontsize',22);
ylabel('Autocorrelation function, $\tilde{r}_{\mu_i\mu_i}^{(k)}(\tau)$',...
    'Interpreter','Latex','Fontsize',24);
xlim([0 15])
ylim([-1 2])
```

MATHLAB –PROGRAMS of Frequency Hopping Mobile Radio Channel Simulator

```
%-----------------------------------------------------------------------
% FH_channel_simulator.m
%-----------------------------------------------------------------------
% This program computes the received envelope without and with frequency-
% hopping.
% Parameterization was done based on Method of exact Doppler spread (MEDS)
% Used file:
%-----------------------------------------------------------------------
% FH_channel_simulator(N_1,N_2,sigma_0,f_max,a,F0)
%-----------------------------------------------------------------------
% Explanation of input parameters:
% N_1: The number of sinusoids of Rayleigh fading channel real part
% N_2: The number of sinusoids of Rayleigh fading channel imag. part
% sigma_0: standard deviation
% a: area profile (0.1086 µs)
% fmax: Doppler maximum frequency
% F0: carrier frequency
%-----------------------------------------------------------------------
% The user will get Figure 3.49(a)and(b) if given parameter is as follows.
% |---------------------------------|-----------------------------------|
% |   N_1                           | 20                                |
% |---------------------------------|-----------------------------------|
% |   N_2                           | 21                                |
% |---------------------------------|-----------------------------------|
% | sigma_0                         |  1                                |
% |---------------------------------|-----------------------------------|
% |   a                             |  0.1086 µs                        |
% |---------------------------------|-----------------------------------|
% |   fmax                          |  91                               |
% |---------------------------------|-----------------------------------|
% |   F0                            |  941.2 in MHz used in GSM         |
% |---------------------------------|-----------------------------------|



function FH_channel_simulator(N_1,N_2,sigma_0,f_max,a,F0)
sigma=sqrt(2*sigma_0);
a=a*10^-6;
f0=F0*10^6;
t=0:.00005:0.099;
mu_1_t=zeros(1,length(t));
mu_2_t=zeros(1,length(t));
for l=1:length(t);

       n=(1:N_1)';
       f1_n=f_max*sin(pi/(2*N_1)*(n-1/2));
       c1_n=sigma*sqrt(2/(N_1))*ones(size(f1_n));
       zita1_n=a*log(1./((1-(n-0.5)/N_1)));
theta1=2*pi*f0.*zita1_n;
mu_1_t(l)=sum(c1_n.*cos(2*pi*f1_n*t(l)+theta1));
```

```
    q=(1:N_2)';
        f2_q=f_max*sin(pi/(2*N_2)*(q-1/2));
        c2_q=sigma*sqrt(2/(N_2))*ones(size(f2_q));

 zita2_q=a*log(1./((1-(q-0.5)/N_2)));
 theta2=2*pi*f0*zita2_q;
  mu_2_t(l)=sum(c2_q.*cos(2*pi*f2_q*t(l)+theta2));
end

ebselon_t=20*log10(abs(mu_1_t+1i*(mu_2_t)));
plot(t,ebselon_t,'k')
set(0,'DefaultAxesFontSize',18);
%  H=legend('Simulation envelope with out frequency hopping,
$\rm{\tilde{\zeta}(t)}(dB)$');
%        set(H,'FontSize',22,'Interpreter','latex');
    ylabel({'Simulation envelope,
$\rm{\tilde{\zeta}(t)}(dB)$'},'FontName','times','FontSize',24,'Interpreter',
'latex');
   xlabel({'Time,
$t(s)$'},'FontName','times','FontSize',24,'Interpreter','latex');
hold on
ylim([-40 20])

h=1;
for l=1:length(t);
      n=(1:N_1)';
       f1_n=f_max*sin(pi/(2*N_1)*(n-1/2));
       c1_n=sigma*sqrt(2/(N_1))*ones(size(f1_n));
       zita1_n=a*log(1./(1-((n-0.5)/N_1)));
       if t(l)<(h*0.004615)
theta_d1=2*pi*f0.*zita1_n;
       else
        f0=f0+((-1)^h)*rand(1)*5000000;
theta_d1=2*pi*(f0).*zita1_n;
h=h+1;

       end
mu_1_t(l)=sum(c1_n.*cos(2*pi*f1_n*t(l)+theta_d1));

    q=(1:N_2)';
        f2_q=f_max*sin(pi/(2*N_2)*(q-1/2));
        c2_q=sigma*sqrt(2/(N_2))*ones(size(f2_q));

 zita2_q=a*log(1./(1-((q-0.5)/N_2)));
      theta_d2=2*pi*f0*zita2_q;
  mu_2_t(l)=sum(c2_q.*cos(2*pi*f2_q*t(l)+theta_d2));

end

figure(2)
 ebselon_t=20*log10(abs(mu_1_t+1i*(mu_2_t)));
 set(0,'DefaultAxesFontSize',18);
 plot(t,ebselon_t, 'k')
%  H=legend('Simulation envelope with frequency hopping,
$\rm{\tilde{\zeta}(t)}(dB)$');
```

```
%        set(H,'FontSize',22,'Interpreter','latex');
   ylabel({'Simulation envelope,
$\rm{\tilde{\zeta}(t)}(dB)$'},'FontName','times','FontSize',24,'Interpreter',
'latex');
   xlabel({'Time,
$t(s)$'},'FontName','times','FontSize',24,'Interpreter','latex');
   ylim([-40 20])
end

%---------------------------------------------------------------------
% FH_ch_model_test.m
%---------------------------------------------------------------------
% Program for testing the behaviour of the ACF and CCF for the
% simulation and reference model.
%---------------------------------------------------------------------
% FH_ch_model_test(Ni,sigma_0,f_max,a,PLOT)
% Explanation of the input parameters:
% Ni: Number of scatters.
% a:  Area profile in micro-second.
% f_max:Maximum Doppler-frequency.
%
% PLOT: Choose one of the value out of 1,2,3,4,5 to select the figure as
% describe in below.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% If PLOT==1;
% Figure(1)
% The behaviour for autocorrelation function of the reference model in
% comparisons with the autocorrelation function of simulation model.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------------------------------------------------------------------------%
% If PLOT==2;
% Figure(2)
% Cross-correlation function for frequency hopping reference channel model,
% with changing for time separation (tau) and frequency difference from
% carrier to another(chi).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------------------------------------------------------------------------%
% If PLOT==3;
% Figure(3)
% Cross-correlation function for frequency hopping simulation channel model,
% with changing for time separation (tau) and frequency difference from
% carrier to another (chi).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------------------------------------------------------------------------%
% If PLOT==4;
% Figure(4)
% Cross-correlation function (reference model) comparison with simulation
% model with chosen Ni=20 and area profile (a=0.1086 micro-second
% sigma_0^2=1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------------------------------------------------------------------------%
% If PLOT==5;
% Figure(5)
% Cross-correlation function (reference model)r_mu_1.mu_2.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function FH_ch_model_test(Ni,sigma_0,f_max,a,PLOT)

 a=a*10^-6;
ci_n=sigma_0*sqrt(2/Ni);
n=1:Ni;
fi_n=f_max*sin((pi/(2*Ni))*(n-0.5));
zitai_n=a*log(1./(1-(n-1/2)/Ni));
tau1=0:1:199;
rmimi_sim1=zeros(1,200);
rmimi_ref=BesselJ(0,2*pi*f_max.*tau1/1000);
X_i=linspace(0,5,4*Ni);

rmimih=sigma_0./(1+(2*pi*a*1000000*X_i).^2);
for i=1:200;
rmimi_sim=cos(2*pi*f_max*tau1(i)/1000.*sin((pi/(2*Ni))*(n-0.5)));
rmimi_sim1(i)=sum(rmimi_sim)/Ni;
end

X_i=linspace(0,5,4*Ni);
tau=linspace(0,40,4*Ni);
%testing CCF_FH

CCF_FHsim=zeros(length(X_i),length(tau));
CCF_FHref=zeros(length(X_i),length(tau));
CCF_FHrefm1m2=zeros(length(X_i),length(tau));

for g=1:length(X_i)
    for q=1:length(tau)
       CCF_FHsim(q,g)=sum(0.5*(ci_n.^2).*cos((2*pi.*fi_n.*tau(g)/1000)+...
           (2*pi.*zitai_n.*1000000*X_i(q))));
       CCF_FHref(q,g)=sigma_0*BesselJ(0,2*pi*f_max.*tau(g)/1000)./...
           (1+(2*pi*a*1000000*X_i(q)).^2);
    end
end

if PLOT==1;
plot(tau1,rmimi_ref,'k')
xlim([0 150])
ylim([-.6 1])
hold on
plot(tau1,rmimi_sim1,'k-.')

xlim([0 170])
    ylim([-.6 1])
    set(0,'DefaultAxesFontSize',18);
    legend({['$\rm{{r}_{\mu_i\mu_i}(\tau)}$ Reference model']...
    ,['$\rm{\tilde{r}_{\mu_i\mu_i}(\tau)}$ Simulation model with Ni = '...
       ,num2str(Ni)]},'FontName','Arial','Location','NorthEast'...
        ,'FontSize',24,'Interpreter','latex');

    xlabel('Time seperation, $\rm{\tau}$ (s)','HorizontalAlignment',...
       'center','FontSize',...
           24,'Interpreter','latex')
       ylabel({'ACF, $\rm{{r}_{\mu_i\mu_i}(\tau)}$'},'FontSize',...
           24,'Interpreter','latex')
```

```
elseif PLOT==2
figure(2)
surf(tau,X_i,real(CCF_FHref));
xlim([0 40])
ylim([0 5])
zlim([-.5 1])
set(gca,'YDir','reverse');
  set(0,'DefaultAxesFontSize',18);
   %legend(['Simulation model'],'FontName','times','Location'...
       %,'NorthEast','FontSize',18,'Interpreter','latex');
   ylabel({'${\chi}(MHz)$'},'FontName','times'...
       ,'FontSize',24,'Interpreter','latex');
   xlabel({'${\tau}(ms)$'},'FontName','times','FontSize',24,'Interpreter'...
       ,'latex');
   zlabel({'CCF,$\rm{{r}_{\mu_1\mu_2}(\tau,{\chi})}$'},'FontName',...
           'times','FontSize',24,'Interpreter','latex')

  elseif PLOT==3
figure(3)
surf(tau,X_i,CCF_FHsim);
xlim([0 40])
ylim([0 5])
zlim([-.5 1])
set(gca,'YDir','reverse');
  set(0,'DefaultAxesFontSize',18);

 ylabel({'${\chi}(MHz)$'},'FontName','times','FontSize',24,...
     'Interpreter','latex');
   xlabel({'${\tau}(ms)$'},'FontName','times','FontSize',24,...
       'Interpreter','latex');
   zlabel({'CCF,$\rm{\tilde{r}_{\mu_1\mu_2}(\tau,{\chi})}$'},'FontName',...
           'times','FontSize',24,'Interpreter','latex')
  elseif PLOT==4
      figure (4)
       plot(X_i,CCF_FHref(:,1),'k')
       hold on
       plot(X_i,CCF_FHsim(:,1),'k.-')
        legend({['$\rm{{r}_{\mu_i\mu_i}(0,\chi)}$ Reference model']...
    ,['$\rm{\tilde{r}_{\mu_i\mu_i}(0,\chi)}$ Simulation model with Ni = '...
       ,num2str(Ni)]},'FontName','Arial','Location','NorthEast'...
        ,'FontSize',20,'Interpreter','latex');
     xlabel({'Frequency seperation, ${\chi}(MHz)$'},'FontName','times'...
       ,'FontSize',24,'Interpreter','latex');
   ylabel({'CCF, $\rm{{r}_{\mu_i\mu_i}(0,{\chi})}$'},'FontName',...
           'times','FontSize',24,'Interpreter','latex')
     xlim([0 5])
     ylim([0 1])
elseif PLOT==5
    figure(5)
    X_i1=linspace(0,40,4*Ni);
    for g=1:length(X_i1)
    for q=1:length(tau)
        CCF_FHrefm1m2(q,g)=-2*pi*a*1000000*X_i1(q)*sigma_0*...
      BesselJ(0,2*pi*f_max.*tau(g)/1000)./(1+(2*pi*a*1000000*X_i1(q)).^2);
    end

end
```

```
    surf(tau,X_i1,CCF_FHrefm1m2);
    set(gca,'YDir','reverse');
  set(0,'DefaultAxesFontSize',18);
   %legend(['Simulation model'],'FontName','times','Location'...
       %,'NorthEast','FontSize',18,'Interpreter','latex');
   ylabel({'${\chi}(MHz)$'},'FontName','times'...
       ,'FontSize',24,'Interpreter','latex');
   xlabel({'${\tau}(ms)$'},'FontName','times','FontSize',24,'Interpreter'...
       ,'latex');
   zlabel({'CCF,$\rm{{r}_{\mu_1\mu^,_2}(\tau,{\chi})}$'},'FontName',...
           'times','FontSize',24,'Interpreter','latex')
xlim([0 40])
ylim([0 40])
zlim([-.6 1])
else
     disp('enter a valid figure number from 1-5');
end
end
```

APPENDIX 4

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
```

APPENDIX 5

SUPPLEMENTARY MATLAB PROGRAMS OF THE TOOLBOX

MATHLAB –PROGRAMS for Computation of PDF and CDF for Different Distributions

```
%-------------------------------------------------------------------------
% UniformPDF.m
%
% This program computes the Uniform distribution which distributed in
% interval [a,b]. The equation of the Uniform distribution is given in
% (2.29) [2].
% Used file :
%-------------------------------------------------------------------------
% UniformCDF(a,b)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
% a,b: Uniformly distributed interval [a,b] of the density function.

function UniformPDF(a,b)

%clear all;
k1=a;k2=b;
temp=1;
for x=a-2:0.01:b+2
    if x>=a  && x<b
        p(1,temp)=1;
    else
        p(1,temp)=0;
    end
    temp=temp+1;
end
pp=p./(k2-k1);
x=a-2:0.01:b+2;

figure;
grid on;%
```

```
hold on;
ylabel('PDF, $p_{\theta}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',20)
xlabel('$\theta$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',20)
title ('Uniform distribution ','HorizontalAlignment','center',...
    'interpreter','latex','FontSize',20)
plot(x,pp,'black','LineStyle','-','LineWidth',1.5);



%------------------------------------------------------------------------
% UniformCDF.m
%
% This program computes the cumulative distribution of the Uniform
% distribution. The equation of the Uniform distribution is given in
% (2.29) [2].
% Used file :
%------------------------------------------------------------------------
% UniformCDF(a,b)
%------------------------------------------------------------------------
% Explanation of the input parameters:
% a,b: Uniformly distributed interval [a,b] of the density function.
function UniformCDF(a,b)

temp=1;
for x=a-2:0.01:b+2
    if x>=a && x<b
        p(1,temp)=(x-a)/(b-a); %y=(x-a)/(b-a)
             elseif x>=b
         p(1,temp)=1;
    else
        p(1,temp)=0;
            end
    temp=temp+1;
end

x=a-2:0.01:b+2;

figure;
grid on;%
hold on;
xlabel (' X axis--->')
ylabel ('Fx(X)')
plot(x,p,'black','LineStyle','-','LineWidth',1.5);
ylabel('CDF, $F_{\theta}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',20)
xlabel('$\theta$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',20)
title ('Cumulative distribution function ','HorizontalAlignment',...
    'center','interpreter','latex','FontSize',20)
```

```
%-------------------------------------------------------------------------
% GaussianPDF.m
%
% This program computes the Gaussian distribution for different values of
% the mean value and variance sigma_0^2. The equation for the Gaussian
% distribution is given in (2.30) [2].
% Used file :
%-------------------------------------------------------------------------
% GaussianPDF(mu,sigma02)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
% mu: mean value of the distribution.
% sigma02: variance sigma_0^2 of the distribution.

function  GaussianPDF(mu,sigma02)

x=-5:0.1:5;
y=(1/sqrt(2*pi*sigma02)).*exp(-((x-mu).^2./(2*sigma02)));
plot(x,y,'black','LineStyle','-','LineWidth',1.5);
ylabel('PDF, $p_{\mu}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
title ('the Gaussian distribution ','HorizontalAlignment','center',...
    'interpreter','latex','FontSize',20)



%-------------------------------------------------------------------------
% GaussianCDF.m
%
% This program computes the Gaussian cumulative distribution for different
% values of the mean value and variance sigma_0^2. The equation for the
% Gaussian cumulative distribution is given in (2.33) [2].
% Used file :
%-------------------------------------------------------------------------
% GaussianCDF(mu,sigma02)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
% mu:mean value of the distribution.
% sigma02:variance sigma_0^2 of the distribution.

function  GaussianCDF(mu,sigma02)

x=-4:0.1:4;
y=0.5*(1+erf((x-mu)./sqrt(2*sigma02)));
plot(x,y,'black','LineStyle','-','LineWidth',1.5);
ylabel('CDF, $F_{\mu}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
title ('The Gaussian cumulative distribution ','HorizontalAlignment',...
    'center','interpreter','latex','FontSize',20)
```

```
%-------------------------------------------------------------------------
% RayPDF.m
%
% This program computes the Rayleigh distribution for different values of
% the variance sigma_0^2. The equation for the Rayleigh distribution is
% given in (2.40) [2].
% Used file :
%-------------------------------------------------------------------------
% RayPDF(sigma21,sigma22,sigma23)
%-------------------------------------------------------------------------
% Explanation of the input parameters:
% sigma21, sigma22, sigma23: Different values of sigma_0^2 characterizing
%                            the Rayleigh distribution.
% The user will get Figure 2.1.(a) if he chose the parameter sigma21,
% sigma22 and sigma23 as follows.
% |--------------------------------|-----------------------------------|
% | sigma21                        |  1                                |
% |--------------------------------|-----------------------------------|
% | sigma22                        |  2                                |
% |--------------------------------|-----------------------------------|
% | sigma23                        |  3                                |
% |--------------------------------|-----------------------------------|

function RayPDF(sigma21,sigma22,sigma23)
x=0:0.001:6;
X1=(x/sigma21).*exp((-x.^2)/(2*sigma21));
X2=(x/sigma22).*exp((-x.^2)/(2*sigma22));
X3=(x/sigma23).*exp((-x.^2)/(2*sigma23));
axes('FontSize',16);
plot(x,X1,'black','LineStyle','.','LineWidth',1.5);
hold on
plot(x,X2,'black','LineStyle','-','LineWidth',1.5);
plot(x,X3,'black','LineStyle','-.','LineWidth',1.5);
ylabel('PDF, $p_{\zeta}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
 h=legend(['$\sigma^2=$',num2str(sigma21)],['$\sigma^2=$',...
     num2str(sigma22)],['$\sigma^2=$',num2str(sigma23)]);
set(h,'FontSize',22,'Interpreter','latex');
hold off


%----------------------------------------------------------------------
% RayCDF.m
%
% This program computes the Rayleigh cumulative distribution for
% different values of the variance sigma_0^2. The equation for the Rayleigh
% cumulative distribution is given in (2.41) [2].
% Used file :
%----------------------------------------------------------------------
% RayCDF(sigma21,sigma22,sigma23)
%----------------------------------------------------------------------
% Explanation of the input parameters:
% sigma21, sigma22, sigma23: Different values of sigma_0^2 characterizing
%                             Rayleigh distribution.
% The user will get Figure 2.1.(b) if he chose the parameter sigma21,
% sigma22 and sigma23 as follows.
% |--------------------------------|----------------------------------|
% | sigma21                        |  1                               |
% |--------------------------------|----------------------------------|
% | sigma22                        |  2                               |
% |--------------------------------|----------------------------------|
% | sigma23                        |  3                               |
% |--------------------------------|----------------------------------|

function RayCDF(sigma21,sigma22,sigma23)
x=0:0.001:6;
X1=1-exp((-x.^2)/(2*sigma21));
X2=1-exp((-x.^2)/(2*sigma22));
X3=1-exp((-x.^2)/(2*sigma23));
axes('FontSize',16);
plot(x,X1,'black','LineStyle','.','LineWidth',1.5);
hold on
plot(x,X2,'black','LineStyle','-','LineWidth',1.5);
plot(x,X3,'black','LineStyle','-.','LineWidth',1.5);
% title('{\itAe}^{-\alpha\itt}sin\beta{\itt} \alpha<<\beta')
% text(7,0.2,'\bfstring_2','EdgeColor','black');
ylabel('CDF, $F_{\zeta}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
% text(2,0.2,'$\sigma_0^2=1$',...
%     'interpreter','latex','FontSize',22,'Margin',100000);
% text(3,0.2,'$\sigma_0^2=2$',...
%     'interpreter','latex','FontSize',22,'Margin',100000);
% text(4,0.2,'$\sigma_0^2=3$',...
%     'interpreter','latex','FontSize',22,'Margin',100000);

 h=legend(['$\sigma^2=$',num2str(sigma21)],['$\sigma^2=$',...
     num2str(sigma22)],['$\sigma^2=$',num2str(sigma23)]);
set(h,'FontSize',22,'Interpreter','latex');
hold off



%--------------------------------------------------------------------
% RicePDF.m
%
% This program computes the Rice distribution (PDF) for different
% values of rho and variance sigma_0^2. The equation for the Rice
% distribution is given in (2.44) [2].
% used file:
%---------------------------------------------------------------------
% RicePDF(ro1,ro2,ro3,sigma_02)
%---------------------------------------------------------------------
% Explanation of the input parameters:
% ro1, ro2, ro3: Different values of rho characterizing the Rice
%                distribution.
% sigma_02: Input value for the sigma_0^2 of the distribution.
%
% The user will get Figure 2.2.(a) if he chose the parameter sigma21,
% sigma22 and sigma23 as follows.
% |--------------------------------|--------------------------------|
% | ro1                             |  0                               |
% |---------------------------------|----------------------------------|
% | ro2                             |  1                               |
% |---------------------------------|----------------------------------|
% | ro3                             |  2                               |
% |---------------------------------|----------------------------------|
% | sigma_02                        |  1                               |
% |---------------------------------|----------------------------------|

function RicePDF(ro1,ro2,ro3,sigma_02)
x=0:0.001:5;
X1=x.*exp((-(x.^2+ro1^2))/(2./sigma_02)).*besseli(0,x*ro1./sigma_02)./...
    sigma_02;
X2=x.*exp((-(x.^2+ro2^2))/(2./sigma_02)).*besseli(0,x*ro2./sigma_02)./...
    sigma_02;
X3=x.*exp((-(x.^2+ro3^2))/(2./sigma_02)).*besseli(0,x*ro3./sigma_02)./...
    sigma_02;
axes('FontSize',16);
plot(x,X1,'black','LineStyle','.','LineWidth',1.5);
hold on
plot(x,X2,'black','LineStyle','-','LineWidth',1.5);
plot(x,X3,'black','LineStyle','-.','LineWidth',1.5);
% title('{\itAe}^{-\alpha\itt}sin\beta{\itt} \alpha<<\beta')
 text(3,0.1,['$\sigma_0^2=$',num2str(sigma_02)],...
     'interpreter','latex','FontSize',22,'Margin',100000);
ylabel('PDF, $p_{\xi}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
h=legend(['$\rho=$',num2str(ro1)],['$\rho=$',num2str(ro2)],['$\rho=$',...
    num2str(ro3)]);
set(h,'FontSize',22,...
  'Interpreter','latex');
hold off


%---------------------------------------------------------------------------
% RiceCDF.m
%
% This program computes the Rice cumulative distribution for different
% values % of rho and the variance sigma_0^2. Equation for the Rice
% Distribution is given in (2.45) [2].
% Used file:
%---------------------------------------------------------------------------
% RiceCDF(ro1,ro2,ro3,sigma_02)
%---------------------------------------------------------------------------
% Explanation of the input parameters:
% ro1, ro2, ro3: Different values of rho characterizing the Rice
%                distribution.
% sigma_02: Input value for the sigma_0^2 of the distribution.
%
% The user will get Figure 2.2.(b) if he chose the parameter ro1,
% ro2, ro3 and sigma_02 as follows.
% |---------------------------------|----------------------------------|
% | ro1                             |  0                               |
% |---------------------------------|----------------------------------|
% | ro2                             |  1                               |
% |---------------------------------|---------------------------------|
% | ro3                             |  2                              |
% |---------------------------------|---------------------------------|
% | sigma_02                        |  1                              |
% |---------------------------------|---------------------------------|

function RiceCDF(ro1,ro2,ro3,sigma_02)
x=0:0.001:6;
X1=1-marcumq(ro1/sqrt(sigma_02),x./sqrt(sigma_02));
X2=1-marcumq(ro2/sqrt(sigma_02),x./sqrt(sigma_02));
X3=1-marcumq(ro3/sqrt(sigma_02),x./sqrt(sigma_02));
axes('FontSize',16);
plot(x,X1,'black','LineStyle','.','LineWidth',1.5);
hold on
plot(x,X2,'black','LineStyle','-','LineWidth',1.5);
plot(x,X3,'black','LineStyle','-.','LineWidth',1.5);
% title('{\itAe}^{-\alpha\itt}sin\beta{\itt} \alpha<<\beta')
  text(3,0.1,['$\sigma_0^2=$',num2str(sigma_02)],...
     'interpreter','latex','FontSize',22,'Margin',100000);
ylabel('CDF, $F_{\xi}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
h=legend(['$\rho=$',num2str(ro1)],['$\rho=$',num2str(ro2)],['$\rho=$',...
    num2str(ro3)]);
set(h,'FontSize',22,...
  'Interpreter','latex');
hold off



%----------------------------------------------------------------------
% LognormalPDF.m
%
% This program computes the Lognormal distribution for different value of
% the variance sigma_mu^2 and expected value m_mu [2].
% Used m-files:
%----------------------------------------------------------------------
% LognormalPDF(sigma21,sigma22,sigma23,m_mu)
%----------------------------------------------------------------------
% Explanation of the input parameters:
% sigma21,sigma22,sigma23:  Different values of sigma_mu^2  characterizing
%                           the distribution.
% m_mu: Input value for the m_mu of the distribution.
% The user will get Figure 2.3.(a) if he chose given  parameter as follows.
% |---------------------------------|-----------------------------------|
% | sigma21                         |  1                                |
% |---------------------------------|-----------------------------------|
% | sigma22                         |  2                                |
% |---------------------------------|-----------------------------------|
% | sigma23                         |  3                                |
% |---------------------------------|-----------------------------------|
% | m_mu                            |  0                                |
% |---------------------------------|-----------------------------------|

function LognormalPDF(sigma21,sigma22,sigma23,m_mu)
x=0:0.001:12;
```

```
%sigma2=sigma21;
y1=(1./(sqrt(2*pi.*sigma21).*x)).*(exp(-((log(x)-m_mu).^2)./(2*sigma21)));
y2=(1./(sqrt(2*pi.*sigma22).*x)).*(exp(-((log(x)-m_mu).^2)./(2*sigma22)));
y3=(1./(sqrt(2*pi.*sigma23).*x)).*(exp(-((log(x)-m_mu).^2)./(2*sigma23)));
hold on

plot(x,y1,'black','LineStyle','--','LineWidth',1.5)
plot(x,y2,'black','LineStyle','.','LineWidth',1.5)
plot(x,y3,'black','LineStyle','-','LineWidth',1.5)

text(4,0.3,['$m_{\mu}=$',num2str(m_mu)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
ylabel('PDF, $p_{\lambda}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)

legend(['\sigma_{\mu}^2 = ',num2str(sigma21)],['\sigma_{\mu}^2 = ',...
    num2str(sigma22)],['\sigma_{\mu}^2 = ',num2str(sigma23)],...
    'FontName',24,'Arial','Location','NorthEast','FontSize',24,...
    'Interpreter','latex');

hold off


%----------------------------------------------------------------------
% LognormalCDF.m
%
% This program computes the Lognormal cumulative distribution for different
% values of variance sigma_mu^2 and expected value m_mu. The equation for
% the Rice cumulative distribution is given in (2.52) [2].
% Used file:
%----------------------------------------------------------------------
% LognormalCDF(sigma21,sigma22,sigma23,m_mu)
%----------------------------------------------------------------------
% Explanation of input parameters:
%
% sigma21,sigma22,sigma23:  Different values for 'sigma_mu^2'  of the
%                           distribution.
% m_mu: Input value for the m_mu of the distribution.
% The user will get Figure 2.3.(b) if he chose given  parameter as follows.
% |--------------------------------|----------------------------------|
% | sigma21                        |  1                               |
% |--------------------------------|----------------------------------|
% | sigma22                        |  2                               |
% |--------------------------------|----------------------------------|
% | sigma23                        |  3                               |
% |--------------------------------|----------------------------------|
% | m_mu                           |  0                               |
% |--------------------------------|----------------------------------|

function LognormalCDF(sigma21,sigma22,sigma23,m_mu)
x=0:0.001:12;
%sigma2=sigma21;
y1=0.5+0.5.*erf(((log (x))-m_mu)./sqrt(2*sigma21));
y2=0.5+0.5.*erf(((log (x))-m_mu)./sqrt(2*sigma22));
```

```
y3=0.5+0.5.*erf(((log (x))-m_mu)./sqrt(2*sigma23));
hold on

plot(x,y1,'black','LineStyle','--','LineWidth',1.5)
plot(x,y2,'black','LineStyle','.','LineWidth',1.5)
plot(x,y3,'black','LineStyle','-','LineWidth',1.5)

text(4,0.3,['$m_{\mu}=$',num2str(m_mu)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
ylabel('CDF, $F_{\lambda}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)

legend(['\sigma_{\mu}^2 = ',num2str(sigma21)],['\sigma_{\mu}^2 = ',...
    num2str(sigma22)],['\sigma_{\mu}^2 = ',num2str(sigma23)],...
    'FontName',24,'Arial','Location','NorthEast','FontSize',24,...
    'Interpreter','latex');

hold off


%----------------------------------------------------------------------
% SuzukiPDF.m
%
% This program computes the Suzuki distribution for different
% values of variance sigma_mu^2','sigma_0^2' and 'm_mu'. The equation
% for the Suzuki distribution is given in (2.55) [2].
% used file:
%----------------------------------------------------------------------
% SuzukiPDF(sigma21,sigma22,sigma23,sigma_02,m)
%----------------------------------------------------------------------
% Explanation of input parameters:
% sigma21,sigma22,sigma23: Different input values of 'sigma_mu^2'
%                          characterizing the Suzuki distribution
% sigma_0: Input value for the sigma_0^2
% m: Input value for m_mu of the Suzuki distribution
%
% The user will get Figure 2.4.(a) if he chose the parameter sigma21,
% sigma22,sigma23 and m as follows.
% |-----------------------------|-------------------------------------|
% | sigma21                     | 0.001(too small value close to 0)   |
% |-----------------------------|-------------------------------------|
% | sigma22                     |  1/3                                |
% |-----------------------------|-------------------------------------|
% | sigma23                     |  1/2                                |
% |-----------------------------|-------------------------------------|
% | sigma_0                     |  1  or 3                            |
% |-----------------------------|-------------------------------------|
% | m                           |  0                                  |
% |-----------------------------|-------------------------------------|
function SuzukiPDF(sigma21,sigma22,sigma23,sigma_02,m)
warning off MATLAB:divideByZero
save data
z=0.01:0.01:25;
x=0.01:0.01:10;
sigma2=sigma21;
```

```
y1=zeros(1,length(x));

for i=1:length(x)
y1(i)=trapz(z,exp(-(x(i)^2)./(2*(z.^2)*sigma_02)).*exp(-((log(z)-m).^2)/...
    (2.*sigma2))./(z.^3));
y1(i)=x(i).*y1(i)./(sqrt(2*pi*sigma2)*sigma_02);
end
sigma2=sigma22;
y2=zeros(1,length(x));
for i=1:length(x)
y2(i)=trapz(z,exp(-(x(i)^2)./(2*(z.^2)*sigma_02)).*exp(-((log(z)-m).^2)/...
    (2.*sigma2))./(z.^3));
y2(i)=x(i).*y2(i)./(sqrt(2*pi*sigma2)*sigma_02);
end
sigma2=sigma23;
y3=zeros(1,length(x));
for i=1:length(x)
y3(i)=trapz(z,exp(-(x(i)^2)./(2*(z.^2)*sigma_02)).*exp(-((log(z)-m).^2)/...
    (2.*sigma2))./(z.^3));
y3(i)=x(i).*y3(i)./(sqrt(2*pi*sigma2)*sigma_02);
end

x=[0,x];y1=[0,y1];y2=[0,y2];y3=[0,y3];
axes('FontSize',16);
plot(x,y1,'black','LineStyle','.','LineWidth',1.5);
hold on
plot(x,y2,'black','LineStyle','-','LineWidth',1.5);
plot(x,y3,'black','LineStyle','-.','LineWidth',1.5);
text(5,0.3,['$m_{\mu}=$',num2str(m)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
ylabel('PDF, $p_{\eta}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
h=legend(['$\sigma_{\mu}^2=$',num2str(sigma21)],['$\sigma_{\mu}^2=$',...
    num2str(sigma22)],['$\sigma_{\mu}^2=$',num2str(sigma23)]);
set(h,'FontSize',22,...
  'Interpreter','latex');

hold off


%-----------------------------------------------------------------------
% SuzukiCDF.m
%
% This program computes the Suzuki cumulative distribution for different
% values of variance sigma_mu^2,sigma_0^2 and m_mu. The equation
% for the Suzuki distribution is given in (2.55) [2].
% used file:
%-----------------------------------------------------------------------
% SuzukiCDF(sigma21,sigma22,sigma23,sigma_0,m)
%-----------------------------------------------------------------------
% Explanation of the input parameters:
% sigma21,sigma22,sigma23: Different input values for 'sigma_mu^2'  of
%                          the Suzuki distribution
% sigma_0: Input value for the sigma_0^2
% m: Input value for m_mu of the Suzuki distribution
```

```
%
% The user will get Figure 2.4.(b) if he chose the parameter sigma21,
% sigma22,sigma23, sigma_o and m as follows.
% |---------------------------------|-------------------------------------|
% | sigma21                         | 0.001(too small value close to 0)   |
% |---------------------------------|-------------------------------------|
% | sigma22                         |  1/3                                |
% |---------------------------------|-------------------------------------|
% | sigma23                         |  1/2                                |
% |---------------------------------|-------------------------------------|
% | sigma_0                         |  1  or 3                            |
% |---------------------------------|-------------------------------------|
% | m                               |  0                                  |
% |---------------------------------|-------------------------------------|
function SuzukiCDF(sigma21,sigma22,sigma23,sigma_0,m)
warning off MATLAB:divideByZero

z=0.01:0.01:25;
x=0.01:0.01:10;

%sigma02=3;
sigma2=sigma21;
y1=zeros(1,length(x));
for i=1:length(x)
y1(i)=trapz(z,exp(-(x(i)^2)./(2*(z.^2)*sigma_0)).*exp(-((log(z)-m).^2)/...
    (2.*sigma2))./(z.^3));
y1(i)=x(i).*y1(i)./(sqrt(2*pi*sigma2)*sigma_0);
end

sigma2=sigma22;
y2=zeros(1,length(x));
for i=1:length(x)
y2(i)=trapz(z,exp(-(x(i)^2)./(2*(z.^2)*sigma_0)).*exp(-((log(z)-m).^2)/...
    (2.*sigma2))./(z.^3));
y2(i)=x(i).*y2(i)./(sqrt(2*pi*sigma2)*sigma_0);
end
sigma2=sigma23;
y3=zeros(1,length(x));
for i=1:length(x)
y3(i)=trapz(z,exp(-(x(i)^2)./(2*(z.^2)*sigma_0)).*exp(-((log(z)-m).^2)/...
    (2.*sigma2))./(z.^3));
y3(i)=x(i).*y3(i)./(sqrt(2*pi*sigma2)*sigma_0);
end

x=[0,x];y1=[0,y1];y2=[0,y2];y3=[0,y3];
Y1=0;
for i=2:length(x)
    temp=trapz(x(1:i),y1(1:i));
    Y1=[Y1,temp];
end
Y2=0;
for i=2:length(x)
    temp=trapz(x(1:i),y2(1:i));
    Y2=[Y2,temp];
end
Y3=0;
for i=2:length(x)
```

```
    temp=trapz(x(1:i),y3(1:i));
    Y3=[Y3,temp];
end
axes('FontSize',16);
plot(x,Y1,'black','LineStyle','.','LineWidth',1.5);
hold on
plot(x,Y2,'black','LineStyle','-','LineWidth',1.5);
plot(x,Y3,'black','LineStyle','-.','LineWidth',1.5);
text(5,0.3,['$m_{\mu}=$',num2str(m)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
ylabel('CDF, $F_{\eta}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
h=legend(['$\sigma_{\mu}^2=$',num2str(sigma21)],['$\sigma_{\mu}^2=$',...
    num2str(sigma22)],['$\sigma_{\mu}^2=$',num2str(sigma23)]);
set(h,'FontSize',22,...
  'Interpreter','latex');
hold off


%----------------------------------------------------------------------
% Nakagami_mPDF.m
%
% This program computes the Nakagami-m  distribution for different
% values of 'm' and 'Omega'. The equation  for the Nakagami-m cumulative
% distribution is given in (2.58) [2].
% Used file:
%----------------------------------------------------------------------
% Nakagami_mCDF(m1,m2,m3,m4,omega)
%----------------------------------------------------------------------
% Explanation of input parameters:
% m1, m2, m3, m4: Different values of m characterizing the Nakagami-m
%                 distribution.
% omega: Input value for the Omega of the Nakagami-m  distribution
%
% The user will get Figure 2.5.(a) if he chose the parameters m1, m2,
% m3, m4 and m as follows.
% |--------------------------------|----------------------------------|
% | m1                             |  1/2                             |
% |--------------------------------|----------------------------------|
% | m2                             |  1                               |
% |--------------------------------|----------------------------------|
% | m3                             |  2                               |
% |--------------------------------|----------------------------------|
% | m4                             |  3                               |
% |--------------------------------|----------------------------------|
% | omega                          |  1                               |
% |--------------------------------|----------------------------------|

function Nakagami_mPDF(m1,m2,m3,m4,omega)
x=0:0.001:4;
m=m1;
y1=((2*m^m).*x.^(2*m-1).*exp(-(m/omega).*x.^2))./(gamma(m).*omega^m);
m=m2;
y2=((2*m^m).*x.^(2*m-1).*exp(-(m/omega).*x.^2))./(gamma(m).*omega^m);
m=m3;
```

```
y3=((2*m^m).*x.^(2*m-1).*exp(-(m/omega).*x.^2))./(gamma(m).*omega^m);
m=m4;
y4=((2*m^m).*x.^(2*m-1).*exp(-(m/omega).*x.^2))./(gamma(m).*omega^m);
axes('FontSize',16);
plot(x,y1,'black','LineStyle','-','LineWidth',1.5);
hold on
plot(x,y2,'black','LineStyle','-','LineWidth',1.5);
plot(x,y3,'black','LineStyle','-','LineWidth',1.5);
plot(x,y4,'black','LineStyle','-','LineWidth',1.5);
% text(3,1,'\Omega==1','Color','w','EdgeColor','black',...
%     'interpreter','tex','FontSize',22,'Margin',5);
text(3,1,['$\Omega=$',num2str(omega)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
ylabel('PDF, $p_{\omega}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
text(1.5,0.6,['$m=$',num2str(m1)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(2.3,0.6,['$m=$',num2str(m2)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(3,0.6,['$m=$',num2str(m3)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(3.5,0.6,['$m=$',num2str(m4)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
hold off



%----------------------------------------------------------------------
% Nakagami_mCDF.m
%
% This program computes the Nakagami-m cumulative distribution for
% different values of 'm' and 'Omega'. The equation for the Nakagami-m
% cumulative distribution is given in (2.60) [2].
% Used file:
%----------------------------------------------------------------------
% Nakagami_mCDF(m1,m2,m3,m4,omega)
%----------------------------------------------------------------------
% Explanation of input parameters:
% m1, m2, m3, m4: Different values of 'm' of characterizing
%                 the Nakagami-m dstribution.
% omega: Input value for the Omega of the Nakagami-m  distribution
%
% The user will get Figure 2.5.(b) if he chose the parameters m1, m2,
% m3, m4 and m as follows.
% |---------------------------------|-----------------------------------|
% | m1                              |  1/2                              |
% |---------------------------------|-----------------------------------|
% | m2                              |  1                                |
% |---------------------------------|-----------------------------------|
% | m3                              |  2                                |
% |---------------------------------|-----------------------------------|
% | m4                              |  3                                |
% |---------------------------------|-----------------------------------|
% | omega                           |  1                                |
% |---------------------------------|-----------------------------------|
function Nakagami_mCDF(m1,m2,m3,m4,omega)
x=0:0.001:4;
m=m1;
y1=((2*m^m).*x.^(2*m-1).*exp(-(m/omega).*x.^2))./(gamma(m).*omega^m);
Y1=0;
for i=2:length(x)
    temp=trapz(x(1:i),y1(1:i));
    Y1=[Y1,temp];
end
m=m2;
y2=((2*m^m).*x.^(2*m-1).*exp(-(m/omega).*x.^2))./(gamma(m).*omega^m);
Y2=0;
for i=2:length(x)
    temp=trapz(x(1:i),y2(1:i));
    Y2=[Y2,temp];
end
m=m3;
y3=((2*m^m).*x.^(2*m-1).*exp(-(m/omega).*x.^2))./(gamma(m).*omega^m);
Y3=0;
for i=2:length(x)
    temp=trapz(x(1:i),y3(1:i));
    Y3=[Y3,temp];
end
m=m4;
y4=((2*m^m).*x.^(2*m-1).*exp(-(m/omega).*x.^2))./(gamma(m).*omega^m);
Y4=0;
for i=2:length(x)
    temp=trapz(x(1:i),y4(1:i));
    Y4=[Y4,temp];
end
axes('FontSize',16);
plot(x,Y1,'black','LineStyle','-','LineWidth',1.5);
hold on
plot(x,Y2,'black','LineStyle','-','LineWidth',1.5);
plot(x,Y3,'black','LineStyle','-','LineWidth',1.5);
plot(x,Y4,'black','LineStyle','-','LineWidth',1.5);
% title('{\itAe}^{-\alpha\itt}sin\beta{\itt} \alpha<<\beta')
text(3,0.1,['$\Omega=$',num2str(omega)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
ylabel('CDF, $F_{w}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
text(1.5,0.6,['$m=$',num2str(m1)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(2.3,0.6,['$m=$',num2str(m2)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(3,0.6,['$m=$',num2str(m3)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(3.5,0.6,['$m=$',num2str(m4)],...
    'interpreter','latex','FontSize',22,'Margin',100000);

hold off
```

```
%-----------------------------------------------------------------------
% Nakagami_qPDF.m
%
% This program computes the Nakagami-q distribution for
% different values of 'q' and 'Omega_x'.The equation given for the
% Nakagami-q distribution is given in (2.67) [2].
% Used file:
%-----------------------------------------------------------------------
% Nakagami_qPDF(q1,q2,q3,q4,omega)
%-----------------------------------------------------------------------
% Explanation of input parameters:
% q1, q2, q3, q4: Different values of q characterizing the Nakagami-q
%                 distribution.
% omegax: Input value for the Omega_x of the Nakagami-q distribution
%
% The user will get Figure 2.6.(a) if he chose parameters q1, q2, q3,
% q4 and omegax as follows.
% |---------------------------------|-----------------------------------|
% | q1                              |  1/2                              |
% |---------------------------------|-----------------------------------|
% | q2                              |  1                                |
% |---------------------------------|-----------------------------------|
% | q3                              |  2                                |
% |---------------------------------|-----------------------------------|
% | q4                              |  3                                |
% |---------------------------------|-----------------------------------|
% | omegax                          |  1                                |
% |---------------------------------|-----------------------------------|

function Nakagami_qPDF(q1,q2,q3,q4,omegax)
x=0:0.001:4;
q=q1;
y1=2/sqrt(2*pi*2).*exp(-x.^2/(4));
q=q2;
y2=x.*(1+q^2)/(q*omegax).*exp(-(x.^2)*((1+q^2)^2)/(4*q^2*omegax)).* ...
    besseli(0,(x.^2)*(1-q^4)/(4*q^2*omegax));

q=q3;
y3=x.*(1+q^2)/(q*omegax).*exp(-(x.^2)*((1+q^2)^2)/(4*q^2*omegax)).* ...
    besseli(0,(x.^2)*(1-q^4)/(4*q^2*omegax));

q=q4;
y4=x.*(1+q^2)/(q*omegax).*exp(-(x.^2)*((1+q^2)^2)/(4*q^2*omegax)).*...
    besseli(0,(x.^2)*(1-q^4)/(4*q^2*omegax));

hold on
plot(x,y1,'black','LineStyle','.','LineWidth',1.5);

plot(x,y2,'black','LineStyle','-','LineWidth',1.5);
plot(x,y3,'black','LineStyle','-','LineWidth',1.5);
plot(x,y4,'black','LineStyle','-','LineWidth',1.5);
text(2,0.4,'$\Omega_X=$','Color','w','EdgeColor','black',...
    'interpreter','tex','FontSize',22,'Margin',5);
text(2,0.4,['$\Omega_X=$',num2str(omegax)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
ylabel('PDF, $p_{X}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
text(2,0.6,['$q=$',num2str(q1)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(2.5,0.6,['$q=$',num2str(q2)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(3,0.6,['$q=$',num2str(q3)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(3.5,0.6,['$q=$',num2str(q4)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
hold off


%----------------------------------------------------------------------
% Nakagami_qCDF.m
%
% This program computes the Nakagami-q cumulative distribution for
% different values of 'q' and 'Omega'. The equation for the Nakagami-q
% cumulative distribution is given in (2.68) [2].
% Used file:
%----------------------------------------------------------------------
% Nakagami_qCDF(q1,q2,q3,q4,omega)
%----------------------------------------------------------------------
% Explanation of input parameters:
% q1, q2, q3, q4: Different values of q characterizing the Nakagami-q
%                 cumulative distribution.
% omega: Input value for the Omega_x of the Nakagami-q  distribution
%
% The user will get Figure 2.6.(b) if he chose the parameters q1, q2,
% q3, q4 and omega as follows.
% |--------------------------------|-----------------------------------|
% | q1                             |  1/2                              |
% |--------------------------------|-----------------------------------|
% | q2                             |  1                                |
% |--------------------------------|-----------------------------------|
% | q3                             |  2                                |
% |--------------------------------|-----------------------------------|
% | q4                             |  3                                |
% |--------------------------------|-----------------------------------|
% | omega                          |  1                                |
% |--------------------------------|-----------------------------------|

function Nakagami_qCDF(q1,q2,q3,q4,omegax)
x=0.001:0.001:4;
q=q1;
y1=2/sqrt(2*pi*2).*exp(-x.^2/(4));
q=q2;
y2=x.*(1+q^2)/(q*omegax).*exp(-(x.^2)*((1+q^2)^2)/(4*q^2*omegax)).* ...
    besseli(0,(x.^2)*(1-q^4)/(4*q^2*omegax));

q=q3;
y3=x.*(1+q^2)/(q*omegax).*exp(-(x.^2)*((1+q^2)^2)/(4*q^2*omegax)).* ...
    besseli(0,(x.^2)*(1-q^4)/(4*q^2*omegax));

q=q4;
```

```
y4=x.*(1+q^2)/(q*omegax).*exp(-(x.^2)*((1+q^2)^2)/(4*q^2*omegax)).* ...
    besseli(0,(x.^2)*(1-q^4)/(4*q^2*omegax));
x=[0,x];y1=[0,y1];y2=[0,y2];y3=[0,y3];y4=[0,y4];
Y1=0;
for i=2:length(x)
    temp=trapz(x(1:i),y1(1:i));
    Y1=[Y1,temp];
end
Y2=0;
for i=2:length(x)
    temp=trapz(x(1:i),y2(1:i));
    Y2=[Y2,temp];
end
Y3=0;
for i=2:length(x)
    temp=trapz(x(1:i),y3(1:i));
    Y3=[Y3,temp];
end
Y4=0;
for i=2:length(x)
    temp=trapz(x(1:i),y4(1:i));
    Y4=[Y4,temp];
end

axes('FontSize',16);
plot(x,Y1,'black','LineStyle','-','LineWidth',1.5);
hold on
plot(x,Y2,'black','LineStyle','-','LineWidth',1.5);
plot(x,Y3,'black','LineStyle','-','LineWidth',1.5);
plot(x,Y4,'black','LineStyle','-','LineWidth',1.5);
text(2,0.4,'$\Omega_X=1$','Color','w','EdgeColor','black',...
    'interpreter','tex','FontSize',22,'Margin',5);
text(2,0.4,['$\Omega_X=$',num2str(omegax)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
ylabel('CDF, $F_{X}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
text(2,0.6,['$q=$',num2str(q1)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(2.5,0.6,['$q=$',num2str(q2)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(3,0.6,['$q=$',num2str(q3)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(3.5,0.6,['$q=$',num2str(q4)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
hold off


%-------------------------------------------------------------------------
% WeibullPDF.m
%
% This program computes the Weibull distribution for different values of
% beta_w and Omega. The equation for the Weibull distribution is
% given in (2.75) [2].
% Used file:
%-------------------------------------------------------------------------
```

```
% WeibullPDF(beta1,beta2,beta3,beta4,omega)
%------------------------------------------------------------------------
% Explanation of the input parameters:
% beta1, beta2, beta3, beta4: Different values of the 'beta_w'
%                             characterizing the Weibull distribution.
% omega: Input value for the Omega of the distribution
%
% The user will get Figure 2.7.(a) if he chose the parameters beta1,
% beta2, beta3, beta4 and omega as follows.
% |--------------------------------|----------------------------------|
% | beta1                          |  1                               |
% |--------------------------------|----------------------------------|
% | beta2                          |  2                               |
% |--------------------------------|----------------------------------|
% | beta3                          |  3                               |
% |--------------------------------|----------------------------------|
% | beta4                          |  4                               |
% |--------------------------------|----------------------------------|
% | omega                          |  1                               |
% |--------------------------------|----------------------------------|

function WeibullPDF(beta1,beta2,beta3,beta4,omega)
x=0:0.001:4;
beta=beta1;
y1=wblpdf(x,omega,beta);
beta=beta2;
y2=wblpdf(x,omega,beta);
beta=beta3;
y3=wblpdf(x,omega,beta);
beta=beta4;
y4=wblpdf(x,omega,beta);
axes('FontSize',16);
plot(x,y1,'black','LineStyle','-','LineWidth',1.5);
hold on
plot(x,y2,'black','LineStyle','-','LineWidth',1.5);
plot(x,y3,'black','LineStyle','-','LineWidth',1.5);
plot(x,y4,'black','LineStyle','-','LineWidth',1.5);
title('Weibull  Distribution')
text(3,1,'\Omega===','Color','w','EdgeColor','black',...
    'interpreter','tex','FontSize',22,'Margin',5);
text(3,1,['$\Omega=$',num2str(omega)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
ylabel('PDF, $p_{_{\mbox{w}}}(\mbox{x})$','HorizontalAlignment',...
    'right','interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
text(2,0.6,['$\beta_{w}=$',num2str(beta1)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(2.5,0.6,['$\beta_{w}=$',num2str(beta2)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(3,0.6,['$\beta_{w}=$',num2str(beta3)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(3.5,0.6,['$\beta_{w}=$',num2str(beta4)],...
    'interpreter','latex','FontSize',22,'Margin',100000);

hold off
```

```
%------------------------------------------------------------------------------
% WeibullCDF.m
%
% This program computes the Weibull cumulative distribution for different
% values of 'beta_w' and 'Omega'. The equation for the Weibull cumulative
% distribution is given in (2.76) [2].
% Used file:
%------------------------------------------------------------------------------
% WeibullCDF(beta1,beta2,beta3,beta4,omega)
%------------------------------------------------------------------------------
% Explanation of the input parameter:
% beta1, beta2, beta3, beta4: Different values of the beta_w characterizing
%                             the Weibull distribution.
% omega: Input value for 'Omega' of the Weibull distribution
%
% The user will get Figure 2.7. (b) if he chose the parameters beta1,
% beta2, beta3, beta4 and omega as follows.
% |---------------------------------|-----------------------------------|
% | beta1                           |  1                                |
% |---------------------------------|-----------------------------------|
% | beta2                           |  2                                |
% |---------------------------------|-----------------------------------|
% | beta3                           |  3                                |
% |---------------------------------|-----------------------------------|
% | beta4                           |  4                                |
% |---------------------------------|-----------------------------------|
% | omega                           |  1                                |
% |---------------------------------|-----------------------------------|

function WeibullCDF(beta1,beta2,beta3,beta4,omega)
x=0:0.001:5;
beta=beta1;
y1=wblcdf(x,omega,beta);
beta=beta2;
y2=wblcdf(x,omega,beta);
beta=beta3;
y3=wblcdf(x,omega,beta);
beta=beta4;
y4=wblcdf(x,omega,beta);
axes('FontSize',16);
plot(x,y1,'black','LineStyle','-','LineWidth',1.5);
hold on
plot(x,y2,'black','LineStyle','-','LineWidth',1.5);
plot(x,y3,'black','LineStyle','-','LineWidth',1.5);
plot(x,y4,'black','LineStyle','-','LineWidth',1.5);
title('Weibull Cumulative Distribution')
 %title('{\itAe}^{-\alpha\itt}sin\beta{\itt} \alpha<<\beta')
text(2,0.3,'\beta===1','Color','w','EdgeColor','black',...
    'interpreter','tex','FontSize',22,'Margin',5);
text(2,0.3,['$\Omega=$',num2str(omega)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
ylabel('CDF, $F_{_{\mbox{w}}}(\mbox{x})$','HorizontalAlignment',...
    'right','interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
text(2,0.6,['$\beta_{w}=$',num2str(beta1)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(2.5,0.6,['$\beta_{w}=$',num2str(beta2)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(3,0.6,['$\beta_{w}=$',num2str(beta3)],...
    'interpreter','latex','FontSize',22,'Margin',100000);
text(3.5,0.6,['$\beta_{w}=$',num2str(beta4)],...
    'interpreter','latex','FontSize',22,'Margin',100000);

hold off
```

MATLAB –PROGRAMS for Study of Elementary Properties of Rayleigh and Rice Channel

```
%--------------------------------------------------------------------------
% JakePSD.m
%
% This program computes the Jakes power spectral density according to given
% values of maximum Doppler frequency f_max and variance sigma_0^2'.
% Equation of the Jakes Power spectral density
% is given in (3.23) [2].
% A plot of the Gaussian ACF is shown in Figure 3.3.(a)
%
% used m-file:
%--------------------------------------------------------------------------
% JakePSD(f_max,sigma02)
%--------------------------------------------------------------------------
% Explanation of the input parameters:
% f_max: maximum Doppler frequency
% sigma02: Mean power sigma_0^2 (variance of the complex Gaussian process
%          mu(t))
%--------------------------------------------------------------------------

function JakePSD(f_max,sigma02)

f=-110:0.5:110;
if abs(f) >= 91;
     s=0;
 else
     s=(sigma02/(pi*f_max))*(1./sqrt(1-(f/f_max).^2));
 end
figure;
plot(f,s,'k');
set(0,'DefaultAxesFontSize',16);
hold on;
ylabel('PSD,  $ S_{\mu{\i}\mu{\i}}\mbox(f)$','HorizontalAlignment',...
    'center','interpreter','latex','FontSize',24)
xlabel('$  Frequency,f(Hz)$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',24)
title ('Jake power spectral density ','interpreter','latex','FontSize',24)

%--------------------------------------------------------------------------
% JakeACF.m
%
% This program computes corresponding autocorrelation function of the Jakes
% power spectral density according to given values of f_max and
% variance sigma_0^2. The equation for corresponding autocorrelation
% function of the Jakes Power spectral density is given in (3.23) [2].
% A plot of the Gaussian ACF is shown in Figure 3.2. (b)
%
% used m-file:
%-----------------------------------------------------------------------------
% JakeACF(f_max,sigma02)
%-----------------------------------------------------------------------------
% Explanation of the input parameters:
% f_max: Maximum Doppler frequency
% sigma02: Mean power sigma_0^2 (variance of the complex Gaussian process
%          mu(t))
%-----------------------------------------------------------------------------

function JakeACF(f_max,sigma02)

z1=(0:0.0001:0.05);
acf=sigma02.*besselj(0,z1*f_max*2*pi);

figure;
grid on;
hold on;
ylabel('ACF,  $ r_{\mu{\i}\mu{\i}}\mbox({\tau})$',...
    'HorizontalAlignment','right','interpreter','latex','FontSize',24)
xlabel('Time difference,$\tau$','HorizontalAlignment','center',...
    'interpreter','latex','FontSize',24)
title ('Corresponding ACF of the JPSD ','interpreter','latex',...
    'FontSize',20)
plot(z1,acf,'k');
set(0,'DefaultAxesFontSize',16);
xlim([0 0.05]);

%-----------------------------------------------------------------------------
% GaussianPSD.m
%
% This program computes  the Gaussian power spectral density according to
% given % input values of maximum Doppler frequency f_max and variance
% sigma_0^2. The equation of the Jakes power spectral density  is given
% in (3.26) [2]. A plot of the Gaussian PSD is shown in Figure 3.3.(a)
%
% used m-file:
%-----------------------------------------------------------------------------
% GaussianPSD(f_max,sigma02)
%-----------------------------------------------------------------------------
% Explanation of the input parameters:
% f_max: Maximum Doppler frequency
% sigma02: Mean power sigma_0^2 (variance of the complex Gaussian process
%          mu(t))
%-----------------------------------------------------------------------------

function GaussianPSD(sigma02,f_max)
l=sqrt(log (2));
f_c=l*f_max;

f=-210:1:210;
s_f=sigma02.*l./(f_c*sqrt(pi)).*exp(-(l.*(f/f_c).^2));
```

```
figure;
hold on;
set(0,'DefaultAxesFontSize',18);
plot(f,s_f,'black','LineStyle','.','LineWidth',1.5);
set(0,'DefaultAxesFontSize',18);
title ('Gaussian PSD ','interpreter','latex',...
    'FontSize',20)
ylabel('PSD,  $ S_{\mu{\i}\mu{\i}}\mbox(f)$','HorizontalAlignment',...
    'center','interpreter','latex','FontSize',24)
xlabel('$  Frequency,f(Hz)$','HorizontalAlignment','center',...
    'interpreter','latex','FontSize',24)

%--------------------------------------------------------------------------
% GaussianACF.m
%
% This program computes corresponding autocorrelation function of the
% Gaussian power spectral density according to given values of maximum
% Doppler frequency f_max and variance sigma_0^2. The equation for
% corresponding autocorrelation function of the Jakes power spectral
% density  is given in (3.27) [2].
% A plot of the Gaussian ACF is shown in Figure 3.3.(b)
%
% used m-file:
%--------------------------------------------------------------------------
% GaussianACF(f_max,sigma02)
%--------------------------------------------------------------------------
% Explanation of the input parameters:
% f_max: Maximum Doppler frequency
% sigma02: Mean power sigma_0^2 (variance of the complex Gaussian process
%          mu(t))
%--------------------------------------------------------------------------

function GaussianACF(f_max,sigma02)
l=sqrt(log (2));
f_c=l*f_max;

tau=0:0.0005:0.05;
r_tau=sigma02.*exp(-(pi*f_c*tau./l).^2);
figure;
hold on;
set(0,'DefaultAxesFontSize',18);
plot(tau,r_tau,'black','LineStyle','-','LineWidth',1.5);
title ('Corresponding ACF of the GPSD ','interpreter','latex',...
    'FontSize',20)
ylabel('ACF,  $ r_{\mu{\i}\mu{\i}}\mbox({\tau})$',...
    'HorizontalAlignment','center','interpreter','latex','FontSize',24)
xlabel('Time difference, $\tau$','HorizontalAlignment','center',...
    'interpreter','latex','FontSize',24)

%--------------------------------------------------------------------------
% RicePDF_SE.m
%
% This program computes the probability density function of the squared
% envelop of Rice process.  The equation is  given in (3.59)[2].
% Used m-file:
%--------------------------------------------------------------------------
% RicePDF_SE(rho1,rho2,rho3,sigma02)
% Explanation of the input parameters:
%
% rho1,rho2,rho_03:  Different values of rho characterzing the distribution.
% sigma02: Mean power sigma_0^2 (variance of the complex Gaussian process
%          mu(t))
%
% The user will get Figure 3.7.(a) if he chose the parameter ro1, ro2, ro3,
% ro4 and sigma02 as follows.
% |--------------------------------|-------------------------------------|
% | rho1                           |  0                                  |
% |--------------------------------|-------------------------------------|
% | rho2                           |  1/2                                |
% |--------------------------------|-------------------------------------|
% | rho3                           |  1                                  |
% |--------------------------------|-------------------------------------|
% | sigma02                        |  1                                  |
% |--------------------------------|-------------------------------------|
function RicePDF_SE(rho1,rho2,rho3,sigma02)
x=0:0.001:10;
X1=(exp(-(x+rho1^2)./(2*sigma02))).*  besseli(0,sqrt(x).*rho1./...
    sigma02)./(2*sigma02);
X2=(exp(-(x+rho2^2)./(2*sigma02))).*  besseli(0,sqrt(x).*rho2./...
    sigma02)./(2*sigma02);
X3=(exp(-(x+rho3^2)./(2*sigma02))).*  besseli(0,sqrt(x).*rho3./...
    sigma02)./(2*sigma02);
axes('FontSize',18);
plot(x,X1,'black','LineStyle','.','LineWidth',1.5);
hold on
plot(x,X2,'black','LineStyle','-','LineWidth',1.5);
plot(x,X3,'black','LineStyle','-.','LineWidth',1.5);
title ('The PDF of the squared envelop of Rice processes ',...
    'interpreter','latex','FontSize',18)
text(4.5,0.35,'\sigma_0^2==','Color','w','EdgeColor','black',...
     'interpreter','tex','FontSize',16,'Margin',5);
 text(4.5,0.35,['$\sigma_0^2=$',num2str(sigma02)],...
     'interpreter','latex','FontSize',16,'Margin',100000);
ylabel('PDF, $p_{\xi^2}(\mbox{x})$','HorizontalAlignment','center',...
    'interpreter','latex','FontSize',24)
xlabel('x','HorizontalAlignment','center',...
    'interpreter','latex','FontSize',24)
% text(2,0.6,'$\rho=1/2$',...
%     'interpreter','latex','FontSize',16,'Margin',100000);
% text(3,0.6,'$\rho=1$',...
%     'interpreter','latex','FontSize',16,'Margin',100000);
% text(4,0.6,'$\rho=2$',...
%     'interpreter','latex','FontSize',16,'Margin',100000);
h=legend(['$\rho=$',num2str(rho1)],['$\rho=$',num2str(rho2)],['$\rho=$',...
    num2str(rho3)]);
set(h,'FontSize',22,...
  'Interpreter','latex');
hold off


%--------------------------------------------------------------------------
% RayleighPDF_SE.m
%
% This program computes the probability density function of the squared
% envelop of the Rayleigh process given in the equation (3.62).[2]
% Used m-file:
%---------------------------------------------------------------------------
% Explanation of the input parameters:
%
% sigma_01,sigma_02,sigma_03: Different values of mean power sigma_0^2
%                             (variance of the complex Gaussian process
%                              mu(t)).
%  the distribution
% The user will get Figure 3.7.(b) if he chose the parameters sigma21,
% sigma22,sigma23 and m as follows.
% |--------------------------------|------------------------------------|
% | sigma01                        |  1/2                               |
% |--------------------------------|------------------------------------|
% | sigma02                        |   1                                |
% |--------------------------------|------------------------------------|
% | sigma03                        |   2                                |
% |--------------------------------|------------------------------------|

function RayleighPDF_SE(sigma01,sigma02,sigma03)
x=0:0.001:10;
sigma=sigma01;
X1=(exp(-(x./(2*sigma)))).(2*sigma);
sigma=sigma02;
X2=(exp(-(x./(2*sigma)))).(2*sigma);
sigma=sigma03;
X3=(exp(-(x./(2*sigma)))).(2*sigma);
set(0,'DefaultAxesFontSize',18);
plot(x,X1,'black','LineStyle','-','LineWidth',1.5);
hold on
set(0,'DefaultAxesFontSize',18);
plot(x,X2,'black','LineStyle','.','LineWidth',1.5);
plot(x,X3,'black','LineStyle','-.','LineWidth',1.5);

title ('The PDF of the squared envelop of Rayleigh processes ',...
    'interpreter','latex','FontSize',18)
ylabel('PDF, $p_{\xi^2}(\mbox{x})$','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',18)
xlabel('x','HorizontalAlignment','right',...
    'interpreter','latex','FontSize',18)

h=legend(['$\sigma_0^2=$',num2str(sigma01)],['$\sigma_0^2=$',...
    num2str(sigma02)],['$\sigma_0^2=$',num2str(sigma03)]);
set(h,'FontSize',16,...
  'Interpreter','latex');
hold off
```