Modeling and Simulation of Random Processes and Fields in Civil Engineering and Engineering Mechanics

Brett A. Benowitz

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY
2013

ABSTRACT

Modeling and Simulation of Random Processes and Fields in Civil Engineering and Engineering Mechanics

Brett A. Benowitz

This thesis covers several topics within computational modeling and simulation of problems arising in Civil Engineering and Applied Mechanics. There are two distinct parts. Part 1 covers work in modeling and analyzing heterogeneous materials using the eXtended Finite Element Method (XFEM) with arbitrarily shaped inclusions. A novel enrichment function, which can model arbitrarily shaped inclusions within the framework of XFEM, is proposed. The internal boundary of an arbitrarily shaped inclusion is first discretized, and a numerical enrichment function is constructed "on the fly" using spline interpolation. This thesis considers a piecewise cubic spline, which is constructed from seven localized discrete boundary points. The enrichment function is then determined by solving numerically a nonlinear equation, which determines the distance from any point to the spline curve. Parametric convergence studies are carried out to show the accuracy of this approach, compared to pointwise and linear segmentation of points, for the construction of the enrichment function in the case of simple inclusions and arbitrarily shaped inclusions in linear elasticity. Moreover, the viability of this approach is illustrated on a Neo-Hookean hyperelastic material with a hole undergoing large deformation. In this case, the enrichment is able to adapt to the deformation and effectively capture the correct response without remeshing.

Part 2 then moves on to research work in simulation of random processes and fields. Novel algorithms for simulating random processes and fields such as earthquakes, wind fields, and properties of functionally graded materials are discussed. Specifically, a methodology is presented to determine the Evolutionary Spectrum (ES) for non-stationary processes from a prescribed or measured non-stationary Auto-Correlation Function (ACF). Previously, the existence of such an inversion was unknown, let alone possible to compute or estimate. The classic integral expression suggested by Priestley, providing the ACF from the ES, is not invertible in a unique way so that the ES could be determined from a given ACF. However, the benefits of an efficient inversion from ACF to ES are vast. Consider for example various problems involving simulation of non-stationary processes or non-homogeneous fields, including non-stationary seismic ground motions as well as non-homogeneous material properties such as those of functionally graded materials. In such cases, it is sometimes more convenient to estimate the ACF from measured data, rather than the ES. However, efficient simulation depends on knowing the ES. Even more important, simulation of non-Gaussian and non-stationary processes depends on this inversion, when following a spectral representation based approach. This work first examines the existence and uniqueness of such an inversion from the ACF to the ES under a set of special conditions and assumptions (since such an inversion is clearly not unique in the most general form). It then moves on to efficient methodologies of computing the inverse, including some established optimization techniques, as well as proposing a novel methodology. Its application within the framework of translation models for simulation of non-Gaussian, non-stationary processes is developed and discussed. Numerical examples are provided demonstrating the capabilities of the methodology.

Additionally in Part 2, a methodology is presented for efficient and accurate simulation of wind velocities along long span structures at a virtually infinite number of points. Currently, the standard approach is to model wind velocities as a multivariate stochastic process, characterized by a Cross-Spectral Density Matrix (CSDM). In other words, the wind velocities are modeled as discrete components of a vector process. To simulate sample functions of the vector process, the Spectral Representation Method (SRM) is used. The SRM involves a Cholesky decomposition of the CSDM. However, it is a well known issue that as the length of the structure, and consequently the size of the vector process, increases, this Cholesky decomposition breaks down (from the numerical point of view). To avoid this issue, current research efforts in the literature center around approximate techniques to simplify the decomposition. Alternatively, this thesis proposes the use of the frequency-wavenumber (F-K) spectrum to model the wind velocities as a stochastic "wave," continuous in both space and time. This allows the wind velocities to be modeled at a virtually infinite number of points along the length of the structure. In this work, the relationship between the CSDM and the F-K spectrum is first examined, as well as simulation techniques for both. The F-K spectrum for wind velocities is then derived. Numerical examples are then carried out demonstrating that the simulated wave samples exhibit the desired spectral and coherence characteristics. The efficiency of this method, specifically through the use of the Fast Fourier Transform, is demonstrated.

Table of Contents

List of Figures **v**

List of Tables **viii**

1 Introduction & outline **1**

1.1 Introduction . . . 1

1.2 Outline . . . 2

I Modeling heterogeneous fields **4**

2 XFEM for modeling inclusions **5**

2.1 Motivation & literature review . . . 5

2.2 Assumptions and governing equations . . . 7

2.2.1 Governing equations . . . 7

2.2.2 Strong and weak form . . . 9

2.3 Formulation of the eXtended Finite Element Method . . . 9

2.4 Enrichment functions . . . 11

2.4.1 Definitions . . . 11

2.4.2 Formulations for simple geometries . . . 11

3 Spline based enrichment formulation for arbitrary shapes **14**

3.1 Cubic spline interpolation . . . 14

3.1.1 Definitions and formulation . . . 14

3.1.2 Parametric cubic splines . . . 15

3.1.3 Localized splines . . . 16

3.2 Calculation of signed distance function . . . 19

3.2.1 Determining the magnitude of the signed distance function . . . 19

3.2.2 Determining the sign of the signed distance function 20
3.2.3 Level set function algorithm 20
3.3 Enrichment function differentiation 24
3.4 Numerical integration 24
3.5 Convergence analysis 25
3.5.1 Level set function convergence 25
3.5.2 Differentiation convergence 26
3.5.3 Classical XFEM comparison for circular inclusion 27
3.5.4 FEM comparison for arbitrary inclusion 29
3.6 Conclusions 30

4 Non-linear XFEM formulation for finite deformations 31
4.1 Definitions and formulation 31
4.2 Non-Linear XFEM example case study 34
4.3 Conclusions 40

II Simulating random procresses & fields 41

5 Simulation of stochastic processes & fields 42
5.1 Introduction 42
5.2 Stationary, univariate, Gaussian stochastic processes 43
5.3 Non-stationary, univariate, Gaussian stochastic processes 45
5.3.1 Non-stationary processes 45
5.3.2 Simulation of amplitude modulated non-stationary processes 47
5.3.3 Simulation of amplitude and frequency modulated non-stationary processes 48
5.4 Non-Guassian stochastic processes 48
5.4.1 Stationary non-Gaussian stochastic processes 48
5.4.2 Non-stationary non-Gaussian stochastic processes 49
5.5 Multi-variate stochastic processes 50
5.6 Multi-dimensional stochastic processes 52
5.6.1 Multi-dimensional random fields 52
5.6.2 Stochastic waves 53

6 Determining Evolutionary Spectra from non-stationary Autocorrelation Functions 55
6.1 Introduction & motivation 55

6.2 Formulation of discrete optimization problem 58
6.3 "Brute Force" Method (BFM) 59
6.3.1 Description of algorithm 59
6.3.2 BFM results 61
6.4 "Multi-Grid" Method (MGM) 66
6.4.1 Description of algorithm 66
6.4.2 MGM results 67
6.4.3 Efficiency comparison 73
6.5 Preprocessing 76
6.5.1 Definition of "Pseudo-Spectrum" 76
6.5.2 Numerical examples 76

7 Simulation of non-stationary and non-Gaussian stochastic processes 84
7.1 Introduction & motivation 84
7.2 Review of methodologies for simulating stationary non-Gaussian processes 86
7.3 Proposed methodology for simulating non-stationary and non-Gaussian processes 89
7.4 Numerical examples 92
7.4.1 Log-normal distribution 92
7.4.2 Uniform distribution 96

8 Simulation of wind velocities along long-span structures 101
8.1 Introduction & motivation 101
8.2 Modeling wind as a stochastic wave 103
8.2.1 Derivation of frequency-wavenumber (F-K) spectrum 103
8.2.2 Verification of F-K spectrum 104
8.3 Simulation of wind velocities as a wave 105
8.4 Numerical example 107
8.4.1 Problem definition 107
8.4.2 Results 108

9 Conclusions 112
9.1 Main contributions 112
9.1.1 Part I: Modeling heterogeneous fields 112
9.1.2 Part II: Simulating random procresses & fields 113
9.2 Future work 113

9.2.1 Part I: Modeling heterogeneous fields 113
9.2.2 Part II: Simulating random procresses & fields 114

Bibliography **115**

A Closest point on spline curve derivation **124**

B Quadrature for C^0 continuous functions **126**
B.1 Introduction 126
B.2 Analytical reference solution 127
B.3 Quadrature convergence 128

C MATLAB codes for the Spectral Representation Method, and FFT implementations **129**
C.1 Introduction 129
C.2 Stationary processes 129
C.2.1 MATLAB codes 129
C.2.2 Efficiency comparison 131
C.3 Non-stationary processes 131
C.4 Stochastic waves 132
C.4.1 MATLAB codes 132
C.4.2 Efficiency comparison 134

List of Figures

2.1 FEM and XFEM meshes for heterogeneous materials 6

2.2 Level set functions 13

3.1 Subset orientation and scalar spline formulations 16

3.2 Stability of scalar splines 17

3.3 Stability of vectorized parametric splines 17

3.4 Localized splines 18

3.5 Point-in-polygon methods 21

3.6 Determining the sign of the level set function 22

3.7 Level set generation 23

3.8 Integration techniques for enriched elements 25

3.9 Problem geometry 26

3.10 Convergence of level set function 26

3.11 Convergence of enrichment derivatives 27

3.12 Circle problem geometry and mesh 28

3.13 Comparison to ‘classical’ XFEM formulation 28

3.14 Mesh comparison to ‘classical’ XFEM formulation 29

3.15 XFEM-FEM comparison for arbitrary inclusion 30

4.1 Plate with hole geometry and loading conditions 36

4.2 von Mises stresses for the plate of Figure 4.1(a) 37

4.3 Plate with hole force-displacement 38

4.4 Plate with hole in bending and pinching: Von Mises stresses on deformed configuration . . . 39

5.1 Stationary vs. non-stationary processes 47

6.1 Loma-Prieta acceleration time history 56

6.2 Sample Functionally Graded Material (FGM) 56
6.3 Brute Force Method (BFM) algorithm 60
6.4 BFM Iterations 61
6.5 Clough-Penzien (CP) spectrum with amplitude and frequency modulation 62
6.6 Jennings, Housner, and Tsai modulating function 62
6.7 BFM convergence 65
6.8 BFM generated samples 66
6.9 Multi-Grid Method (MGM) algorithm 68
6.10 MGM iterations 69
6.11 MGM convergence 70
6.12 MGM generated samples 71
6.13 Optimization comparison 71
6.14 MATLAB `peaks` function 72
6.15 MGM MATLAB `peaks` example results 74
6.16 MGM MATLAB `peaks` example errors 75
6.17 `peaks` MGM generated samples 75
6.18 Pseudo-spectrum results for Kanai-Tajimi example 77
6.19 Pseudo-spectrum generated samples 78
6.20 MGM with pre-processing: Kanai-Tajimi 79
6.21 Pseudo-spectrum results for Clough-Penzien example 79
6.22 Pseudo-spectrum generated samples 80
6.23 MGM with pre-processing: Clough-Penzien 80
6.24 Pseudo-spectrum results for linear chirp example 82
6.25 MGM with pre-processing: linear chirp 82
6.26 MGM with pseudo-spectrum generated samples 83

7.1 Stationary and non-Gaussian algorithm 88
7.2 Proposed non-stationary and non-Gaussian algorithm 90
7.3 Log-normal distribution 93
7.4 Log-normal Kanai-Tajimi process: evolutionary spectra 94
7.5 Log-normal Kanai-Tajimi convergence 95
7.6 Log-normal correlation distortion 96
7.7 Uniform distribution 97
7.8 Uniform Kanai-Tajimi process: evolutionary spectra 98
7.9 Uniform Kanai-Tajimi convergence 99

7.10 Uniform correlation distortion 100

8.1 The Tacoma Narrows bridge 102
8.2 Simulation of wind velocities as a wave 106
8.3 Kaimal vs. estimate spectra 109
8.4 Davenport vs. estimated coherence 110
8.5 Generated sample realization of wind velocities modeled as a stochastic wave 111

B.1 Quadrature test problem geometry 127
B.2 Quadrature convergence 128

C.1 Efficiency comparison 131
C.2 Efficiency comparison 134

List of Tables

6.1 BFM and MGM efficiency comparison 73

Acknowledgments

The work that this dissertation represents, both the studies within, and the career it culminates, would not have been possible without many people. Firstly, I would like to thank my girlfriend, Sandy, for her steadfast support during this process, for welcome distractions when needed, and for being a role model after finishing law school and passing the bar. Being able to come home to you made this all a lot easier. Plus, I think the entire department is thankful for your baked goods. I especially need to thank my parents, for their support in every way imaginable, and for instilling a desire for, and value of, education from a very young age. My father Marc, as much as I make fun of his dorkiness and Brooklyn accent, has always been, and will always be, my biggest role model. Since I was three years old I knew I would get a Ph.D. from this very department. My mother Nancy, as much as she makes fun of *my* dorkiness, I know that she has always been proud and supporting of me. I also would like to thank my sister Jackie, and of course Penny, for always being there when I needed family.

I, of course, also need to thank my advisors here at Columbia. Both Prof. George Deodatis and Prof. Haim Waisman always had open doors when I needed guidance and advice - be it technical, professional, or personal. I am eternally grateful for their assistance, support, and guidance. Prof. Raimondo Betti, as my undergraduate advisor, played a pivotal role in helping to decide if, where, and when to pursue my doctorate, and was always there for help along the way. I would also like to thank Kington Chan for his technical support along the way, and always going above and beyond to help ensure we had the best computational resources available. I am indebted to the entire faculty and staff of the Department of Civil Engineering and Engineering Mechanics, which has been my home for the past seven years, and for the department's support through teaching and research fellowships. I would also like to thank Prof. Patricia Culligan, and the National Science Foundation, for the life changing opportunities that were made available to me through working with the NSF IGERT, and the Urban Ecology Studio. The ability to travel all over the world, and work on interesting problems, was a unique and important element in my career.

Last, but certainly not least, I need to thank my friends and colleagues here at Columbia. Those that were ahead of me, for their endless patience and support in helping me, and those that were along side me from beginning until end. You guys made these years among the best of my life. Thanks for the lunch breaks, the laughs, the trouble shooting, and the endless fun. I will inevitably forget some names, and for that I apologize, but I would like the specifically thank: Adrian, Arturo, Badri, Ben, Colin, Dan, Kirubel, Luc, Luciana, Mady, Mahesh, Manolis, Matt, Mike Lackey, Mike Shields, Pablo, Packy, Po-Hua, Rafi, Rishee, Suparno, Tyler, the rest of the CEEM students, and all of the IGERTs.

Thank you all.

To my parents, for teaching me to always ask "why?"

Chapter 1

Introduction & outline

1.1 Introduction

In this thesis, some contributions to the fields of modeling and simulating physical phenomena, with particular applications within civil engineering and applied mechanics, are presented. Although, in general, the terms "modeling" and "simulating" are often used interchangeably, in this work a distinction is made. The term "modeling" will be used to denote a deterministic computational model of physical phenomena. On the other hand, the term "simulation" will refer to the generation of sample functions of stochastic processes and fields. These two areas divide the thesis into its separate parts. Part I discusses contributions in computational modeling of solid mechanics problems involving weak discontinuities (i.e. heterogeneous materials), and Part II discusses the simulation of stochastic processes and fields.

More specifically, Part I introduces a novel formulation for modeling heterogeneous materials with arbitrarily shaped inclusions or flaws within the eXtended Finite Element Method (XFEM). This method is then applied to non-linear finite deformation problems in hyperelasticity. The method developed has applications in new material design/optimization, uncertainty quantification, and flaw detection. Part II, then moves on to the simulation of stochastic processes and fields. The first contribution discussed in this part is in the simulation of non-stationary and non-Gaussian stochastic processes and fields. Then, a novel formulation for simulating wind velocity fields along long-span structures is presented. A more detailed overview of both of these parts is given in the outline below.

Both parts of this thesis, while rooted in the detailed theory and/or numerics, are concerned with practical necessities in the field. Strong computer models, coupled with solid simulation, are necessary for strong physical understanding of our engineered systems. Be they aircraft, bridges, skyscrapers, computer hardware, mobile smart phones, or anything in between – computer models are invariably used in the physical design.

Usually these analyses are deterministic, though increasingly probabilistic consideration is given to the design of these systems. As an example, consider the critical infrastructure systems of a nation/society. The design of these systems will generally be governed by extreme loads – storm winds, earthquakes, or other natural or man-made hazards/catastrophes. The only thing that is known with any certainty about these loads *a priori* is that they are highly uncertain. When consideration is given to climate change as well, the uncertainty only grows. This is where the importance of probabilistic modeling in design lies. For truly resilient, and by result, sustainable infrastructure, these highly uncertain conditions must be modeled and simulated in an efficient, accurate, and robust manner during the design process.

1.2 Outline

This dissertation is organized into two distinct parts. Part I revolves around computational modeling of heterogeneous fields. Chapter 2 introduces XFEM, and presents a brief review of XFEM for modeling heterogeneous materials. Then, Chapter 3 develops the proposed formulation for arbitrary inclusions, as well as reviews spline interpolation, which is the basis for the proposed method. Chapter 3 also examines the convergence of the proposed method, in all stages of implementation - from the level set function to the XFEM implementation for an arbitrary inclusion. Finally, Chapter 4 develops and examines an application of this methodology to non-linear finite deformation problems.

Part II then moves on to discuss simulation of random processes and fields. First, in Chapter 5, existing simulation techniques are reviewed. In particular, the Spectral Representation Method (SRM) is reviewed for the simulation of various classes of random processes/fields, including stationary, non-stationary, Gaussian, non-Gaussian, multi-variate, and multi-dimensional processes/fields (and combinations thereof). Chapter 6 then moves on to non-stationarity, and in particular the inversion of the Evolutionary Spectrum (ES) from a prescribed or measured Auto-Correlation Function (ACF). The existence and uniqueness of this solution is examined, and novel, efficient methodologies are presented for the inversion. This computation is important for the simulation of non-stationary processes, and critical for the simulation of non-stationary and non-Gaussian processes within the framework of the SRM. Following the developments presented in Chapter 6, Chapter 7 then develops a novel algorithm for simulating non-stationary and non-Gaussian stochastic processes. In particular, the underlying Gaussian ES for a prescribed non-Gaussian process is found, for use in a translation based simulation with the SRM. Finally, Chapter 8 proposes an innovative method for simulating wind velocities along long-span bridges. In contrast to the current methodology of modeling these velocities as discrete components of a stochastic vector process, they are modeled as a stochastic wave, continuous in time and space. This method is more accurate, robust, and efficient than previous.

Chapter 9 concludes the thesis with a summary, and presents the major contributions of the work.

Following this conclusion is the bibliography and appendices. In Appendix A, the derivation of the closest point on a spline curve is presented. Appendix B presents a convergence study for numerical quadrature on weakly discontinuous functions. Finally, some sample MATLAB codes for simulation of stochastic processes are provided in Appendix C, as is a brief demonstration of the efficiency gains due to the use of the Fast Fourier Transform (FFT).

Part I

Modeling heterogeneous fields

Chapter 2

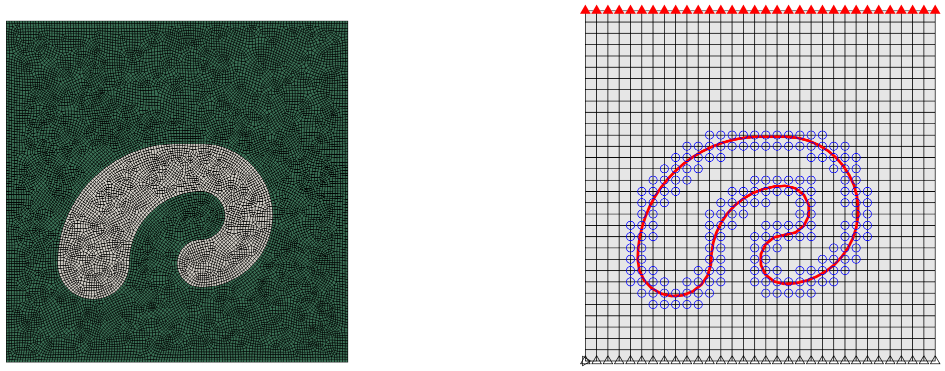
XFEM for modeling inclusions

2.1 Motivation & literature review

Multiphase materials are among, if not, the most prevalent materials on earth - from concrete to nano-fiber reinforced polymers. Accurate modeling and simulation of these complex macro- and micro-structures is non-trivial and computationally intensive. The eXtended Finite Element Method (XFEM) has proven to be efficient in the modeling of multiphase materials. However, it has mainly been developed for inclusions with somewhat simplified geometries — e.g. circular [1] and elliptical [2] internal boundaries. Sometimes the inhomogeneities are, or can be approximated as, simple shapes such as circles or ellipses. However, there are many materials (e.g. concrete) in which the inhomogeneities are arbitrarily shaped. Thus, it is of great interest to model these materials efficiently and accurately for use in analysis and design. This thesis discusses the numerical modeling of arbitrarily shaped inclusions within the framework of the XFEM.

XFEM was first introduced by Belytschko and co-workers [3, 4] for modeling crack growth. It possesses a major advantage over the conventional Finite Element Method (FEM) in that it is mesh independent. For crack growth, this is advantageous because there is no need for remeshing as the crack grows. For inhomogeneous materials or voids this is also advantageous because there is no need to generate complicated meshes that conform exactly to internal boundaries, as shown in Figure 2.1. XFEM was applied to such problems of holes and inclusions by Sukumar et. al. [1]. Corrected enrichments were proposed by Moës et. al. [5], and Fries and Belytschsko [6, 7], as well.

The methods proposed in [1] are easily applied to simple shapes, such as circles [1] and ellipses [2], where an analytical expression for the necessary level set function (as will be described shortly) is readily available. This work extends these methods to arbitrary shapes using a numerical algorithm, as opposed to the analytical expressions, which had been used previously for simple shapes. Through a discrete description of internal



(a) FEM mesh (b) XFEM mesh

Figure 2.1: FEM and XFEM meshes for heterogeneous materials. The FEM mesh on the left is complicated as it needs to conform to internal geometries. The XFEM mesh on the right, however, does not need to conform to internal geometries, and is regular. The highlighted nodes have added enriched degrees of freedom to account for this mesh independence.

boundaries, a numerical enrichment function is developed, which can handle these arbitrary internal geometries. Numerical enrichment functions have been used previously: Waisman and Belytschko [8] proposed a numerical updating scheme for the enrichment function, and Menk and Bordas [9] proposed a numerical formulation for bi-material anisotropic fracture. Liu and Taciroglu [10, 11] used a spline based enrichment function in an enriched Reproducing Kernal Particle Method for multi-phase piezoelectric materials. Other noteworthy uses of numerical, or non-analytical, enrichment functions in the literature include: Chahine et. al. [12] used the addition of an adaptive discretization for problems consisting of unknown crack tip displacements; Aquino et. al. [13] proposed a proper orthogonal decomposition for developing enrichment functions from observed or simulated data; and Abbas et. al. [14] employed a set of regularized Heaviside functions as enrichments for problems with high gradients. However, this study is the first such application to problems of inclusions and voids, including complex geometries, in XFEM. An alternative method is presented in [15], in which a higher-order XFEM mesh is used to represent curved interfaces to a higher resolution.

One major application of the methods described in this thesis is in nonlinear finite deformation problems. Consider, for example, a piece of rubber with a circular hole in the center, being stretched. As the piece of rubber is deformed, the hole will deform as well. Thus, an algorithm that can handle the evolving shape of the hole or inclusion is advantageous. Problems of interface tracking have been the focus of the Level Set Method (LSM) and Fast Marching Method (FMM) [16–18]. The LSM and FMM were first introduced by Osher and Sethian [16, 17] to model the propagation of interface boundaries. It has been coupled with XFEM (and other mesh independent formulations) for modeling crack propagation [19–21] and material interfaces [1, 22–24]. In the LSM and FMM, the interface is modeled by the zero contour of a level set function that is of one higher dimension. The interface is propagated forward through advection. However, for nonlinear finite deformation problems, as considered in this work, use of the LSM is cumbersome. Instead, in this work, the interface tracking will be handled by "re-initialization" at each incremental step of the analysis, which will be discussed further in Chapter 4.

Another application of these methods would be in optimization problems dealing with inclusion geometry. For example, detection and quantification of flaws in structures using XFEM and genetic algorithms (GA) was shown in [25–27]. However, these studies only used simple, radially symmetric flaw shapes. The proposed methods would not only work for arbitrary shapes accurately, but also, through the use of a few discrete points and spline interpolation to describe internal boundaries, the proposed methodology can reduce the number of parameters necessary to describe these shapes, which is of the highest importance in optimization. Another application is in uncertainty quantification — e.g. in Monte-Carlo or collocation methods where repeated solutions of different geometries are necessary. Therefore, the need to remesh at each realization would be avoided through the use of XFEM, as shown in [2]. To summarize, a spline-based enrichment function is capable of describing arbitrary shaped inclusions and holes and in addition adapt to new and evolving shapes.

Part I of this thesis is organized as follows: In Chapter 2, the XFEM formulation for inclusions/holes in a domain is briefly reviewed. Then, Chapter 3 develops the proposed formulation for arbitrary inclusions, as well as reviews spline interpolations. Chapter 3 also examines the convergence of the proposed method, in all stages of implementation - from the level set function to the XFEM implementation for an arbitrary inclusion. Finally, Chapter 4 develops and examines an application of this methodology to non-linear finite deformation problems.

2.2 Assumptions and governing equations

2.2.1 Governing equations

In this chapter, problems will assumed to be linear elastic. In Chapter 4, the formulation will be extended to nonlinear problems. The development of the governing equations, including strong and weak forms, follows the derivation in Fish & Belytschko [28]. To begin, four critical assumptions are made:

1. deformations are small;

2. the behavior of the material is linear;

3. dynamic effects are neglected;

4. no gaps or overlaps occur during the deformation of the solid.

Detailed justification for these assumptions is beyond the scope of this work. Linear elasticity has been generally accepted for over 300 years for appropriate problems. The assumption of small deformation is acceptable because it will be constrained to be so. Linear material behavior is a safe assumption for most common materials, especially at low deformations. As for the static analysis, this will be prescribed in the problem definitions. Finally, the fourth assumption is valid as the material is not expected to fracture or crack at the low loads considered.

There are three governing equations in linear elasticity. The first being the kinematic equation, which relates strain to displacements:

$$\varepsilon = \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{bmatrix} = \nabla_s \mathbf{u} = \nabla_s \begin{bmatrix} u_x \\ u_y \end{bmatrix} \tag{2.1}$$

where ε is the strain vector, $\mathbf{u}$ is the displacement vector, and ∇_s is a symmetric gradient operator defined as:

$$\nabla_s = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \tag{2.2}$$

The second governing equation is the equilibrium equation, which essentially requires the sum of the forces acting on the system to be zero.

$$\nabla_s^T \sigma + \mathbf{b} = \mathbf{0} \tag{2.3}$$

In Eqn. (2.3), σ is the stress vector, $\sigma = [\sigma_{xx}, \sigma_{yy}, \sigma_{xy}]^T$, and superscript $[\cdot]^T$ denotes transpose. The final governing equation is the constitutive equation, which gives the material law. This takes the form of:

$$\sigma = \mathbf{D}\varepsilon \tag{2.4}$$

where $\mathbf{D}$ is the 3×3 Hookean matrix dependent on the problem. For an isotropic material, the following two expressions for $\mathbf{D}$ are used for plane stress and plane strain problems, respectively.

- Plan stress:

$$\mathbf{D} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix} \tag{2.5}$$

- Plan strain:

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix} \tag{2.6}$$

In Eqns. (2.5) and (2.6), E is the Young's modulus, and ν is the Poission's ratio for the material.

2.2.2 Strong and weak form

Eqns. (2.1), (2.3), and (2.4) are combined, with the addition of boundary conditions, to write the strong form of the problem as [28]:

$$(S) = \begin{cases} \nabla \cdot \sigma + \mathbf{b} = 0 \text{ on } \Omega \\ \sigma = \mathbf{D}\nabla_s \mathbf{u} \\ \sigma \cdot \mathbf{n} = \mathbf{t} \text{ on } \Gamma_t \\ \mathbf{u} = \bar{\mathbf{u}} \text{ on } \Gamma_u \end{cases} \tag{2.7}$$

where Ω is the domain, $\mathbf{n}$ is the normal vector to the boundary. The domain boundary Γ is made up of $\Gamma = \Gamma_t + \Gamma_u$, the Neumann (traction) and Dirichlet (displacement) boundaries, respectively. The prescribed traction on Γ_t is $\mathbf{t}$, and the prescribed displacement on Γ_u is $\bar{\mathbf{u}}$.

Once again, following the derivation in [28], multiplying by the weight functions, $\mathbf{w}$ and integrating by parts yields the weak form:

$$\begin{aligned} &\text{Find } \ \mathbf{u} \in U \ \text{ such that} \\ &\int_\Omega (\nabla_s \mathbf{w})^T \mathbf{D}\nabla_s \mathbf{u}\, \mathrm{d}\Omega = \int_{\Gamma_t} \mathbf{w}^T \bar{\mathbf{t}}\, \mathrm{d}\Gamma + \int_\Omega \mathbf{w}^T \mathbf{b}\, \mathrm{d}\Omega \quad \forall \mathbf{w} \in U_0 \\ &\text{where } \ U = \left\{ \mathbf{u} \,\middle|\, \mathbf{u} \in H^1, \mathbf{u} = \bar{\mathbf{u}} \ \text{ on } \ \Gamma_u \right\} \quad U_0 = \left\{ \mathbf{w} \,\middle|\, \mathbf{w} \in H^1, \mathbf{w} = 0 \ \text{ on } \ \Gamma_u \right\} \end{aligned} \tag{2.8}$$

where $H^1 \subset C^0$. Though, this subset possesses square integrable derivatives, so not all C^0 functions are H^1.

2.3 Formulation of the eXtended Finite Element Method

In this section, a brief review of XFEM for modeling inclusions and voids is presented. For further reading on the subject the reader is encouraged to see [1, 4, 7, 29–32].

The XFEM extends the displacement field approximation of standard FEM by an 'enrichment' function. The displacement field then takes the form:

$$\mathbf{u}^h(\mathbf{x}) = \sum_i N_i(\mathbf{x})\,\mathbf{u}_i + \sum_j N_j(\mathbf{x})\,\psi(\mathbf{x})\,\mathbf{a}_j \tag{2.9}$$

where, $N_i(\mathbf{x})$ are the standard FEM shape functions (in this work bi-linear shape functions will be used), and $\psi(\mathbf{x})$ is the enrichment function. The degrees of freedom (DOFs) are segregated into the physical nodal displacement, standard FEM, DOFs $\mathbf{u}_i$, and the enriched DOFs $\mathbf{a}_j$. A shifted formulation is also possible, such that the displacement field at each node is identical to the nodal displacements at the classical physical DOFs. The displacement field is then approximated as:

$$\mathbf{u}^h(\mathbf{x}) = \sum_i N_i(\mathbf{x})\,\mathbf{u}_i + \sum_j N_j(\mathbf{x})\left[\psi(\mathbf{x}) - \psi(\mathbf{x}_j)\right]\mathbf{a}_j \tag{2.10}$$

where, $\psi(\mathbf{x}_j)$ is the enrichment function evaluated at node-j. This shifted formulation is used in the numerical examples presented in this thesis, in order to simplify comparisons between solutions.

The enrichment function and additional DOFs are chosen to be consistent with the physics of the problem. For problems involving inhomogeneous materials, such as those considered in this work, the solutions should have continuous displacements, but discontinuous stresses at the material interface(s). Thus, the enrichment function must possess a weak discontinuity over the interface, i.e., it should be a C^0 function with discontinuous first derivative. Enrichment functions are discussed briefly in the following sections, but for a more complete discussion of enrichment functions and their requirements, or enrichment functions for problems other than inhomogeneous materials, see [1, 4, 6, 7, 29–34].

The system of equations solved in XFEM looks quite similar to that of FEM. In the linear elastic case the system of equations to solve is:

$$\mathbf{K}\mathbf{u}_{xfem} = \mathbf{f} \tag{2.11}$$

where $\mathbf{K}$ is the global stiffness matrix, $\mathbf{u}_{xfem}$ is the displacement vector, consisting of both physical and enriched DOFs such that:

$$\mathbf{u}_{xfem} = \begin{Bmatrix} \mathbf{u} \\ \mathbf{a} \end{Bmatrix} \tag{2.12}$$

and $\mathbf{f}$ is the force vector, as defined in FEM. The global matrices and vectors are constructed through an assembly process, similar to FEM. The element stiffness matrix in XFEM formulation may be written as the following block matrix [32]:

$$\mathbf{K}^e_{ij} = \begin{bmatrix} \mathbf{K}^{\mathbf{uu}}_{ij} & \mathbf{K}^{\mathbf{ua}}_{ij} \\ \mathbf{K}^{\mathbf{au}}_{ij} & \mathbf{K}^{\mathbf{aa}}_{ij} \end{bmatrix} \tag{2.13}$$

where $\mathbf{u}$ and $\mathbf{a}$ denote the classical and enriched degrees of freedom, respectively. In this form, the stiffness matrix is constructed in the same way as in classical FEM [32]:

$$\mathbf{K}^{rs}_{ij} = \int_{\Omega^e} \left(\mathbf{B}^r_i\right)^T \mathbf{D}\mathbf{B}^s_j \, \mathrm{d}\Omega \quad (r, s = \mathbf{u}, \mathbf{a}) \tag{2.14}$$

where the B-matrix is the matrix of the shape function derivatives. The subcomponent $\mathbf{B}^{\mathbf{u}}_i$ is therefore the two columns associated with the nodal displacement DOFs, and is identical to the $\mathbf{B}^{\mathbf{u}}_i$ matrix of classical FEM. The B-matrix for the enriched degrees of freedom is defined as [32]:

$$\mathbf{B}^{\mathbf{a}}_i = \begin{bmatrix} \frac{\partial}{\partial x}(N_i\psi) & 0 \\ 0 & \frac{\partial}{\partial y}(N_i\psi) \\ \frac{\partial}{\partial y}(N_i\psi) & \frac{\partial}{\partial x}(N_i\psi) \end{bmatrix} = \left(\mathbf{B}^{\mathbf{u}}_i\right)^T \psi + \begin{bmatrix} N_i\frac{\partial\psi}{\partial x} & 0 \\ 0 & N_i\frac{\partial\psi}{\partial y} \\ N_i\frac{\partial\psi}{\partial y} & N_i\frac{\partial\psi}{\partial x} \end{bmatrix} \tag{2.15}$$

2.4 Enrichment functions

2.4.1 Definitions

Sukumar et. al. [1] proposed the signed distance function obtained directly by the level set function, to model arbitrary inclusions and holes. The signed distance function is defined at a point as the distance from that point to the internal boundary. It is defined to be positive on the exterior of the inclusion/hole, and negative on the interior, as shown in Figure 2.2(b) for the arbitrary "kidney" shaped inclusion of Figure 2.2(a). This function possesses two benefits. First, the absolute value of this function can be used as the enrichment function in XFEM, to produce a weak discontinuity over the material boundary. Thus, the enrichment function is given as $\psi(\mathbf{x}) = \min_{\mathbf{x}_\Gamma} |\mathbf{x} - \mathbf{x}_\Gamma|$, where $\mathbf{x}_\Gamma$ denotes the internal boundary. The enrichment function, for the same kidney shape, is shown in Figure 2.2(c), and its derivatives are shown in Figures 2.2(d) and 2.2(e). The enrichment function shown in Figure 2.2(c) has a "ridge" along the inclusion, which exhibits the weak discontinuity over the material interface boundary.

The signed distance function shown in Figure 2.2(b) is also used to determine which elements are to be enriched. Let the signed distance function be $f(\mathbf{x})$. By definition, $f(\mathbf{x})$ has negative values for points on the interior of the internal boundary, and positive values for exterior points. Thus, for a given element, if $f_{max}(\mathbf{x}) \times f_{min}(\mathbf{x}) < 0$, then the element and all of its nodes are to be enriched because this element must cross the internal boundary (i.e. it has at least one positive, external, node and at least one negative, internal, node).

Several modified versions of this enrichment function have been proposed to deal with certain limitations. For instance, Moës et. al. [5], Fries [6] and Fries and Belytschsko [7] have each proposed enrichment functions that address issues pertaining to blending elements. Well known issues in blending elements (elements that "bridge" between enriched and non-enriched elements) have been found. These modified enrichments take the signed distance function based enrichment, and modify it (in different ways, respectively) such that the enrichment function vanishes away from the interface, and restricts its support region to enriched elements.

2.4.2 Formulations for simple geometries

For simple, radially symmetric shapes, the enrichment function and its derivatives can be formulated in a closed form. The expressions for circular and elliptical inclusions will be given here for reference, since they are later used in a comparative study with the proposed formulation. For details on their derivations see their associated references.

For a circle of radius r centered at (x_c, y_c), the signed distance function at a point (x, y) is given as [1]:

$$f(x,y) = \sqrt{(x - x_c)^2 + (y - y_c)^2} - r \tag{2.16}$$

The derivatives of the enrichment function $\psi = |f(x, y)|$ are given as [1]:

$$\frac{\partial \psi}{\partial x} = \operatorname{sign}(d - r)\,\frac{x - x_c}{d} \tag{2.17a}$$

$$\frac{\partial \psi}{\partial y} = \operatorname{sign}(d - r)\,\frac{y - y_c}{d} \tag{2.17b}$$

where $d = \sqrt{(x - x_c)^2 + (y - y_c)^2}$.

For an ellipse centered at (x_c, y_c), of major axis a, minor axis b, and major axis oriented at an angle θ with the x-axis, an alternate level set function was proposed by Hiriyur et. al. [2]:

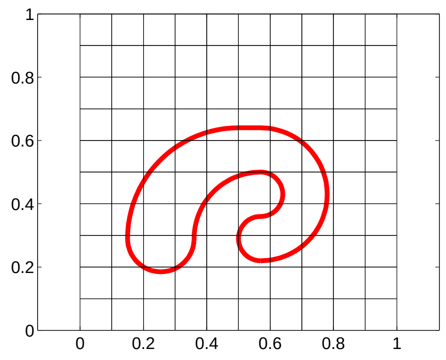
$$f(\xi, \eta) = \sqrt{\frac{\xi^2}{a^2} + \frac{\eta^2}{b^2}} - 1 \tag{2.18}$$

where (ξ, η) are the transformed coordinates $\xi = (x - x_c)\cos\theta + (y - y_c)\sin\theta$ and $\eta = -(x - x_c)\sin\theta + (y - y_c)\cos\theta$. Once again, the derivatives of the enrichment function $\psi = |f(\xi, \eta)|$ is given as [2]:

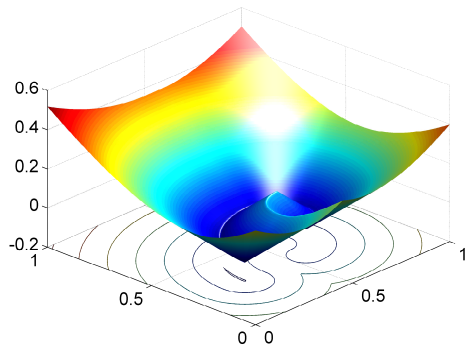
$$\frac{\partial \psi}{\partial x} = \operatorname{sign}(f(\xi, \eta))\,\frac{\frac{\xi}{a^2}\cos\theta - \frac{\eta}{b^2}\sin\theta}{\sqrt{\frac{\xi^2}{a^2} + \frac{\eta^2}{b^2}}} \tag{2.19a}$$

$$\frac{\partial \psi}{\partial y} = \operatorname{sign}(f(\xi, \eta))\,\frac{\frac{\xi}{a^2}\sin\theta + \frac{\eta}{b^2}\cos\theta}{\sqrt{\frac{\xi^2}{a^2} + \frac{\eta^2}{b^2}}} \tag{2.19b}$$

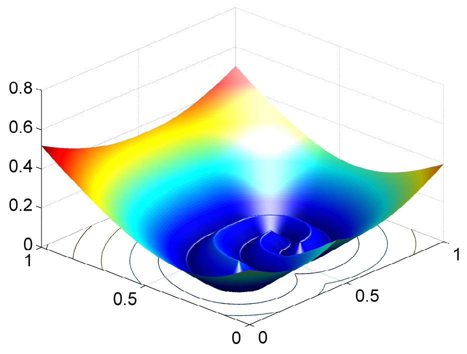
While these expressions are useful, in practical applications it is often necessary to model arbitrarily shaped inclusions. However, for a generic arbitrarily shaped inclusion there is no closed form expression for the level set function. One possible solution is to approximate the shape, for instance, by linear segments. However, for some shapes it may not capture well the boundary. Alternatively, in this chapter a spline-based formulation that is shown to be quite accurate for arbitrary shapes is developed.



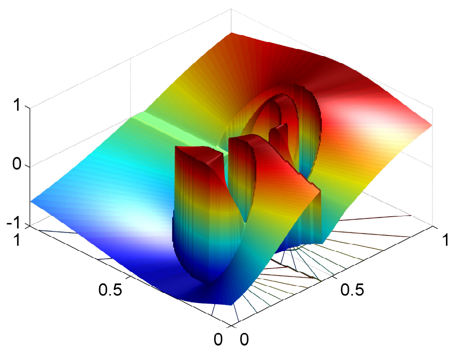
(a) Domain geometry



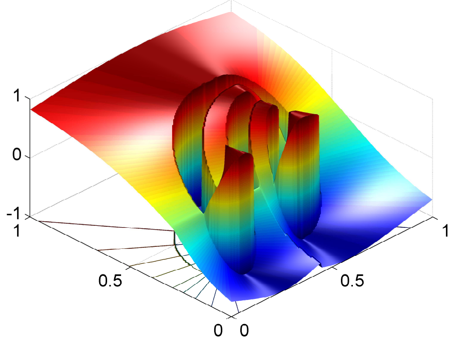
(b) Signed distance function



(c) Enrichment function



(d) $\frac{\mathrm{d}\psi}{\mathrm{d}x}$



(e) $\frac{\mathrm{d}\psi}{\mathrm{d}y}$

Figure 2.2: Level set functions

Chapter 3

Spline based enrichment formulation for arbitrary shapes

3.1 Cubic spline interpolation

3.1.1 Definitions and formulation

In order to model the arbitrary interfaces, it is assumed that in the most general case the interface is either given, or can be described as, a set of discrete points:

$$\mathbf{x}_\Gamma = \left\{ \begin{array}{c} \left(x_1^{inc}, y_1^{inc}\right) \\ \left(x_2^{inc}, y_2^{inc}\right) \\ \vdots \\ \left(x_{ndp}^{inc}, y_{ndp}^{inc}\right) \end{array} \right\} \tag{3.1}$$

where $\left(\mathbf{x}^{inc}, \mathbf{y}^{inc}\right)$ are the x and y coordinates of the discrete points along the inclusion, and ndp is the number of discrete points used to describe the inclusion.

Cubic splines are then used to interpolate the shape of the inclusion. It is well known that polynomial interpolation for large ndp results in high errors; thus piecewise polynomial interpolation is often used [35]. Higher order splines or other formulations are viable options as well. One particular method for modeling curves and curved surfaces that has gained recent attention are Non-Uniform Rational B-Splines (NURBS) [36, 37]. NURBS would possess the added benefit of integration with CAD models widely used in practice.

Cubic splines are chosen (a) for their tried and tested accuracy as an interpolating polynomial, (b) because only data values are required (as opposed to additional information on derivatives), and (c) because they preserve continuity up to and including second derivatives (i.e. C^2 continuous).

Given data points (x_i, y_i), a cubic spline $P(x)$ is constructed of piece-wise cubics $s_i(x)$ of the form:

$$P(x) = \begin{cases} s_1(x) = a_1(x-x_1)^3 + b_1(x-x_1)^2 + c_1(x-x_1) + d_1 & x_1 \le x \le x_2 \\ \vdots & \vdots \\ s_i(x) = a_i(x-x_i)^3 + b_i(x-x_i)^2 + c_i(x-x_i) + d_i & x_i \le x \le x_{i+1} \\ \vdots & \vdots \\ s_n(x) = a_n(x-x_n)^3 + b_n(x-x_n)^2 + c_n(x-x_n) + d_n & x_n \le x \le x_{ndp} \end{cases} \tag{3.2}$$

where the $s_i(x)$ must satisfy the following conditions [35]:

$$\begin{aligned} s_i(x_i) &= y_i \quad \forall\, i = 1,2,...,n \\ s_i(x_{i+1}) &= y_{i+1} \quad \forall\, i = 1,2,...,n \\ \left.\frac{\mathrm{d}s_i}{\mathrm{d}x}\right|_{x=x_{i+1}} &= \left.\frac{\mathrm{d}s_{i+1}}{\mathrm{d}x}\right|_{x=x_{i+1}} \quad \forall\, i = 1,2,...,n-1 \\ \left.\frac{\mathrm{d}^2 s_i}{\mathrm{d}x^2}\right|_{x=x_{i+1}} &= \left.\frac{\mathrm{d}^2 s_{i+1}}{\mathrm{d}x^2}\right|_{x=x_{i+1}} \quad \forall\, i = 1,2,...,n-1 \end{aligned} \tag{3.3}$$

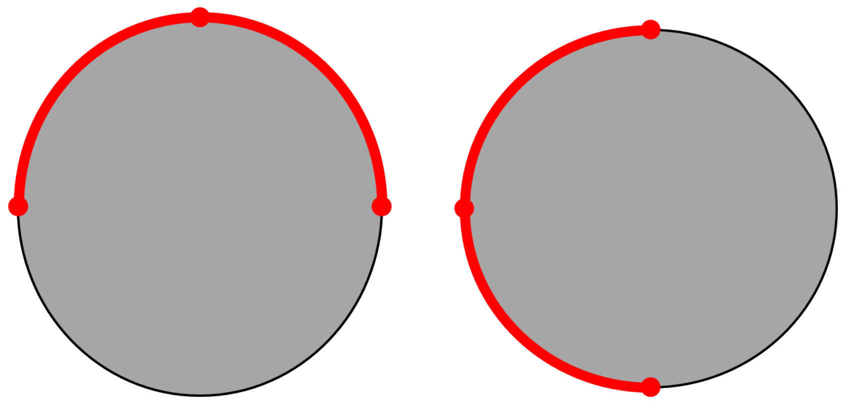
For ndp data points, this yields $n = ndp - 1$ cubic functions, with $4n$ unknowns. The conditions in Eqn. (3.3) give only $4n-2$ linear equations, and thus two additional conditions are necessary to solve for the $4n$ unknowns. Typically, this is accomplished by a so-called *natural spline* with the second derivative set to zero at the two boundaries, though many other configurations exist [35].

3.1.2 Parametric cubic splines

The above formulation works when the abscissae x_i are distinct, and the curve is a *well defined* function, i.e. for any x_i there can be only one y_i. By definition, however, the internal boundary around an inclusion is a closed-loop, and thus is not a well defined function (even if the given x_i happen to be unique). There are two ways to address this issue:

- For a small, ordered and contiguous, subset of points $\left(\hat{\mathbf{x}}^{inc}, \hat{\mathbf{y}}^{inc}\right) \subset \left(\mathbf{x}^{inc}, \mathbf{y}^{inc}\right)$ a localized spline interpolation can be constructed. In this case, the localized spline will have to be chosen as either $y = P(x)$ or $x = P(y)$, depending on the orientation of the subset $\left(\hat{\mathbf{x}}^{inc}, \hat{\mathbf{y}}^{inc}\right)$. For example, the three points in Figure 3.1(a) depict a horizontally ($y = P(x)$) oriented subset, while Figure 3.1(b) depict a vertically ($x = P(y)$) oriented subset.

- Alternatively, a parametric spline interpolation can be constructed such that $[x, y] = \mathbf{P}(t)$. This formulation is "orientation-independent," and can describe closed, self-intersecting, and virtually any arbitrary curve.



(a) Horizontally oriented subset of points, approximated as $y = P(x)$

(b) Vertically oriented subset of points, approximated as $x = P(y)$

Figure 3.1: Subset orientation and scalar spline formulations

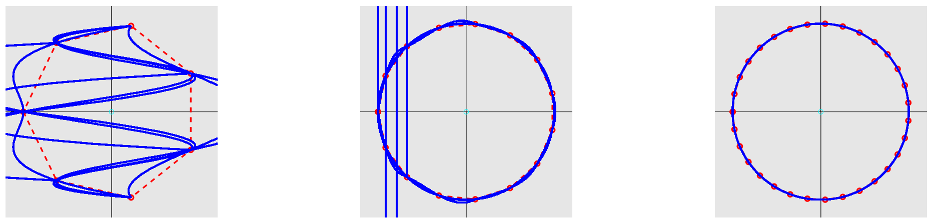
Figure 3.2 and 3.3 show spline curves for the two formulations, respectively, for a circular shape with varying number of discrete points. In these figures, the red dots show the discrete points used for interpolations, the dashed red lines show straight-line interpolations between these points, and the solid blue curves show the spline interpolations. As can be seen in Figure 3.2, the first approach is unstable and inaccurate for small *ndp*. The second approach, however, is extremely stable and accurate, independent of *ndp*, as can be seen in Figure 3.3. Thus this approach is adopted in this study. The parametric spline $\mathbf{P}(t)$ is defined in the same way as Eqn. (3.2). However, now it is a vector-valued function of a parameter t comprising two uncoupled cubics in x and y, i.e. the individual cubics $s_i(x)$ in Eqn. (3.2) are now vector valued functions $\mathbf{s}_i(t)$ of the form:

$$\mathbf{s}_i(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} a_{xi}t^3 + b_{xi}t^2 + c_{xi}t + d_x \\ a_{yi}t^3 + b_{yi}t^2 + c_{yi}t + d_{yi} \end{bmatrix} \tag{3.4}$$

For more information on the derivation, formulation, and implementation of splines, see [35, 38–40].

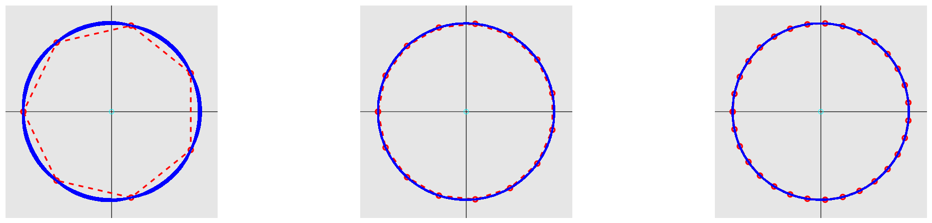
3.1.3 Localized splines

The formulation presented in this chapter uses a localized spline made up of 7 local points. As shown in Figure 3.4, it is clear that 3- and 5-point splines (solid green and dash-dotted blue curves, respectively) do not approximate the curve well enough, but that 7-point splines (red dashed curve) converge quite well. This is especially true between nodes 4 and 6, which is the region that is of interest. If the closest discrete point is node 5, then the *actual* closest point on the curve *must* lie between points 4 and 6. As the errors in piece-wise cubic splines are greatest closest to its end points, the 7-point spline pushes this error far away from the region-of-interest, and better approximates the curve at the center (near node 5).



(a) 8 Points (b) 16 Points (c) 32 Points

Figure 3.2: Scalar spline formulation (e.g. $y = P(x)$). Red dots show the discrete points, red dashed lines show staight-line interpolations between the discrete points, and the solid blue curves show the spline interpolations using these discrete points.



(a) 8 Points (b) 16 Points (c) 32 Points

Figure 3.3: Vectorized parametric spline formulation (i.e. $[x, y] = \mathbf{P}(t)$). Red dots show the discrete points, red dashed lines show staight-line interpolations between the discrete points, and the solid blue curves show the spline interpolations using these discrete points.

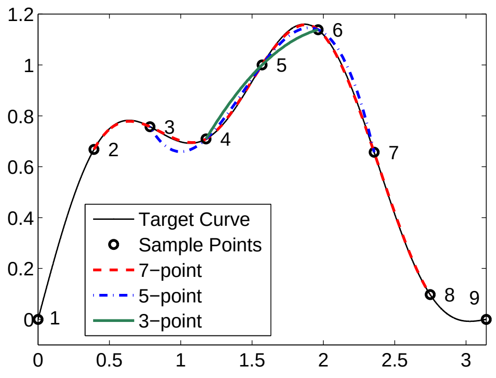


Figure 3.4: Localized splines approximating a target curve (black solid curve), centered on node 5: black circles are the sample points used; solid green curve is a spline constructed from 3 points; dash-dot blue line is constructed from 5 points; and dashed red curve is constructed from 7 points. The 7-point spline approximates the curve much more closely, especially in the region-of-interest between nodes 4 and 6, with errors closer to the boundaries at nodes 2 and 8.

3.2 Calculation of signed distance function

There is only one part of XFEM that depends on the geometry of the problem, and that is the formulation of the level set function. The level set function determines which nodes are enriched, and defines the enrichment function (see section 2.4 for more details). Thus, to solve systems with arbitrarily shaped inclusions, it is necessary to develop the signed distance function numerically as it is not available analytically. The absolute value of this numerical function is then the enrichment function. Formulation of the signed distance function (and consequently the enrichment function) can be broken into two separate tasks: first, the *magnitude* must be determined, and second, the *sign* must be determined. These separate steps are first discussed independently in the following subsections, and then the full algorithm is presented. The formulation and calculation of the derivatives of the enrichment function are then presented in the following section.

3.2.1 Determining the magnitude of the signed distance function

The first step in computing the signed distance function is determining its magnitude. To do so, it is necessary to determine the distance from a point (x_p, y_p) to the internal boundary modeled by the spline curve $\mathbf{x}_\Gamma = \mathbf{P}(t)$. Finding the distance from a point to a curve is a well posed problem in differential calculus, though a closed form solution does not always exist. The distance d from a point (x_p, y_p) to the spline curve is calculated as:

$$d(t) = \sqrt{(x_p - x_\Gamma(t))^2 + (y_p - y_\Gamma(t))^2} \tag{3.5}$$

where $x_\Gamma(t)$ and $y_\Gamma(t)$ are the x and y components of the spline curve of Eqn. (3.4). The closest point corresponds to the value of t which minimizes d. The minimization of d results in the following polynomial equation in t, whose solution gives the optimal value of t.

$$C_1 t^5 + C_2 t^4 + C_3 t^3 + C_4 t^2 + C_5 t + C_6 = 0 \tag{3.6}$$

where:

$$\begin{aligned}
C_1 &= 3a_x^2 + 3a_y^2 \\
C_2 &= 5a_y b_y + 5b_x a_x \\
C_3 &= 2b_x^2 + 2b_y^2 + 4a_y c_y + 4c_x a_x \\
C_4 &= -3y_p a_y - 3a_x x_p + 3b_y c_y + 3b_x c_x + 3a_y d_y + 3d_x a_x \\
C_5 &= -2y_p b_y + c_y^2 + 2b_y d_y - 2x_p b_x + c_x^2 + 2b_x d_x \\
C_6 &= -y_p c_y + c_y d_y - x_p c_x + c_x d_x
\end{aligned} \tag{3.7}$$

where the $a_x, a_y, b_x, b_y, c_x, c_y, d_x,$ and d_y are the coefficients from the spline curve (Eqn. (3.4)). Here, the subscript i has been dropped for simplicity. More detailed derivation of Eqn. (3.8) is given in Appendix A.

Since this is a *quintic* equation, which, in general, has no closed form solution, a numerical root finder must be used. An analogous expression for scalar splines may be found in [10].

3.2.2 Determining the sign of the signed distance function

The next step after computing the magnitude of the distance function, is to determine whether the value is positive or negative, depending on whether the point is inside or outside the inclusion/void. Several methods exist for determining if a point lies within a polygon [41]. One such method is ray-crossing: starting from a known point, a path to the point in question is drawn; if the number of times the path crosses the interface is even, then the target point is on the same side of the interface as the known point, as depicted in Figure 3.5(a). This method can however result in inaccuracies when the point is very close to a side of the polygon, due to numerical approximation errors, or when the path crosses a vertex of the polygon. Another method uses the residue theorem: if the angle between two vectors going from the point in question (x_p, y_p) to two successive points on the polygon $\left(x_i^{inc}, y_i^{inc}\right)$ and $\left(x_{i+1}^{inc}, y_{i+1}^{inc}\right)$ is θ_i, then:

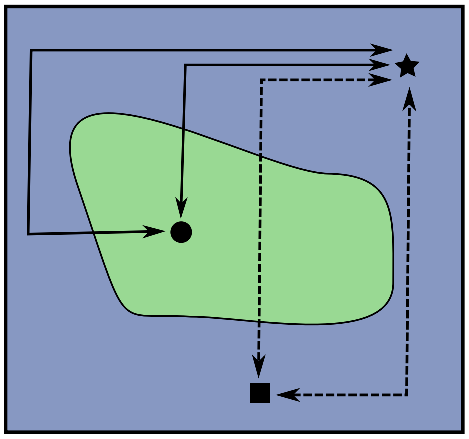
$$\sum_{i}^{ndp} \theta_i = \begin{cases} 0 & \mathbf{x}_p \text{ outside } \mathbf{x}_\Gamma \\ 2\pi & \mathbf{x}_p \text{ inside } \mathbf{x}_\Gamma \end{cases} \tag{3.8}$$

This method is depicted in Figure 3.5(b). The latter is conveniently implemented in MATLAB in the `inpolygon` function [42].

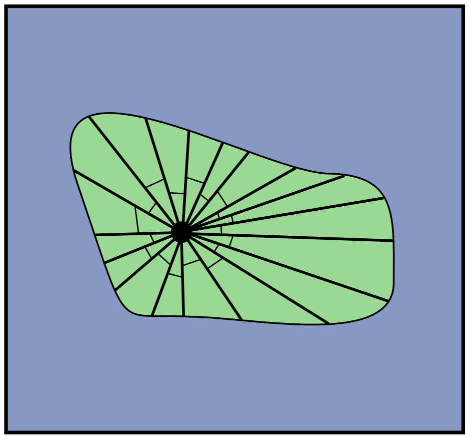
Direct implementation of either of these on the *ndp* discrete points $\left(\mathbf{x}^{inc}, \mathbf{y}^{inc}\right)$ would however fail to use any additional information given by the splines. To this end, it is proposed to add the closest point on the curve (obtained in Section 3.2.1) to the vector of discrete points (Eqn. (3.1)). Adding this point before implementing a point-in-polygon algorithm will describe the local curvature well enough to determine the correct sign of the signed distance function at the point in question. This is illustrated in Figure 3.6, where the value of the signed-distance function at the point denoted by the star is being calculated. The closest point on the red spline curve is found to be the red circle. This point is then added into the vector of discrete points (blue circles). The dashed blue lines represent this new polygon, showing that the point is correctly determined as inside the polygon. On the other hand, excluding the closest point (red point in Figure 3.6), the polygon is defined by the solid lines, and the point is incorrectly found to be outside of the polygon, thus giving an incorrect sign to the signed distance function.

3.2.3 Level set function algorithm

With both the magnitude and sign of the distance function determined as above, the level set and enrichment functions are fully defined. In this section, we propose an algorithm based on the preceding formulations, which computes the level set function for an inclusion/void defined by a set of discrete points (Eqn. (3.1)).



(a) Ray-crossing method: Starting from a known point (black star), a "ray" is drawn to the point in question. If the number of times that the ray crosses the internal boundary is even (black square, dashed lines), then the point is on the same side of the boundary; if the ray crosses an odd number of times (black circle, solid lines), then the point is on the opposite side of the boundary.



(b) Angle sum method: If the sum of the angles between the rays connecting the point in question with the discrete points on the boundary is 2π, then the point is inside the boundary; if the sum is equal to 0, then the point is outside the boundary.

Figure 3.5: Point-in-polygon methods

Given a point (x_p, y_p) (denoted by the star in Figure 3.7), the closest discrete point $\mathbf{x}_{min}^{inc}$ (denoted by the yellow circle and dashed arrow in Figure 3.7) among all the discrete points describing the internal boundary $\left(\mathbf{x}^{inc}, \mathbf{y}^{inc}\right)$ (denoted by the red circles in Figure 3.7), is first determined. Then, a localized parametric spline $\mathbf{P}\left(t\right)$ is constructed centered about this point $\mathbf{x}_{min}^{inc}$ (Section 3.1.2). Alternatively, the localized splines can be generated and stored earlier. This yields about a 40% reduction in computation time, but higher memory usage. The closest point on this spline curve is then determined (the solid arrow in Figure 3.7), using a numerical nonlinear equation solver (Section 3.2.1). This point is then augmented into the vector of discrete points, and used to determine the sign of the level set function (Section 3.2.2). Algorithm 1 summarizes the steps of this algorithm.

While the spline approach presented in this chapter is applied to 2D problems, an extension to 3D is certainly possible. One should especially note that the parametric, vector valued splines lend themselves nicely to this extension [40]. Issues with closed surfaces can be avoided by using this formulation. A 3D spline could be constructed as a function of either one ($\mathbf{x} = \mathbf{P}\left(t\right)$) or two ($\mathbf{x} = \mathbf{P}\left(t, s\right)$) parameters, where $\mathbf{x}$ now has 3 components: $[x, y, z]^T$. The 3D algorithm would conceptually be very similar to the 2D algorithm, with the biggest issues arising in determining the sign of the Level Set Function. Specifically, determining

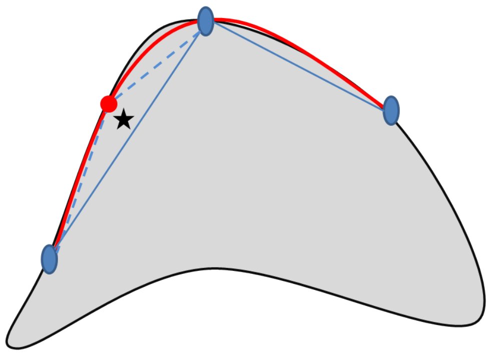


Figure 3.6: Determining the sign of level set function with splines, for the point denoted by a black star: *blue dots* are discrete boundary points; the *red curve* is the spline approximation; the *red dot* is the closest point on the spline curve. Using only the linear approximation (solid blue lines), the incorrect sign would be given; adding the red dot, and using the dashed blue lines, will give the correct sign.

the local 'vector' of points and creating the localized spline. The extension of these algorithms to 3D will be considered for future work.

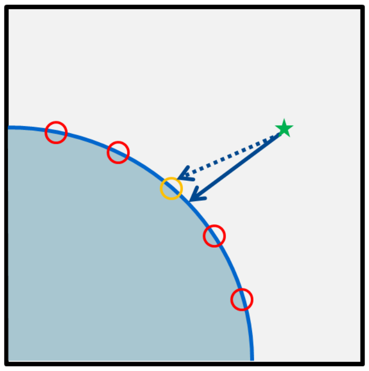


Figure 3.7: Level set generation: The value at a point (green star) is found by first finding the closest discrete boundary point (yellow circle, dashed arrow), then creating a spline curve (blue curve) through the closest point and its neighbors (red circles); the closest point on this curve is then found (solid arrow).

Algorithm 1 Determining the signed distance function for arbitrarily shaped inclusions

Given a set of discrete points $\left(\mathbf{x}^{inc}, \mathbf{y}^{inc}\right)$*, find the value of the signed distance function at a point* (x_p, y_p)*.*

1. **Find** the discrete point, $\mathbf{x}^{inc}_{min}$, that is closest to $\mathbf{x}_p$ by solving:

$$\mathbf{x}^{inc}_{min} = \arg\min_{\mathbf{x}\in\mathbf{x}^{inc}} ||\mathbf{x}_p - \mathbf{x}||_2 \tag{3.9}$$

2. **Construct** a localized spline $\mathbf{P}\left(t\right)$ centered on $\mathbf{x}^{inc}_{min}$ using the formulation in Section 3.1.2 and Eqn. (3.4).

3. **Solve** Eqn. (3.6) for the closest point on the spline curve, $\tilde{\mathbf{x}}_{min} = \mathbf{P}\left(t_{min}\right)$

4. **Add** the closest point, $\tilde{\mathbf{x}}_{min}$, back into the vector of discrete boundary points, $\mathbf{x}^{inc}$, to create an "extended" vector of boundary points, $\tilde{\mathbf{x}}^{inc}$

5. **Determine** the sign of the signed distance function using a point-in-polygon subroutine with the "full" $\tilde{\mathbf{x}}^{inc}$, as described in Section 3.2.2.

3.3 Enrichment function differentiation

The XFEM formulation presented above depends not only on the enrichment function, but its derivatives, as well. Namely, the B-matrix in Eqn. (2.15) depends on these derivatives. As derived above in Eqns (3.5), (3.6), and (3.8), the enrichment function at a point (x, y) is given as:

$$\psi(x,y) = \sqrt{\left(x - x_\Gamma\left(t_{min}\right)\right)^2 + \left(y - y_\Gamma\left(t_{min}\right)\right)^2} \tag{3.10}$$

where t_{min} is the minimum point found by solving Eqns. (3.6) and (3.8). To find the derivative $\frac{\partial\psi}{\partial x}$, we take the limit:

$$\frac{\partial\psi}{\partial x} = \lim_{h\to 0} \frac{\psi(x+h,y) - \psi(x,y)}{h} \tag{3.11}$$

In general, t_{min} will differ depending on x and y — i.e. t'_{min} found for the point $(x+h, y)$ will be different from the original t^0_{min} found for the point (x, y). However, in the limit that $h \to 0$, we can say that $t'_{min} \to t^0_{min}$. With this assumption, t_{min} is constant in x and y, and thus Eqn. (3.10) can directly be differentiated to:

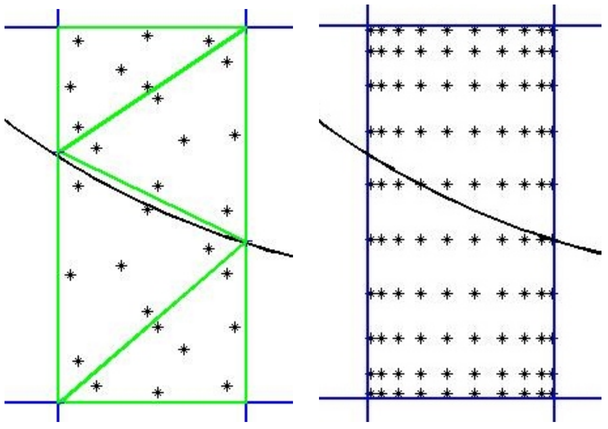
$$\frac{\partial\psi}{\partial x} = \frac{x - x_\Gamma\left(t_{min}\right)}{\psi(x,y)} \tag{3.12}$$

and, similarly:

$$\frac{\partial\psi}{\partial y} = \frac{y - y_\Gamma\left(t_{min}\right)}{\psi(x,y)} \tag{3.13}$$

3.4 Numerical integration

Many XFEM codes use a “discontinuous Q4” element, or “subdomain decomposition” [30, 43]. A discontinuous Q4 element decomposes the Q4 quad into triangular elements that conform to the internal boundary, and uses the Gauss points of these triangles (Figure 3.8(a)). This quadrature rule, however, gives rise to sensitivity issues when comparing two XFEM solutions, as slightly different enrichment functions (analytical vs. numerical, in this study) result in different Gauss points being used in each solution. Instead, for the convergence analyses in this work a uniform Gaussian quadrature (of varying degree) is used for both the “classical” and numerical formulations (Figure 3.8(b)), so as to ensure that the Gauss points are consistent between the two solutions being compared. The use of such high-order Gaussian quadrature in lieu of subdomain decomposition is present in the literature [30]. A convergence study of Gaussian, as well as Newton-Cotes, quadrature on C^0 continuous functions is carried out in Appendix B.



(a) Discontinuous Q4 quad (b) Order 10 Gaussian Quadrature

Figure 3.8: Integration techniques for enriched elements

3.5 Convergence analysis

In this section, convergence analyses for the proposed formulation are carried out. First, the convergence of the numerical formulation of the level set function to a known analytical level set function is tested. Then, the numerical differentiation of this known function is examined. Following that, a full XFEM implementation of the proposed methodology is compared to a "classical" XFEM formulation for simple inclusions. Finally, the proposed methodology is compared to FEM for an arbitrary inclusion.

3.5.1 Level set function convergence

To test the convergence of the proposed numerical approximation of the level set function, a circular inclusion of radius 0.25 in a 1×1 square domain is considered (Figure 3.9). The analytical value of the level set function is first obtained using Eqn. (2.16). The convergence of the spline based level set function is then compared with (i) a simple method based on discrete points describing the inclusion boundary, and (ii) a linear interpolation method. For the convergence analysis, the number of points used to discretize the shape of the inclusion is varied, and for each discretization, the $||L||_2$-norm of the relative error over a 32×32 mesh of the domain is computed as:

$$\epsilon_{rel} = \frac{||\mathbf{g}_{approx} - \mathbf{g}_{exact}||_2}{||\mathbf{g}_{exact}||_2} \tag{3.14}$$

where $\mathbf{g}_{approx}$ is a vector of the "approximate" values for the quantity of interest (in this case, the level set function values), and $\mathbf{g}_{exact}$ is a vector of the "exact" values for the quantity of interest.

As illustrated in Figure 3.10, all three methods are convergent as the inclusion is increasingly discretized.

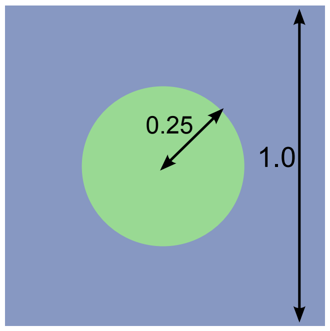


Figure 3.9: Problem geometry

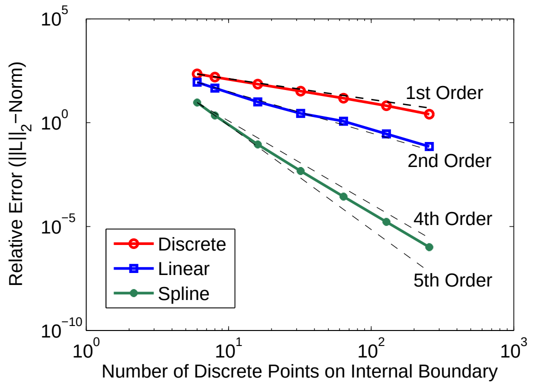


Figure 3.10: Convergence of level set function

Also as expected, the linear interpolation converges faster than the discrete points, and the spline interpolation converges many orders of magnitude faster than the other two methods. Specifically, the discrete representation converges roughly linearly; the linear approximation converges roughly quadratically; and the spline based method converges at a rate between 4th and 5th order. Figure 3.10 shows that the proposed spline method is more accurate as it converges at a higher rate than the linear or discrete representations for any given number of representative boundary points.

3.5.2 Differentiation convergence

The derivatives in Eqns. (3.12) and (3.13) are now tested for convergence, using the same circle shown in Figure 3.9. The differentiation of the proposed numerical level set function is performed for varying discretizations of the inclusion boundary and compared to the analytical derivatives of Eqns. (2.17a) and (2.17b). These results are presented in Figure 3.11. The derivatives proposed in Eqns. (3.12) and (3.13) exhibit a cubic rate of convergence, with respect to the boundary discretization.

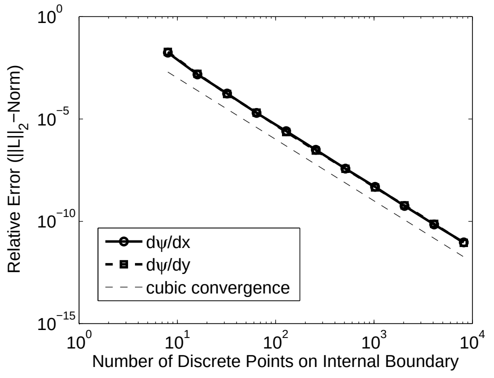


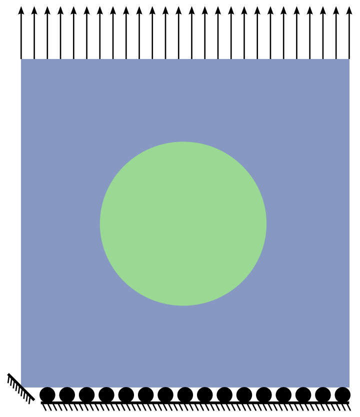
Figure 3.11: Convergence of enrichment derivatives

3.5.3 Classical XFEM comparison for circular inclusion

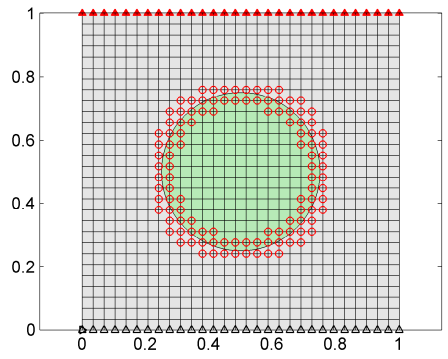
The three numerical approximations to the level set function considered in Section 3.5.1, i.e. using only the distinct points defining the inclusion boundary, and linear and spline interpolations of the inclusion boundary, are now compared with the “classical” XFEM formulation (Eqns. (2.16), (2.17a), and (2.17b), for circular inclusions) in this section. The same circular inclusion and domain are considered as in the previous sections (Figure 3.9). A 32×32 mesh is adopted for both the classical and proposed XFEM approaches, as shown in Figure 3.12(b), with the enriched nodes and elements highlighted in the figure. An order 20 Gaussian quadrature and a 2nd order Gaussian quadrature are used for numerical integration in the enriched and non-enriched elements, respectively.

The “matrix” material around the inclusion has a Young’s modulus of 10^5 units and a Poisson’s ratio of 0.3, while the inclusion material has a Young’s modulus of 10^6 units and a Poisson’s ratio of 0.3. The 2D problem is considered to be plane stress. The bottom edge is on a “roller” with vertical displacement constrained, and the bottom left point is fixed. The top edge is then subjected to a vertical, upwards, force of 500 units. The problem geometry and boundary conditions are shown in Figure 3.12(a).

Figure 3.13(a) shows the convergence behavior, comparing the “numerical” algorithms with the “classical” XFEM solution. Figure 3.13(b) shows the run times for all three numerical-approximation based methods. It is evident that the proposed spline formulation gives orders of magnitude higher accuracy as compared to the other two methods, for a given number of points describing the inclusion boundary. Thus, for applications in which the number of points used to describe the inclusion must be limited to a minimum, the proposed spline-interpolation based method is optimal. Additionally, when considering computational efficiency, Figure 3.13(b) shows that in order to obtain relative errors below about 10^{-5}, the proposed method requires the least computational cost. In fact, there is very low incremental cost in the proposed method, such that errors on the order or 10^{-12} can be attained with little extra cost.

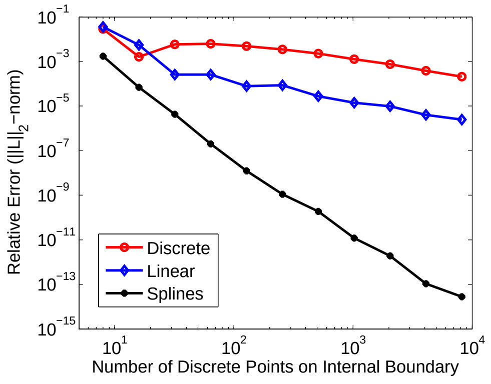


(a) Problem geometry and boundary conditions

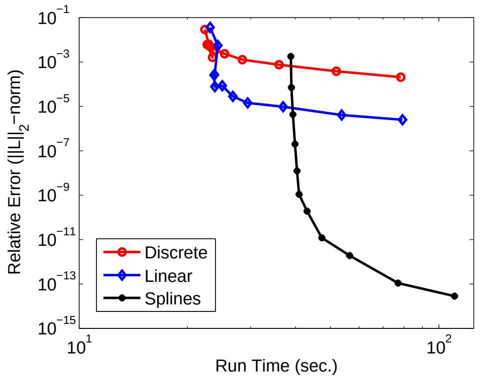


(b) XFEM mesh

Figure 3.12: Circle problem geometry and mesh



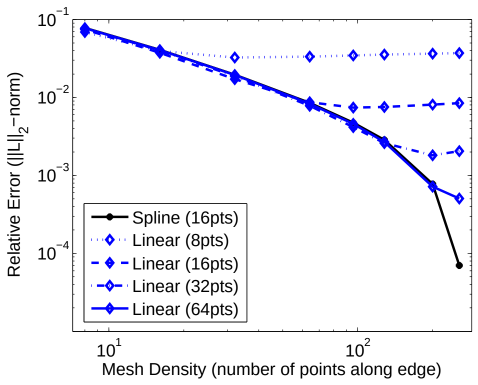
(a) Convergence results



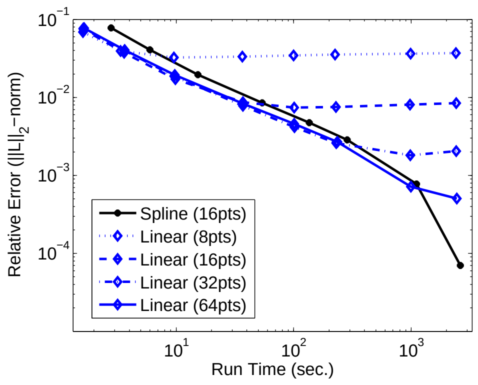
(b) Relative error vs. run time

Figure 3.13: Comparison to 'classical' XFEM formulation

To examine the effects of the mesh refinement, the problem is re-examined with varying meshes (8^2, 16^2, 32^2, 64^2, 128^2, 200^2, and 256^2) using 8, 16, 32, and 64 discrete points along the boundary. The reference solution is obtained using a standard XFEM implementation with an analytical enrichment function [1] on a 256×256 mesh. The results are presented in Figure 3.14 for the linear, and spline representations of the interface. Figure 3.14(a) shows the convergence behavior of refining the mesh for several different boundary point discretizations in the linear case, and compared with 16 boundary points for the splines. Figure 3.14(b) shows the run times for the same cases. It can clearly be seen that the spline formulation is able to achieve higher accuracy than the linear formulations, which with up to 64 boundary points is unable to achieve the same accuracy as the splines with 16 points. Furthermore, there exists a crossing point (see Figure 3.14(b)) where the spline method also becomes more efficient than the linear representation.



(a) Convergence results



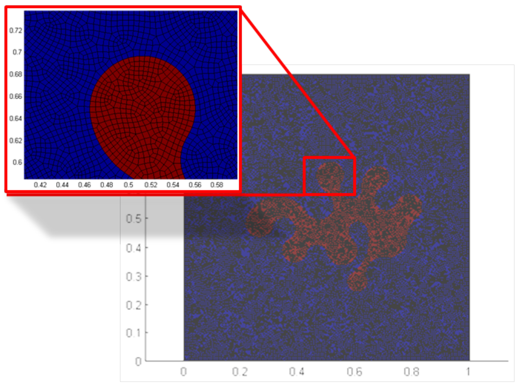
(b) Relative error vs. run time

Figure 3.14: Mesh comparison to 'classical' XFEM formulation

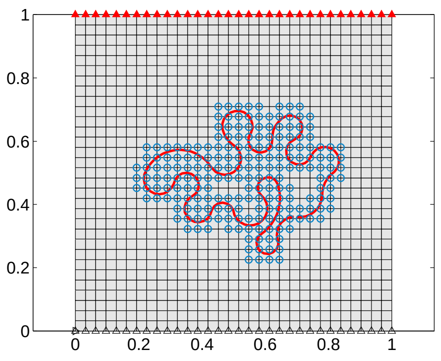
3.5.4 FEM comparison for arbitrary inclusion

In Sections 3.5.1 and 3.5.3 it has been shown that the proposed method converges to the analytical level set function and the 'classical' XFEM solution for simple inclusions. In order to examine the convergence of the proposed method for an arbitrarily shaped inclusion, it is now compared with a finely meshed classical FEM solution. The FEM mesh, shown in Figure 3.15(a) with a zoom-in to show the detail of the mesh, used for comparison has 48,387 nodes and 47,986 elements. The XFEM mesh density and the discretization of the inclusion boundary are both varied, and the XFEM solutions are compared to the FEM reference solution. The material properties, boundary conditions, and problem definition remains the same as in Section 3.5.3, with only the internal boundary shape changing.

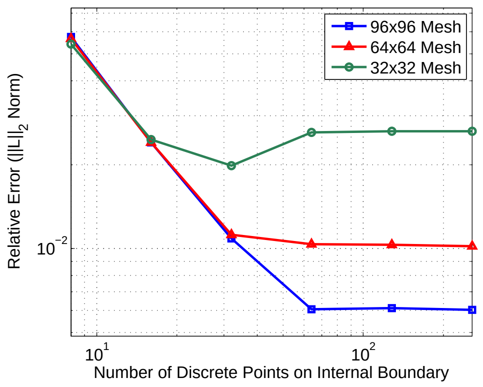
Figure 3.15(c) presents these results for the proposed spline enrichment based XFEM on three meshes: 32×32, 64×64, and 96×96 (Figure 3.15(b) shows the 32×32 XFEM mesh). The solution increases in accuracy with both mesh density and boundary discretization. Nonetheless, there is a clear coupling between the mesh discretization and the boundary discretization. This is evident since an increase in one parameter alone may not result in an increased accuracy - e.g. refining just the mesh density will not always lead to more accurate results. Instead, both the mesh and boundary discretization must be refined in pair.



(a) FEM mesh (47,986 elements)



(b) XFEM mesh (32 × 32)



(c) FEM comparison

Figure 3.15: XFEM-FEM comparison for arbitrary inclusion

3.6 Conclusions

In this chapter, a spline based enrichment function to model arbitrarily shaped inclusions and holes in an XFEM framework was proposed. The convergence of the numerical, spline based, enrichment function to analytical enrichment functions was shown. It was shown to converge between a 4th and 5th order rate. Similarly, the derivatives of the enrichment function were shown to converge at a cubic rate. When compared to an XFEM reference solution for a linear elastic problem with a simple inclusion, an increased accuracy and rapid convergence of the spline method was observed. Varying the mesh and the boundary discretization showed clear convergence for the spline method. Furthermore, it was shown that the spline method was more efficient compared to the discrete or linear representations for some problems. The spline method was then compared with an FEM reference solution for a linear elastic problem with a highly arbitrary inclusion and showed it was able to converge to the true irregular inclusion. This example showed, once again, the convergence of the proposed method.

Chapter 4

Non-linear XFEM formulation for finite deformations

4.1 Definitions and formulation

While the previous chapter has been concerned with linear elastic analysis, one of the major advantages of the XFEM formulation proposed in this work is its adaptability to non-linear problems, specifically problems involving finite deformations. These problems are interesting as the internal boundary depends on the deformation. Hence, the shape of the inclusion may change significantly with the deformation. In terms of XFEM implementations in non-linear applications, the existing literature is comparatively sparse, with the majority of work concentrating on finite deformation in crack problems [44–48]. Although there has been some studies related to heterogeneous materials, they have mostly focused on contact problems [49], explicit dynamics [50, 51], damage and failure [52, 53], and plasticity [54–58]. Furthermore, these studies mostly adopted Lagrangian formulations, and therefore they do not track the evolving material boundaries. In that vein, an Updated Lagrangian formulation is proposed here, since the XFEM formulation proposed in the previous chapter can track the evolving interface. This chapter thus examines the application of the proposed methodology to finite deformation of heterogeneous hyperelastic materials. A combination of the formulations in [59] and [60] is used, and adapted for XFEM in this work, as follows. To this end, the hyperelastic/Neo-Hookean constitutive behavior is first defined.

The Cauchy stresses are defined as [59]:

$$\sigma = \frac{1}{J}\left[\lambda_0 \ln J\mathbf{I} + \mu_0\left(\mathbf{F}\mathbf{F}^T - \mathbf{I}\right)\right] \tag{4.1}$$

where μ_0 and λ_0 are the Lamé constants of the linearized theory (material parameters), $J = \det\mathbf{F}$, and $\mathbf{F}$ is the deformation gradient defined as:

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} \tag{4.2}$$

where $\mathbf{x}$ and $\mathbf{X}$ are the deformed and undeformed coordinates, respectively. Eqn. (4.2) may also be rearranged in terms of displacements [61].

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \left(\frac{\partial \mathbf{X}}{\partial \mathbf{x}}\right)^{-1} = \left(\mathbf{I} - \frac{\partial \mathbf{u}}{\partial \mathbf{x}}\right)^{-1} = \left(\mathbf{I} - \mathcal{B}^T_{xfem}\mathbf{u}_{xfem}\right)^{-1} \tag{4.3}$$

where $\mathcal{B}$ is used to denote the B-matrix with derivatives computed on the deformed or updated configuration. The subscript $xfem$ is used to denote the presence of possible enriched DOFs, such that the B-matrix might take the form of either $\mathcal{B}^{\mathbf{u}}$ or $\mathcal{B}^{\mathbf{a}}$, as in Eqn. (2.15), or as follows for the Updated Lagrangian formulation:

$$\mathcal{B}^{\mathbf{u}} = \frac{\partial}{\partial \mathbf{x}}(\mathbf{N}) \tag{4.4}$$

$$\mathcal{B}^{\mathbf{a}} = \frac{\partial}{\partial \mathbf{x}}(\mathbf{N}\psi) \tag{4.5}$$

The internal nodal forces are now expressed as:

$$f_i^{int} = \int_\Omega \mathcal{B}^T_{i,xfem}\sigma \, \mathrm{d}\Omega \tag{4.6}$$

and the external nodal forces are expressed as:

$$f_i^{ext} = \int_\Omega \mathbf{N}_i^T \mathbf{b} \, \mathrm{d}\Omega + \int_\Gamma \mathbf{N}_i^T \bar{\mathbf{t}} \, \mathrm{d}\Gamma \tag{4.7}$$

where $\mathbf{b}$ and $\bar{\mathbf{t}}$ are the body and traction forces, respectively. From Eqns. (4.6) and (4.7) the residual is calculated as:

$$\mathbf{R} = \mathbf{f}^{ext} - \mathbf{f}^{int} \tag{4.8}$$

This nonlinear system of equations can be solved using Newton's method. To solve for the incremental displacement, the following linearized system is solved:

$$\mathbf{K}_{tan}\mathrm{d}\mathbf{u} = \mathbf{R} \tag{4.9}$$

Following the formulation in [60], the tangent stiffness matrix is decomposed into material and geometric tangent stiffness matrices, as $\mathbf{K}_{tan} = \mathbf{K}_{tan}^{mat} + \mathbf{K}_{tan}^{geo}$, the material and geometric tangent stiffness matrices for an XFEM element being defined as:

$$\mathbf{K}_{tan}^{mat,e} = \int_{\Omega_e} \mathcal{B}^T_{xfem}\mathbf{D}\mathcal{B}_{xfem} \, \mathrm{d}\Omega \tag{4.10}$$

$$\mathbf{K}_{tan}^{geo,e} = \int_{\Omega_e} \mathcal{B}^T_{xfem}\left[\sigma\right]\mathcal{B}_{xfem} \, \mathrm{d}\Omega \tag{4.11}$$

where $\mathbf{D}$ is the constitutive tensor. Note the use of $\mathcal{B}$ (i.e. spatial derivatives taken with respect to the deformed configuration), which is consistent with the Updated Lagrangian formulation. Eqns. (4.3), (4.6), (4.10), and (4.11) fully extend the non-linear FEM formulation to account for the enriched DOFs.

It should be noted that the inclusion boundary may evolve significantly with the deformation and hence the enrichment function should be updated. An outline of the nonlinear XFEM implementation is shown in Algorithm 2. At every Newton iteration, after the current solution $\mathbf{u}_k$ is found, the location of the internal boundary is updated. The enrichment function is then recalculated, or reinitialized, with this new geometry. While using the Level Set Method (LSM) is possible, in the current work a Lagrangian description is employed, where the motion of the interface is tracked incrementally through marker particles placed along the boundary. If instead, the LSMwere employed, there would be a need to define a finite difference grid as well as the advecting velocities (which is cumbersome in the case of nonlinear hyperelasticity). In addition, one would need to interpolate values to gauss points, which may result in higher errors and reduce the effectiveness of the proposed splines approach. To this end, it is more straightforward to simply “re-initialize” the level sets at every incremental step.

Algorithm 2 Newton's Method with updating

```
1. Make an initial guess for the displacement, $\mathbf{u}_0$. Typically, $\mathbf{u}_0 = \mathbf{0}$ is sufficient.
2. **Loop** over each load increment
   (a) **Loop** until convergence, i.e. while $||\mathbf{R}_k|| > tol$, where $\mathbf{R}_k$ is calculated from Eqn. (4.8).
       i. Calculate the enrichment in the current (deformed) configuration, using the formulation of Section 3 and Algorithm 1.
       ii. Calculate the tangent stiffness matrix, $\mathbf{K}_{tan}$ (Eqns. (4.10) and (4.11)), and the internal and external forces, $\mathbf{f}^{int}$ and $\mathbf{f}^{ext}$ (Eqns. (4.6) and (4.7)), respectively, for the current $\mathbf{u}_k$.
       iii. Calculate the residual: $\mathbf{R}_k = \mathbf{f}^{ext} - \mathbf{f}^{int}$
       iv. Solve for the incremental displacement: $\mathrm{d}\mathbf{u} = (\mathbf{K}_{tan})^{-1}\,\mathbf{R}_k$
       v. Update the displacement: $\mathbf{u}_{k+1} = \mathbf{u}_k + \mathrm{d}\mathbf{u}$
       vi. Update internal boundary using Eqn. (4.12) or (4.13).
   (b) **End Loop**
   (c) Increase load
3. **End Loop**
```

One final issue in this implementation is the updating scheme used for the internal boundary. The displacement $\mathbf{u}$ must be interpolated at the discrete points $\left(\mathbf{x}^{inc}, \mathbf{y}^{inc}\right)$ describing the internal boundary from the XFEM nodes. However, the interpolation (Eqns. (2.9) and (2.10)) is dependent on the shape functions (i.e. enrichment functions), which are by definition dependent on the location of the internal boundary. Such an update may be performed explicitly or implicitly. Explicit updating would consider the previous location of the internal boundary, and thus the previous $\mathbf{u}$:

$$\mathbf{x}_{\Gamma,k} = \mathbf{x}_{\Gamma,k-1} + \mathbf{N}\mathbf{u}_{k-1} \tag{4.12}$$

where $\mathbf{x}_{\Gamma,k} = \left(\mathbf{x}_k^{inc}, \mathbf{y}_k^{inc}\right)$ is the array of internal boundary coordinates at iteration k, $\mathbf{x}_{\Gamma,k-1}$ is the previous configuration of the internal boundary, and $\mathbf{u}_{k-1}$ is the previous solution.

Alternatively, an implicit update requires a "nested" nonlinear problem setup in which the residual, defined in Eqn. (4.13), is minimized at each iteration, thus

$$\mathbf{r} = \mathbf{x}_{\Gamma,k} - \mathbf{N}_{xfem}\left(\mathbf{u}_k, \mathbf{x}_{\Gamma,k}\right)\mathbf{u}_k \tag{4.13}$$

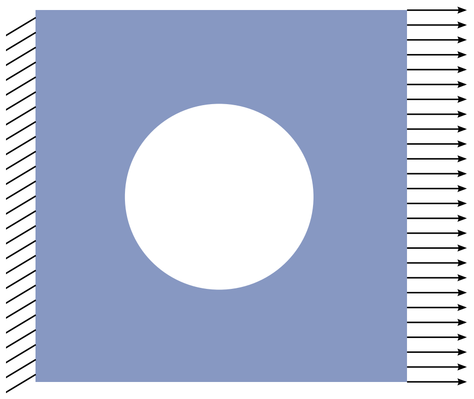
where $\mathbf{N}_{xfem}$ are the XFEM shape functions which are dependent on the displacement as well as the internal boundaries. Thus, at the solution $\mathbf{u}_k$ for iteration k, in order to update the inclusion, we need to solve for $\mathbf{x}_{\Gamma,k}$ which minimizes the residual in Eqn. (4.13). Out of these two options, it is found that the former is more accurate, stable, and computationally efficient, and is thus adopted in the examples below.

4.2 Non-Linear XFEM example case study

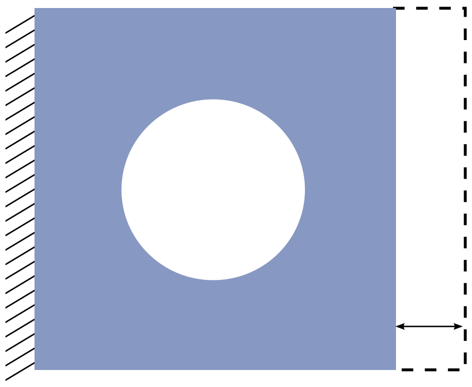
The example problem considered in this study for the nonlinear XFEM implementation assumes a square plate with a circular hole in the center, shown in Figure 4.1. The left edge of the plate is fixed. Two loading conditions are considered: first, a uniform tension is applied on the right edge; then, in a second test, the right edge is first pulled in tension via controlled displacement, followed by a downward force in the middle of the top edge, while constraining the right edge in the stretched configuration, finally followed by an identical upwards force on the bottom edge. Loading condition 1 stretches the plate and hole. Loading condition 2 stretches, bends, and pinches the plate and hole. These two loading conditions are shown in Figures 4.1(a), 4.1(b), 4.1(c), and 4.1(d). Clearly, the combined loading scenario is a difficult example that leads to significant changes in inclusion geometry with the deformation. The material has a Young's Modulus $E = 15 \times 10^6$ units and Poisson's ratio $\nu = 0.3$. The load is applied in increments of 4×10^5 units. Based on the convergence behavior shown in Section 3.5, a reasonable discretization is chosen for both the XFEM mesh (32×32 elements) and the internal boundary ($ndp = 100$). The formulation used for the hole is similar to that in [1] and [62].

In the first loading condition (Figure 4.1(a)), a distributed force is applied along the right edge of the plate. This force is applied in increments of 4×10^5 units. The results for this loading condition are shown in Figures 4.2 and 4.3. Figure 4.2 compares the von Mises stresses on the deformed configurations for the XFEM solution and with a reference FEM solution, for three different load increments. The reference FEM mesh has 794 elements and 878 nodes, and is shown in the right column of Figure 4.2. Figure 4.3 shows the force-displacement curve for several different XFEM mesh densities (all with 100 discretization points on the internal boundary) and the FEM reference solution. The force is measured as the distributed force applied along the right edge of the plate. The displacement is measured as the average horizontal displacement along the right edge of the plate. The proposed methodology agrees reasonably well with the reference FEM solution in all of these comparisons. Figure 4.3 shows that the force-displacement behavior converges quickly to the FEM solution as the mesh is refined. At the final load increment, the $||L||_2$-norm of the relative error between the displacements of the 32×32 mesh XFEM and the FEM reference solution is only 6×10^{-3}.

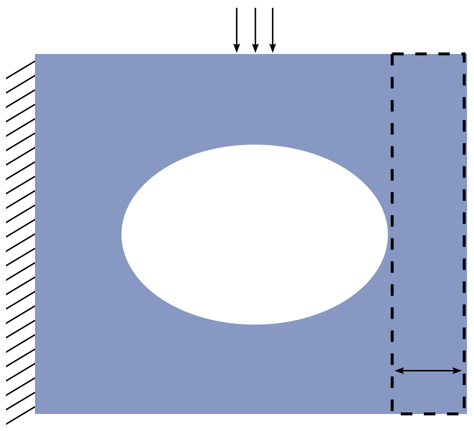
As a final example, the second loading condition shown in 4.1(b), 4.1(c), and 4.1(d) is considered. The right edge is first pulled rightward (via an applied displacement), in this example. The displacement is applied in increments of 0.05 units. Then, after being stretched for 15 increments, the right edge is held in place while a downward force is applied in the center of the top edge in increments of 4×10^5 units, causing bending action. This downward force is applied for 5 steps. Finally, an upwards force on the bottom edge is applied, also of 4×10^5 unit increments, in 4 steps, causing a "pinching" behavior. The nature of the proposed methodology allows the internal boundary to be tracked, even through a complicated evolution, as in this case from a circle, to an ellipse, to a "kidney" shape, and finally to a more complicated shaped hole. This evolution is shown in Figure 4.4. It should be noted that the deformation of the internal geometry is tracked even as it is distorted heavily, with no mesh complications.



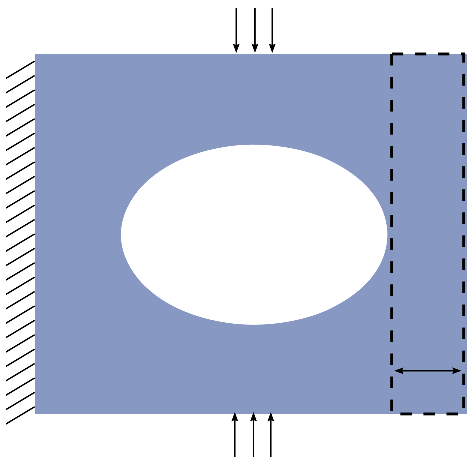
(a) Loading condition 1 (Plate in tension)



(b) Loading condition 2 (Step 1 - Tension phase)

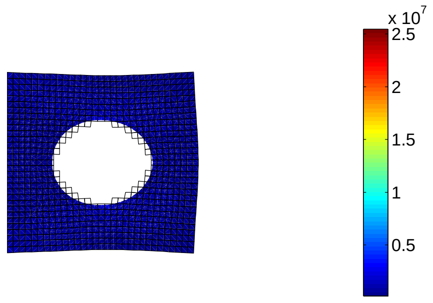


(c) Loading condition 2 (Step 2 - Bending phase)

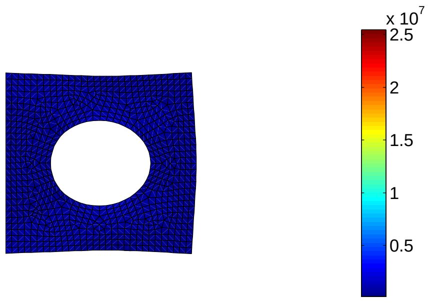


(d) Loading condition 2 (Step 3 - Pinching phase)

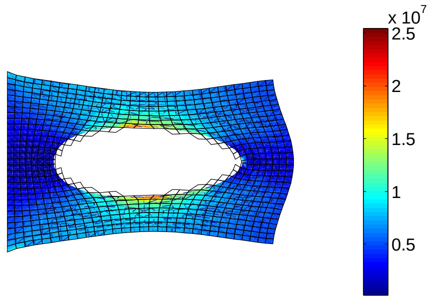
Figure 4.1: Plate with hole geometry and loading conditions



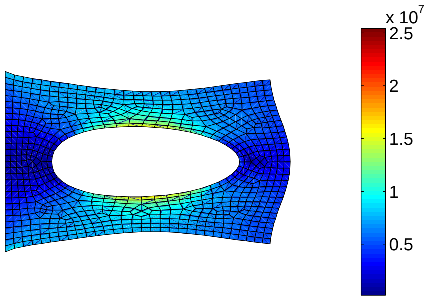
(a) XFEM step 1



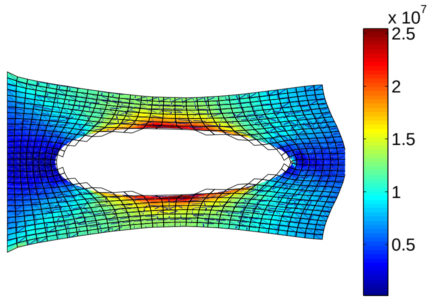
(b) FEM step 1



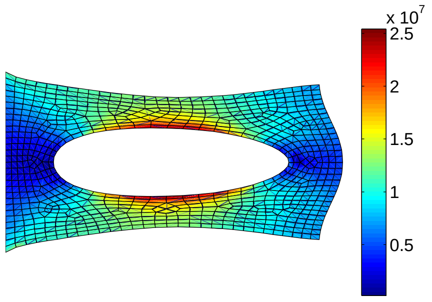
(c) XFEM step 7



(d) FEM step 7



(e) XFEM step 14



(f) FEM step 14

Figure 4.2: von Mises stresses for the plate of Figure 4.1(a)

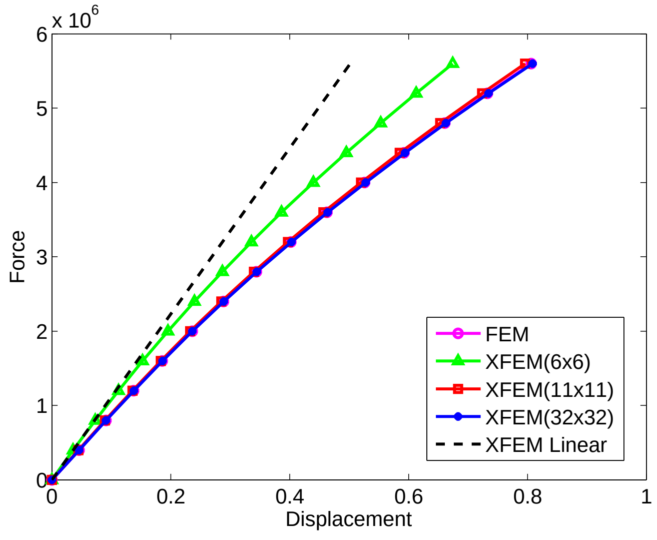
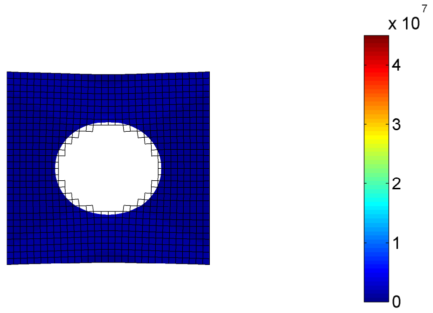
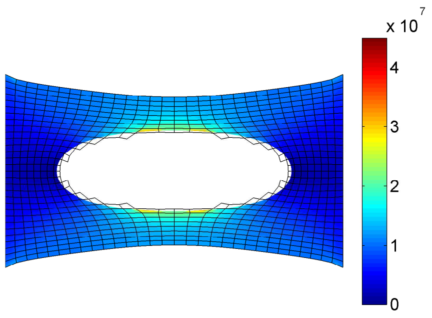


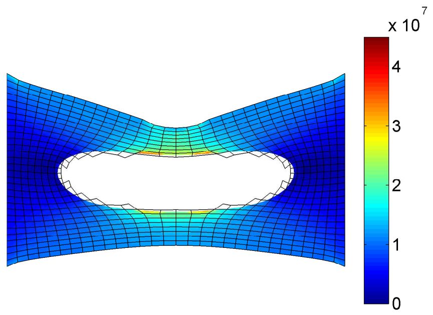
Figure 4.3: Plate with hole force-displacement



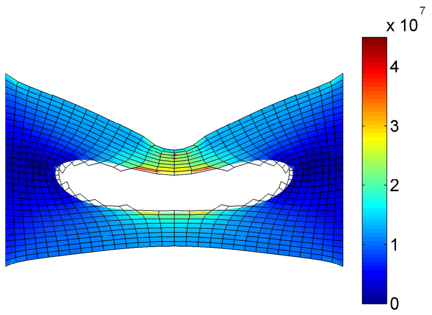
(a) XFEM step 1 (Tension phase)



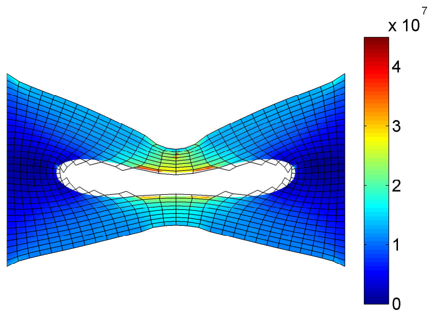
(b) XFEM step 15 (Tension phase)



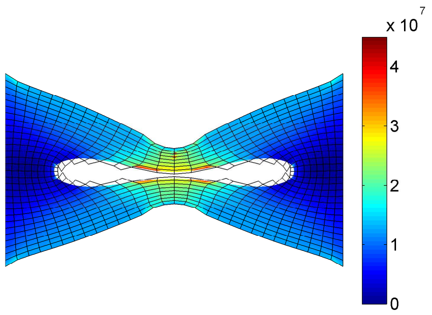
(c) XFEM step 18 (Bending phase)



(d) XFEM step 20 (Bending phase)



(e) XFEM step 22 (Pinching phase)



(f) XFEM step 24 (Pinching phase)

Figure 4.4: Plate with hole in bending and pinching: Von Mises stresses on deformed configuration

4.3 Conclusions

In this chapter, a nonlinear XFEM formulation was developed for hyperelastic Neo-Hookean material using an Updated Lagrangian formulation. Example problems were carried out in which plates with holes were severely deformed, such that the internal hole geometry varied greatly over the deformation. It was shown that the proposed spline formulation of the previous chapter is able to adapt to the deformation with significant changes in geometric description thus illustrating the viability of this modeling approach for arbitrary shaped inclusions, and large deformation problems.

Part II

Simulating random procresses & fields

Chapter 5

Simulation of stochastic processes & fields

5.1 Introduction

Through increases in both the computational power and efficiency of algorithms available to scientists and engineers, it is becoming increasingly possible to consider problems in a fully stochastic framework that had once only been able to be considered deterministically. Monte-Carlo simulation remains at the forefront of these analyses for its robustness. For many classes of problems, Monte-Carlo simulation is the only option to fully characterize the system. This holds true for many problems of practical interest. Take, for example, the structural response to non-stationary seismic ground motion; the characterization of statistically non-homogeneous random media, like Functionally Graded Materials (FGMs); the response of bridges to non-Gaussian loads such as wind fields; or any of these in combination with nonlinearities. All of these problems are commonly, and most effectively, solved using Monte-Carlo simulation.

The bedrock of Monte-Carlo methods is the efficient and accurate simulation of sample realizations. For the examples previously listed, this would translate to generating sample earthquake time histories, sample FGM fields, and sample wind velocity time histories/fields. The Spectral Representation Method (SRM) [63] is widely used for the simulation of sample realizations of stochastic processes, and will be used throughout this dissertation. However, it should be noted that there are other simulation techniques available (e.g. Autoregressive Moving Average (ARMA) models [64–69] and the Karhunen-Loève decomposition [70–73], among others). A review and comparison of the different methods is beyond the scope of this thesis, though it will be said that each has their advantages and disadvantages. The SRM, in particular, is commonly used within the fields of civil engineering and applied mechanics, due in no small part to its nice physical interpretation - namely its basis of the Spectral Density Function (SDF) making use of the very physical frequency domain. Additionally, it is comparatively mathematically simple, as the simulated samples are generated by a finite sum of cosine functions. The remainder of this chapter will outline the SRM in detail for various classes of processes. Stochastic processes and fields can be described by the following four characteristics:

- stationary (homogeneous) vs. non-stationary (inhomogeneous),
- Gaussian vs. non-Gaussian,
- univariate (scalar) vs. multi-variate (vector),
- and one dimensional vs. multi-dimensional.

Following this breakdown, the remainder of this chapter progresses as follows. Section 5.2 discusses the simulation of stationary, univariate, Gaussian stochastic processes in the SRM. This is the most basic case, and the theory can be extended to non-stationary, non-Gaussian, multi-variate, or multi-dimensional cases (or combinations there-of). Section 5.3 covers extension to non-stationarity, Section 5.4 non-Gaussianity, Section 5.5 multi-variate processes, and Section 5.6 multi-dimensionality. These sections are roughly laid out in the order in which they appear in later chapters of this thesis.

5.2 Stationary, univariate, Gaussian stochastic processes

Before describing the SRM, let us define two important quantities. Consider a stationary random process, $X(t)$. The auto-correlation function (ACF), $R_{XX}(t,\tau)$, describes how different time instances are correlated with each other:

$$R_{XX}(t,\tau) = E\left[X(t)\cdot X(t+\tau)\right] \tag{5.1}$$

where t is time, τ is the time lag, and $E[\cdot]$ denotes expected value. For stationary processes, the ACF is only a function of the lag, i.e. $R_{XX}(t,\tau) = R_{XX}(\tau)$. The Spectral Density Function (SDF), $S_{XX}(\omega)$, is a measure of the distribution of the power of the process in the frequency domain, where ω is angular frequency. The SDF serves as the basis for the SRM. For stationary processes, these two quantities, S and R, are related via a Fourier pair known as the Wiener-Khinchin theorem/transform [74–77].

$$S_{XX}(\omega) = \frac{1}{2\pi}\int\limits_{-\infty}^{\infty} R_{XX}(\tau)\, e^{-i\omega\tau}\,\mathrm{d}\tau \tag{5.2a}$$

$$R_{XX}(\tau) = \int\limits_{-\infty}^{\infty} S_{XX}(\omega)\, e^{i\omega\tau}\,\mathrm{d}\omega \tag{5.2b}$$

where $i = \sqrt{-1}$ is the imaginary unit. Therefore, switching between the SDF and ACF is extremely fast and efficient via the Fast Fourier Transform (FFT). Therefore, whether the SDF or ACF is known or estimated, simulation is still possible using the SRM through the efficient transformation between the two quantities.

In the SRM, a sample realization of a stationary and Gaussian, univariate process $X(t)$ can be generated using Eqn. (5.3). For simulation of a random field, the expression would be the analogous but in terms of space and wave-number[1].

$$X(t) = \sqrt{2} \sum_{j=0}^{N-1} \sqrt{2S_{XX}(\omega_j)\,\Delta\omega} \cos(\omega_j t + \phi_j) \tag{5.3}$$

where ϕ_j are independent uniformly distributed random phase angles between 0 and 2π. The frequency domain is discretized as:

$$\Delta\omega = \frac{\omega_u}{N} \tag{5.4a}$$

$$\omega_j = j\Delta\omega \tag{5.4b}$$

The upper cutoff frequency, ω_u is chosen such that $S(\omega > \omega_u)$ can be assumed to be negligible[2]. Formally, this amounts to choosing ω_u such that:

$$2\int_0^{\omega_u} S(\omega)\,\mathrm{d}\omega \geq \alpha \mathrm{Var}(X) \tag{5.5}$$

for some factor, α (e.g. 95%). Eqn. (5.5) makes use of the property of the SDF that its integral over $\omega \in (-\infty, \infty)$ is the variance of $X(t)$. In terms of discretization, it should also be noted that the time increments, Δt, are restricted by the Nyquist frequency in order to avoid aliasing [78]. Thus,

$$\Delta t \leq \frac{2\pi}{2\omega_u} \tag{5.6}$$

The following properties of the simulated samples should be noted, as well:

- The simulated samples are periodic with period [78]:

$$T = \frac{2\pi}{\Delta\omega} \tag{5.7}$$

- Over multiples of this period, or as the length of time approaches infinity, samples are ergodic in the mean and autocorrelation [78]. This is ensured if $S(\omega = 0) = 0$, or through the use of the Frequency Shifting Theorem [79, 80].

[1] For the remainder of this dissertation "processes" and "fields" will be used interchangeably. All of the theory will be developed for random processes, but is equally applicable to random fields.

[2] For the remainder of this dissertation, when considering univariate processes, the subscript will be dropped, and the spectral density function will simply be denoted as $S(\omega)$.

- Through the Central Limit Theorem, the simulated samples are asymptotically Gaussian as $N \to \infty$. Though, it has been shown that N need not be very large before the samples tend to Gaussianity [78].

The FFT can be used to drastically speed up generation of sample functions through the SRM. In order to make use of the FFT, Eqn. (5.3) is rearranged as [78, 81]:

$$X(t) = \mathrm{Re}\left\{ \sqrt{2} \sum_{j=0}^{N-1} \sqrt{2S(\omega_j)\Delta\omega} e^{i\phi_j} e^{i\omega_j t} \right\} \tag{5.8}$$

where $\mathrm{Re}\{\cdot\}$ denotes the real part, and i is the imaginary unit. If a vector, $\mathbf{B}$, is defined as:

$$B_j = 2\sqrt{S(\omega_j)\Delta\omega} e^{i\phi_j} \tag{5.9}$$

then, the sample function can be generated as:

$$\mathbf{X} = \mathrm{Re}\{\mathrm{FFT}(\mathbf{B})\} \tag{5.10}$$

where the notation $\mathrm{FFT}(\mathbf{B})$ is used to denote the FFT of the vector $\mathbf{B}$. Furthermore, the sample function $X_k = X(t_k)$ is now restricted to be discrete. A further discussion of implementation of the FFT for simulating sample functions, including MATLAB code, is included in Appendix C. As an example, in a test case run in MATLAB R2012b (8.0.0.783) on a 64-bit Windows 7 machine with an Intel(R) Core(TM) 2 Quad Q9550 @ 2.83GHz quad-core processor and 4GB RAM, the speed up was by a factor of about 400 (for $N = 1024$). Computation through a direct summation (i.e. Eqn. (5.3)) took, on average, 0.24 seconds. Computation using the FFT (i.e. Eqn. (5.10)), on the other hand, took, on average, 0.0006 seconds.

5.3 Non-stationary, univariate, Gaussian stochastic processes

5.3.1 Non-stationary processes

A process is said to be non-stationary if its statistical properties vary in time. There are, of course, varying degrees of non-stationarity. A process may be non-stationary in the mean, in which case the mean value is a function of time, i.e. $E[X(t)] = \mu(t)$. A process may be non-stationary in higher moments, as well, to an arbitrary degree. This means that the spectral characteristics of the process may vary in time, as well.

A plethora of practical examples of non-stationary processes exist, even when limited just to the field of civil engineering or applied mechanics. Arguably, non-stationary processes are more prominent in the physical world than stationary processes. Take, for example, seismic ground motions. Earthquake acceleration will always "ramp up" at the beginning, and die down at the end of their duration - thus making it by definition non-stationary in amplitude, at the very least. Though, often, their frequency content will differ along the duration as well. If soil liquefaction occurs, then the process becomes extremely non-stationary [82]. Another motivating example are material properties of Functionally Graded Materials (FGMs), where there exists a gradient in the material. As a naturally occurring example of FGMs, consider bamboo. In bamboo, the fiber-matrix density varies along the radius, giving the stalk more tensile strength along the perimeter - useful in resisting the bending moments induced by wind [83, 84]. Engineered FGMs are among the state-of-the-art in new material design, possessing the benefit of marrying the properties of two or more distinct materials in optimal configurations. The ability to simulate non-stationary processes and fields is of the utmost importance for designing resilient structures from the standpoint of either extreme loads or new materials.

Several different theories for modeling the evolving spectral characteristics exist, including the Instantaneous Power Spectrum [85] and the Wigner-Ville Spectrum [86–88]. In this thesis, however, we will work with Priestley's Evolutionary Spectrum (ES) [89]. Priestley's theory of non-stationary processes with evolutionary power has a particularly useful physical meaning because it preserves the concept of frequency, which is obscured in some other theories of non-stationary processes. Applications abound in civil engineering and engineering mechanics, as mentioned earlier, including actions on structures such as earthquakes, and characterization of materials with random microstructure such as some FGMs. The ES is characterized by a modulating function $A\left(t,\omega\right)$:

$$S^{E}\left(t,\omega\right)=\left|A\left(t,\omega\right)\right|^{2}S\left(\omega\right) \tag{5.11}$$

where $A\left(t,\omega\right)$ and $S\left(\omega\right)$ are the so-called modulating function and measure (or "stationary spectrum") respectively. Priestly established the following relationship between the Evolutionary Spectrum (ES) and the non-stationary ACF $R\left(t,\tau\right)$:

$$R\left(t,\tau\right)=\int\limits_{-\infty}^{+\infty}A\left(t,\omega\right)A^{*}\left(t+\tau,\omega\right)e^{i\omega\tau}\,\mathrm{d}S\left(\omega\right) \tag{5.12}$$

If $A\left(t,\omega\right)$ and $S\left(\omega\right)$ are known, it is therefore possible to determine the non-stationary autocorrelation function $R\left(t,\tau\right)$. However, the inverse is not possible as both the modulating function and the measure need to be determined from $R\left(t,\tau\right)$. The topic of computing this inverse transformation will be covered in great detail in Chapter 6.

There are two classes of non-stationary processes of this form that are important to consider:

1. Amplitude modulated processes, where $A\left(t,\omega\right)=A\left(t\right)$, thus:

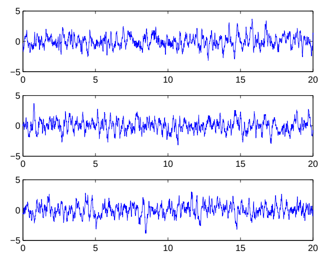
$$S^{E}\left(t,\omega\right)=\left|A\left(t\right)\right|^{2}S\left(\omega\right) \tag{5.13}$$

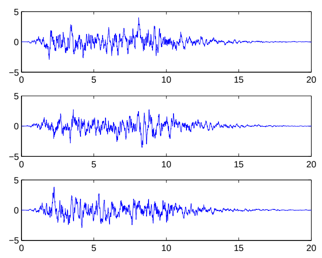
2. Amplitude and frequency modulated processes, of the form in Eqn. (5.11)

Figure 5.1 compares three sample realizations of a stationary process (Figure 5.1(a)), an amplitude modulated non-stationary process (Figure 5.1(b)), and a frequency and amplitude non-stationary process (Figure 5.1(c)).

The difference in frequency content in the latter should be evident, as well as the enveloped nature of the amplitude modulated.

Evolutionary spectra of the form in Eqn. (5.13) are separable - their time and frequency components can be separated into distinct and independent functions of one variable or the other. Spectra of the form in Eqn. (5.11), however, are not separable, and thus are said to be modulated in both amplitude and frequency. The former can be simulated with a more straightforward application of the stationary SRM, while the latter require an extension to the theory. Both cases are discussed below.


(a) Three realizations of a stationary process


(b) Three realizations of an amplitude modulated process

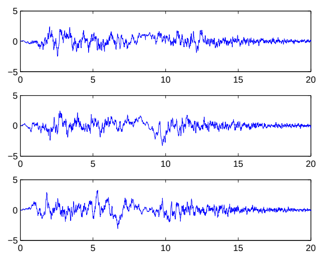

(c) Three realizations of a frequency and amplitude modulated process

Figure 5.1: Comparison of (a) stationary, (b) amplitude modulated, and (c) amplitude & frequency modulated processes

5.3.2 Simulation of amplitude modulated non-stationary processes

In the case of amplitude modulated (or sometimes referred to as "uniformly modulated") non-stationary processes, simulation is a straightforward extension of the stationary case. As alluded to earlier, Eqn. (5.13) is separable. Therefore, just as $A(t)$ can be thought of as an amplitude modulation on the underlying stationary spectrum, $S(\omega)$, it can also be thought of as an envelope on the process itself, $X(t)$. In other words, if there is a stationary process $Y(t)$ which corresponds to $S(\omega)$, then the non-stationary process $X(t)$ is:

$$X(t) = |A(t)|^2 Y(t) \tag{5.14}$$

where the sample realizations of the stationary process, $Y(t)$, are generated using either Eqn. (5.3) or Eqn. (5.10).

5.3.3 Simulation of amplitude and frequency modulated non-stationary processes

When the non-stationarity is due to combined amplitude and frequency modulation, then the spectral characteristics actually vary in time, and Eqn. (5.14) will be inaccurate. Though, even in this case, Liang et. al. [90] have shown that a direct extension of Eqn. (5.3) to include the ES, rather than the stationary SDF, is valid. Thus, sample realizations of a non-stationary process with amplitude and frequency modulation can be generated as:

$$X\left(t\right)=\sqrt{2}\sum_{j=0}^{N-1}\sqrt{2S^{E}\left(t,\omega_j\right)\Delta\omega}\cos\left(\omega_j t+\phi_j\right) \tag{5.15}$$

where, once again,

$$\Delta\omega=\frac{\omega_u}{N} \tag{5.16a}$$

$$\omega_j=j\Delta\omega \tag{5.16b}$$

and it is assumed that $S\left(t,\omega=0\right)=0$. Note, though, that in this case there is no direct application of the FFT. So, while Eqn. (5.15) is no more ‘complex’ than Eqn. (5.3) mathematically speaking, it is considerably more ‘complex’ from a computational or numerical standpoint. It should be noted, that although there is no direct application of the FFT in this form, Li and Kareem [91] have demonstrated an alternative formulation that makes use of the FFT, though this formulation only approximates the ES.

5.4 Non-Guassian stochastic processes

5.4.1 Stationary non-Gaussian stochastic processes

Due to the central limit theorem, samples generated in the SRM are asymptotically Gaussian [78]. The physical world, however, is in many cases non-Gaussian. Take, for example, any material property which is, by definition, non-negative. These properties are therefore inherently non-Gaussian, as the Gaussian support is the entire real line. Wind velocities are also a decidedly non-Gaussian physical process.

To that end, Grigoriu [92] defined the notion of a *translation process.* Essentially, a translation process is a non-linear mapping of an underlying Gaussian process to a non-Gaussian counterpart. For any Gaussian process, $Y\left(t\right)$, then the process

$$X\left(t\right)=g\left(Y\left(t\right)\right) \tag{5.17}$$

is not Gaussian unless the mapping $g\left(\cdot\right)$ is a linear function [92]. Grigoriu therefore showed that this mapping, $g\left(\cdot\right)$ can be chosen such that the process $X\left(t\right)$ matches the desired distribution. Specifically, for an arbitrary

Cumulative Distribution Function (CDF), F_{NG}, one can choose $g = F_{NG}^{-1} \circ F_G$, where $F_G(x)$ is the standard normal CDF. Then, the process

$$\begin{aligned} X(t) &= F_{NG}^{-1} \circ F_G[Y(t)] \\ &= F_{NG}^{-1}\{F_G[Y(t)]\} \end{aligned} \tag{5.18}$$

is a translation process, and $X(t)$ matches the desired distribution, $F_{NG}(x)$ [92]. Furthermore, Grigoriu [92] showed that the correlation structure of the two processes are related as:

$$R_{NG}(\tau) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F_{NG}^{-1}\{F_G[x_1]\} \cdot F_{NG}^{-1}\{F_G[x_2]\} \times \Phi\{x_1, x_2; \rho_G(\tau)\}\, \mathrm{d}x_1 \mathrm{d}x_2 \tag{5.19}$$

where $\Phi\{x_1, x_2, \rho_G(\tau)\}$ is the joint Gaussian probability density:

$$\Phi\{x_1, x_2, \rho_G(\tau)\} = \frac{1}{2\pi\sigma^2\sqrt{1-\rho_G^2(\tau)}} \times \exp\left(-\frac{x_1^2 + x_2^2 - 2\rho_G(\tau)\, x_1 x_2}{2\sigma^2\left(1-\rho_G^2(\tau)\right)}\right) \tag{5.20}$$

and $\rho(\tau)$ is the normalized Gaussian correlation:

$$\rho_G(\tau) = \frac{R_G(\tau)}{\sigma^2} \tag{5.21}$$

This transformation is always possible in a *forward* direction, i.e. from Gaussian to non-Gaussian. For an arbitrary pair of R_{NG} and F_{NG}, the reverse ($NG \to G$) is not always possible. When this reverse transformation is not possible, R_{NG} and F_{NG} are called "incompatible" according to translation theory. Thus, approximate techniques must be used to determine the underlying Gaussian process [93–97]. Shields et. al. [97] proposed a simple and efficient iterative technique making use of Eqn. (5.19) and the Wiener-Khinchin transform. This will be discussed in further detail in Chapter 7.

Once the underlying Gaussian process is determined, the procedure for simulating stationary non-Gaussian sample functions is straightforward. First, a Gaussian sample function is generated using either Eqn. (5.3) or Eqn. (5.10). This Gaussian sample function can then easily be mapped to the non-Gaussian distribution using Eqn. (5.18).

5.4.2 Non-stationary non-Gaussian stochastic processes

Grigoriu's theory for translation processes [92] was later applied to non-stationary, non-Gaussian processes by Ferrante et. al. [98]. The mapping takes a similar form to that in Eqn. (5.18), though, it must be generalized for CDFs that may vary in time:

$$X(t) = F_{NG}^{-1}\{F_G[Y(t), t], t\} \tag{5.22}$$

The relationship between the Gaussian and non-Gaussian ACFs (Eqn. (5.19)) was extended, as well, as:

$$R_{NG}(t,\tau) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F_{NG}^{-1}\{F_G[x_1, t], t\} \cdot F_{NG}^{-1}\{F_G[x_2, t+\tau], t+\tau\} \times \Phi\{x_1, x_2; \rho_G(t,\tau)\}\, \mathrm{d}x_1 \mathrm{d}x_2 \tag{5.23}$$

where Φ is of similar form as Eqn. (5.20), accounting for the parameterization in time, and $\rho_G(t,\tau)$ is of the form:

$$\rho_G(t,\tau) = \frac{R_G(t,\tau)}{\sigma(t)\,\sigma(t+\tau)} \tag{5.24}$$

Once again, the inverse transformation is not always possible analytically. However, in this case, methodologies for computing this inverse transformation are not available. Shields et. al. [99] proposed an approximate method, though this method is only accurate for weakly non-stationary processes. Chapter 7 will develop and discuss a robust and more accurate method for determining the underlying Gaussian process for an arbitrary pair of non-Gaussian ACF/ES and CDF.

Once the underlying Gaussian process is determined, simulation follows the same procedure as the stationary case. First, a sample realization is generated using Eqn. (5.15). This Gaussian sample function is then mapped to the non-Gaussian CDF through Eqn. (5.22).

5.5 Multi-variate stochastic processes

So far in this chapter, only scalar processes of one dimension were considered. In many cases, though, there will be some spatial variability as well as temporal. Consider, for example, a bridge subjected to seismic ground acceleration. If this bridge is supported by two towers, one can easily imagine that the acceleration at these two supports will vary. Of course, this variation can range from a simple lag to completely different shape and amplitude - this will be determined in large part by the distance between the supports. In order to model the variation between these two locations, there are two options [100]: (1) the ground motion acceleration "field" can be modeled as a vector process, with two components or (2) the ground motion can be modeled as a "wave," which is continuous in space and time. The former will be discussed in this section, and the latter will be discussed in Section 5.6. For this section, let us consider only stationary and Gaussian vector processes, for simplicity. Extensions to non-stationarity or non-Gaussianity follow similar methodologies outlined above for univariate processes. In the non-stationary case, a discussion can be found in [82].

With that in mind, let us define the notion of a *vector process.* A vector process is made up of individual, discrete, random processes:

$$\mathbf{X}(t) = \begin{bmatrix} X_1(t) \\ X_2(t) \\ \vdots \\ X_j(t) \\ \vdots \\ X_n(t) \end{bmatrix} \tag{5.25}$$

where $X_j(t)$ is component j of the n-variate vector process. Individually, each component is a univariate stochastic process, which can be considered in the framework of the earlier sections of this chapter. However, when considered as a whole, the correlation between different components is important. These correlations are modeled via the Cross Correlation Matrix (CCM) $R_{jk}(\tau)$ or the Cross Spectral Density Matrix (CSDM) $S_{jk}(\tau)$. The cross-correlations and cross-spectra form Fourier pairs, just as in Eqn. (5.2):

$$S_{jk}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} R_{jk}(\tau) e^{-i\omega\tau} \, \mathrm{d}\tau \tag{5.26a}$$

$$R_{jk}(\tau) = \int_{-\infty}^{\infty} S_{jk}(\omega) e^{i\omega\tau} \, \mathrm{d}\omega \tag{5.26b}$$

In the homogeneous case (i.e. 'stationary' in space), the CSDM is typically constructed of prescribed auto-spectra, $S_{jj}(\omega)$, and coherence model, $\gamma(\omega, \xi)$ [100]:

$$S_{jk}(\omega) = \sqrt{S_{jj}(\omega) S_{kk}(\omega)} \cdot \gamma(\omega, \xi) \tag{5.27}$$

where ξ is the separation distance between nodes j and k. The auto-spectrum, S_{jj}, and the coherence function, γ, are chosen for the problem at hand. There are a multitude of options in the literature for most practical problems. For example, for wind velocities, the auto-spectrum is typically chosen to be the Kaimal Spectrum [101] and the coherence model is typically chosen to be the Davenport coherence function [102]. For seismic ground motions, the auto-spectrum is typically the Kanai-Tajimi [103, 104] or Clough-Penzion [105] spectra. For seismic coherency models, some typical choices include the Harichandran and Vanmarcke model [106], or the Luco and Wong model [107].

With the CSDM known — either from prescribed properties or estimated from data — the SRM can once again be used to simulate sample realizations of the vector process. First, the CSDM is decomposed as:

$$\mathbf{S}(\omega) = \mathbf{H}(\omega) \mathbf{H}^{T*}(\omega) \tag{5.28}$$

where the superscript ^{T} denotes the matrix transpose, and * denotes complex conjugate. This can be thought of as a typical Cholesky decomposition, and thus $\mathbf{H}$ is lower triangular:

$$\mathbf{H}(\omega) = \begin{bmatrix} H_{11}(\omega) & 0 & 0 & \cdots & 0 \\ H_{21}(\omega) & H_{22}(\omega) & 0 & \cdots & 0 \\ H_{31}(\omega) & H_{32}(\omega) & H_{33}(\omega) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ H_{n1}(\omega) & H_{n2}(\omega) & H_{n3}(\omega) & \cdots & H_{nn}(\omega) \end{bmatrix} \tag{5.29}$$

See [63, 82, 108] for discussion of the properties and restrictions on this matrix.

From the decomposition of Eqns. (5.28) and (5.29), sample realizations can be generated from the following formula, as $N \to \infty$ [63, 82, 108]:

$$X_j\left(t\right) = 2\sum_{m=1}^{j}\sum_{l=0}^{N-1} \left|H_{jm}\left(\omega_l\right)\right| \sqrt{\Delta\omega} \cos\left(\omega_l t + \Theta_{jm}\left(\omega_l\right) + \phi_{ml}\right); \;\; j = 1, 2, \ldots, n \tag{5.30}$$

where:

$$\omega_l = l\Delta\omega, \; l = 0, 1, \ldots, N-1 \tag{5.31a}$$

$$\Delta\omega = \frac{\omega_u}{N} \tag{5.31b}$$

$$\Theta_{jm}\left(\omega_l\right) = \tan^{-1}\left(\frac{\mathrm{Im}\left[H_{jm}\left(\omega_l\right)\right]}{\mathrm{Re}\left[H_{jm}\left(\omega_l\right)\right]}\right) \tag{5.31c}$$

where $\mathrm{Im}\left[\cdot\right]$ and $\mathrm{Re}\left[\cdot\right]$ denote the imaginary and real parts, respectively; and ϕ_{ml}, once again, represents independent random phase angles between $[0, 2\pi]$. In this case, though, ϕ_{ml} is a $n \times N$ matrix. Conversely, ϕ_{ml} can be thought of as n independent series of length N. In the form shown in Eqn. (5.30), the generated samples are not ergodic, in contrast to the univariate samples. An alternative form shown in [82, 109] double-indexes the frequencies in order to overcome this limitation. As in the univariate, stationary case, the FFT can again be used to speed up computation of Eqn. (5.30).

5.6 Multi-dimensional stochastic processes

In the previous section, spatial variability was modeled via a vector process. Alternatively, in this section multi-dimensional stochastic processes will be discussed. Although mathematically similar, the discussion will be broken into two parts: (1) multi-dimensional fields and (2) stochastic waves. The difference being, (1) is a function of two or more space dimensions and (2) is a function of time and space. For simplicity, all processes in this section will be assumed stationary, Gaussian, and univariate. Furthermore, they will be restricted to only two dimensions, when necessary, with no loss of generality — this is only done for ease of notation.

5.6.1 Multi-dimensional random fields

In the case of a multi-dimensional random field, there is a direct extension from Eqn. (5.3) [63]. Let us consider a two-dimensional (2D), stationary, uni-variate, Gaussian process, $X\left(\mathbf{x}\right) = X\left(x_1, x_2\right)$, where

(x_1, x_2) are Cartesian coordinates in $\mathbb{R}^2$. The ACF and SDF are now also 2D:

$$S(\boldsymbol{\kappa}) = \frac{1}{(2\pi)^2} \int\limits_{-\infty}^{\infty} R(\boldsymbol{\xi})\, e^{-i\boldsymbol{\kappa}\cdot\boldsymbol{\xi}}\, \mathrm{d}\boldsymbol{\xi} \tag{5.32a}$$

$$R(\boldsymbol{\xi}) = \int\limits_{-\infty}^{\infty} S(\boldsymbol{\kappa})\, e^{i\boldsymbol{\kappa}\cdot\boldsymbol{\xi}}\, \mathrm{d}\boldsymbol{\kappa} \tag{5.32b}$$

where $\boldsymbol{\kappa} = [\kappa_1, \kappa_2]$ is the wave-number vector, $\boldsymbol{\xi} = [\xi_1, \xi_2]$ is the separation vector, and $\boldsymbol{\kappa}\cdot\boldsymbol{\xi}$ denotes their inner product. Thus, in actuality, both integrals in Eqn. (5.32) are double integrals in this case, or m-fold integrals for m-dimensional cases. The $(2\pi)^2$ in Eqn. (5.32a) would be generalized to $(2\pi)^m$ for an m-dimensional process. Using the SRM, once again, a sample realization can be generated with the following formula [110]:

$$X(\mathbf{x}) = \sqrt{2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \sum_{I_2=\pm1} \sqrt{2S(\kappa_{1k_1}, I_2\kappa_{2k_2})\,\Delta\kappa_1\Delta\kappa_2} \cdot \cos\left(\kappa_{1k_1}x_1 + I_2\kappa_{2k_2}x_2 + \phi_{k_1k_2}^{I_2}\right) \tag{5.33}$$

where $\phi_{k_1k_2}^{I_2}$ are again random phase angles uniformly distributed between 0 and 2π, and:

$$\kappa_{jk_j} = j\Delta\kappa_j, \quad k_j = 1, 2, \ldots, N_j, \quad j = 1, 2 \tag{5.34a}$$

$$\Delta\kappa_j = \frac{\kappa_{uj}}{N_j} \tag{5.34b}$$

Once again, the FFT can be used to speed up computation of Eqn. (5.33).

5.6.2 Stochastic waves

In contrast to the multi-dimensional fields in Section 5.6.1, we will now look at stochastic waves. A *stochastic wave* is a random process in two or more dimensions, $X(t, \mathbf{x})$, where one dimension is temporal and one or more are spatial. This is particularly useful for modeling random processes in time which have spatial variability – an alternative to the multi-variate formulation discussed earlier, in many cases. As an example, seismic ground motion can be modeled as a stochastic wave [80, 100, 111, 112]. In situations where many points in space are necessary, it would be advantageous to use a continuous representation rather than a discrete representation, as in the multi-variate formulation. In Chapter 8 we will show that in some cases it is actually impossible using the multi-variate form. For simplicity, in this section we will again assume stationarity, Gaussianity, and that the process is univariate and of only one spatial dimension.

A stochastic wave is characterized by the Frequency-wavenumber (F-K) spectrum, $S(\omega, \kappa)$. From the F-K spectrum, the SRM can be used to simulate sample waves [111, 112]:

$$X(x, t) = \sqrt{2} \sum_{l=1}^{N_\kappa} \sum_{m=1}^{N_\omega} \sum_{I_\omega=\pm1} \sqrt{2 \cdot S_f(I_\omega\omega_m, \kappa_l) \cdot \Delta\omega\Delta\kappa} \cdot \cos\left[I_\omega\omega_m t + \kappa_l x + \phi_{ml}^{I_\omega}\right] \tag{5.35}$$

where

$$\omega_m = m\Delta\omega \tag{5.36a}$$

$$\Delta\omega = \frac{\omega_u}{N_\omega} \tag{5.36b}$$

$$\kappa_l = l\Delta\kappa \tag{5.36c}$$

$$\Delta\kappa = \frac{\kappa_u}{N_\kappa} \tag{5.36d}$$

The FFT can once again be used to speed up computation of Eqn. 5.35. In fact, the effects are compounded for this multi-dimensional case, compared to the univariate, one-dimensional, stationary case described earlier. A further discussion of how to implement the FFT for this case, and comparisons of computation times, is carried out in Appendix C.

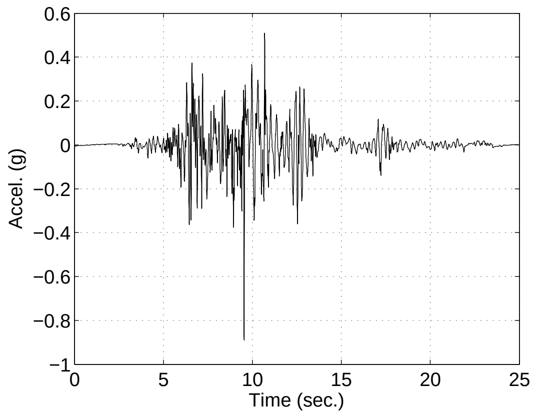
Chapter 6

Determining Evolutionary Spectra from non-stationary Autocorrelation Functions

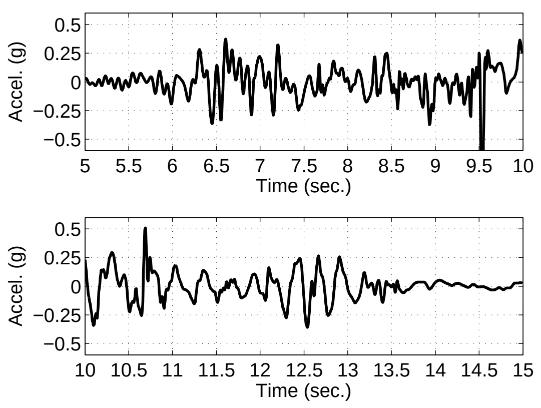
6.1 Introduction & motivation

As was briefly introduced in Chapter 5, there are a multitude of practical examples of non-stationary processes, even when limited just to the fields of civil engineering or applied mechanics. Specifically, earthquakes and functionally graded materials (FGMs) were cited as examples. Earthquakes exhibit both amplitude and frequency modulation, as can be seen in Figure 6.1. This figure shows actual recorded acceleration time history data for Station 16 LGPC, Record # NGA0779 on the PEER Ground Motion Database [113] - the event shown is the October 18th, 1989 Loma Prieta earthquake. Figure 6.1(a) shows the entire history, and Figure 6.1(b) shows detail zooms for two separate 5 second windows. From Figure 6.1(a), a clear amplitude modulation can be seen — the amplitude of the oscillations follows a fairly clear envelope, ramping up between about 3 seconds and 6 seconds, and then dies down between about 12 seconds and 20 seconds. There is also a clear change in frequency content over the course of the earthquake, though this is perhaps more easily seen in the zoom-in detail of Figure 6.1(b). This figure shows two windows of the acceleration time history of Figure 6.1(a), the top window is from 5 seconds through 10 seconds, and the bottom window is from 10 seconds to 15 seconds. It can be seen, qualitatively, that the more dominant frequencies in the two windows are significantly different — the top has a higher frequency than the bottom.

The other example of non-stationarity cited earlier was Functionally Graded Materials (FGMs). A FGM



(a) Ground acceleration



(b) Zoom detail on ground accel.

Figure 6.1: Example seismic ground acceleration: 1989 Loma Prieta earthquake. Data source: Pacific Earthquake Engineering Research Center (PEER) [113], Record #: NGA0779, ATH: LOMAP/LGP-UP.

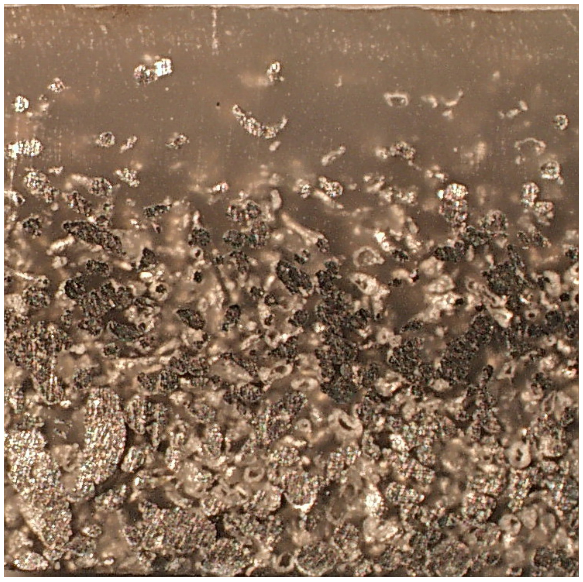


Figure 6.2: Sample Functionally Graded Material (FGM): aluminum particulates in a high density polyethylene matrix (AL-HDPE). Image courtesy of Po-Hua Lee, Columbia University, 2013

is a composite material in which a gradient exists in the properties. Bamboo was discussed as a naturally occurring FGM in Chapter 5, though FGMs are often synthetic and engineered, as well. For example, Figure 6.2 shows a bi-material FGM, where the particles are aluminum (Al) and the matrix is high density polyethylene (HDPE). The volume fraction of Al clearly varies from the bottom to the top — where there is a high density at the bottom and low density at the top. Therefore, the composite will exhibit more of the Al properties towards the bottom of the material, and more of the HDPE characteristics towards the top. Such forms are advantageous where a design calls for marrying mechanical strength, thermal properties, ductility, or other properties of multiple materials. Rather than discretely layering multiple materials, FGMs provide a more efficient and optimum design in many situations. Modeling and simulating a FGM, such as Al-HDPE or bamboo, would require accounting for this non-stationarity/inhomogeneity.

Naturally, modeling and simulating non-stationary processes requires special treatment. Standard stationary process theory will fall short – e.g. Eqns. (5.2) and (5.3). Within the framework of the SRM, it is modeling the evolving spectral characteristics that is of the highest importance, and biggest challenge. Several different theories for modeling the evolving spectral characteristics exist, including the Instantaneous Power Spectrum [85] and the Wigner-Ville Spectrum [86–88]. In this thesis, however, we will work with Priestley's Evolutionary Spectrum (ES) [89]. Priestley's theory of non-stationary processes with evolutionary power has a particularly useful physical meaning because it preserves the concept of frequency, which is obscured in some other theories of non-stationary processes. The ES was introduced and discussed in Section 5.3, already, specifically in Eqns. (5.11) through (5.16b).

Though, as primary motivation for this chapter, Eqn. (5.12) will be reproduced here. Priestley [89] established the following relationship between the ES and the non-stationary ACF, $R(t,\tau)$:

$$R(t,\tau) = \int_{-\infty}^{+\infty} A(t,\omega)\, A^*(t+\tau,\omega)\, e^{i\omega\tau}\, \mathrm{d}S(\omega) \tag{6.1}$$

This transformation is definitely *non-reversible* — i.e. given an ACF, a unique ES cannot be found analytically. In fact, this has been proven mathematically [89, 114]. Often, a simplifying assumption of $S(\omega) = 1$ is used. Still, in this case it is not clear whether the transform becomes reversible in a unique way.

Recall from Eqn. (5.15) that the ES is necessary for simulation using the SRM. However, sometimes only ACF is available. More importantly, transformation between ACF and ES is necessary to determine the ES of non-Gaussian, non-stationary processes with evolutionary power. This was briefly introduced in Section 5.4, and will be discussed in great detail further in Chapter 7.

To this end, we are primarily concerned with the following two questions:

1. Is there a solution to the inverse transformation: $R(t,\tau) \to S(t,\omega)$?
2. If a solution exists, is it unique?

With that in mind, this chapter is organized as follows. Section 6.2 will first formulate the inversion as a discrete optimization problem. Section 6.3 will examine the existence and uniqueness of a solution to this inversion. Sections 6.4 and 6.5 will then move on to describe more efficient methodologies for computing this inverse. Numerical examples will be examined throughout.

6.2 Formulation of discrete optimization problem

In the most general case, Priestley [89] derived the relationship between the ACF of a process and its spectral representation as shown in Eqn. (6.1). In this general case, it has been shown that no unique inverse exists [89, 114]. We, however, are interested in the particular case where:

1. $S(\omega)$ exists such that $\mathrm{d}S(\omega) = S(\omega)\,\mathrm{d}\omega$
2. $A(t,\omega)$ is a real and non-negative function of both t and ω
3. $S(\omega)$ is assumed to be white noise for all ω such that the evolutionary spectrum reduces to: $S^E(t,\omega) = A^2(t,\omega)$

Given assumptions #1 and #3, we can simplify the integral in Eqn. (6.1) to:

$$R(t,\tau) = \int_{-\infty}^{+\infty} A(t,\omega)\,A(t+\tau,\omega)\,e^{i\omega\tau}\,\mathrm{d}\omega \tag{6.2}$$

Note, also, that because the modulating function $A(t,\omega)$ and the ES $S^E(t,\omega)$ completely define each other under these assumptions (specifically the third assumption), the two will be used interchangeably throughout the rest of the chapter.

Let us start by formulating the inversion as a discrete optimization problem. It is assumed that a target ACF is given (or can easily be estimated), $R^{target}(t,\tau)$. The ACF, in general, is discretized as:

$$R_{jk} = R(t_j,\tau_k) = R(j\Delta t, k\Delta\tau) \tag{6.3}$$

Typically, $\Delta t = \Delta\tau$. Additionally, we must define some measure of the error, ε:

$$\varepsilon = \frac{\left\| R_{jk}^{target} - R_{jk}^{computed} \right\|}{\left\| R_{jk}^{target} \right\|} \times 100\% \tag{6.4}$$

where $R_{jk}^{computed}$ is the estimated autocorrelation function. For all of the methods described in this chapter, the following norm is adopted:

$$\|X_{jk}\| = \sqrt{\sum_{j=1}^{N_t} \sum_{k=1}^{N_\tau} X_{jk}^2} \tag{6.5}$$

where

$$N_t = \frac{t_{max}}{\Delta t} \tag{6.6a}$$

$$N_\tau = \frac{\tau_{max} - \tau_{min}}{\Delta \tau} \tag{6.6b}$$

The optimization problem can be stated as the following. We would like to find some $A(t, \omega)$ that, when integrated as in Eqn. (6.2), approximates as closely as possible the target ACF, $R^{target}(t, \tau)$. For computational purposes, we discretize $A(t, \omega)$ as:

$$A_{lm} = A(t_l, \omega_m) = A(l\Delta t, m\Delta\omega) \tag{6.7}$$

Eqn. (6.2) can then be numerically integrated to find $R_{jk}^{computed}$. For all methods developed in this chapter, numerical integration is performed using Simpson's rule. We are interested in finding A_{lm} which minimizes the error in the ACF (Eqn. 6.4).

6.3 "Brute Force" Method (BFM)

6.3.1 Description of algorithm

The primary intentions for the Brute Force Method (BFM) described in this section are threefold:

1. to determine whether or not the inversion process converges;
2. if it does converge, whether or not it converges to a unique solution; and
3. whether this unique solution is the correct one.

To that end, a numerical "Brute Force" approach was proposed in the appendix of Shields' Ph.D. thesis [114]. Though the convergence of this method does not prove uniqueness, it is a strong indicator of such. In addition, because the convergence path is stochastic, as will be shown shortly, repeated convergence to the same ES would imply uniqueness (though not a rigorous proof, of course). The errors are measured in the ACF (Eqn. (6.4)) and as such, the method does not depend on the ES, only the prescribed ACF.

The BFM is illustrated in Figure 6.3. In brief, the algorithm works as follows. We begin with a prescribed ACF. Note, though, that for tests in this chapter we start from a prescribed ES which we integrate (Eqn. (6.2)) to find the target ACF. This target ES is then 'forgotten.' An initial guess is then chosen for the ES, say, some flat spectrum of a positive number (e.g. $A(t, \omega) = 50$). This ES is integrated to find the associated ACF, and the error between the computed ACF and the target ACF is calculated. Then, a point is chosen at random, and it is shifted. If this shift results in less error than previously, the shift is accepted, otherwise it is discarded. This process is repeated until convergence. A progression of these iterations are

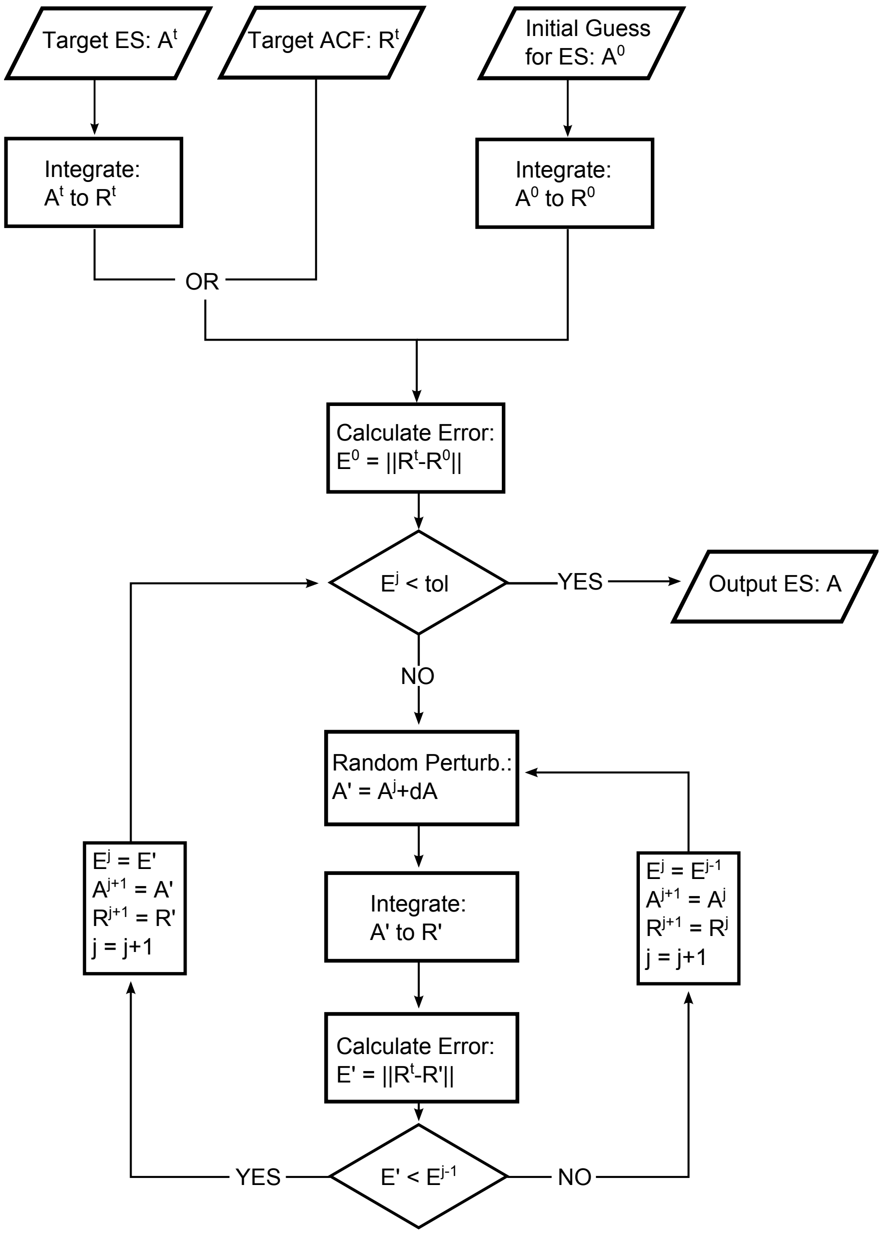
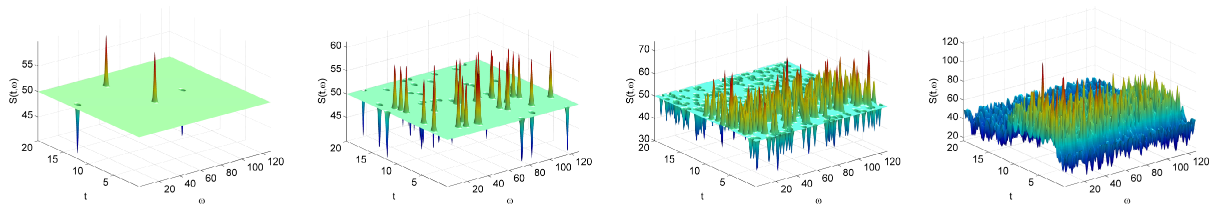

Target ES: A^t
Target ACF: R^t
Initial Guess for ES: A^0
Integrate: A^t to R^t
Integrate: A^0 to R^0
OR
Calculate Error: $E^0 = ||R^t-R^0||$
$E^j <$ tol
YES
Output ES: A
NO
Random Perturb.: $A' = A^j+dA$
Integrate: A' to R'
Calculate Error: $E' = ||R^t-R'||$
$E' < E^{j-1}$
YES
NO
$E^j = E'$ $A^{j+1} = A'$ $R^{j+1} = R'$ $j = j+1$
$E^j = E^{j-1}$ $A^{j+1} = A^j$ $R^{j+1} = R^j$ $j = j+1$

Figure 6.3: Brute Force Method (BFM) algorithm



(a) 10 iterations (b) 100 iterations (c) 1000 iterations (d) 10000 iterations

Figure 6.4: Progression of the BFM over the first 10,000 iterations

shown in Figure 6.4 after 10, 100, 1000, and 10,000 iterations. It can be seen how power begins to build up through the random perturbations, though it is also clearly very slow convergence. The spectrum at any given iteration is quite "bumpy," due to the random perturbations. To preserve some continuity and smoothness, the perturbations used were essentially Gaussian bell curves:

$$\Delta A\left(t,\omega\right) = C\exp\left(\frac{\left(t-t'\right)^2}{\alpha_t}\right)\cdot\exp\left(\frac{\left(\omega-\omega'\right)^2}{\alpha_\omega}\right) \tag{6.8}$$

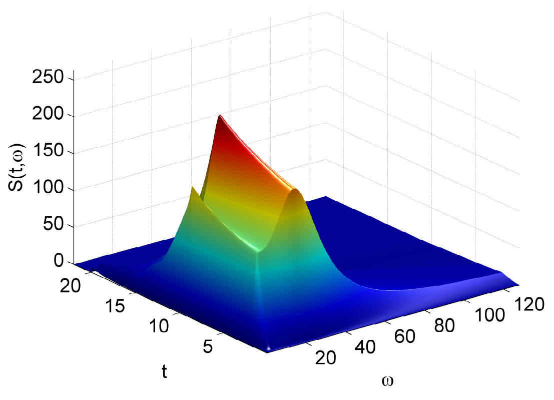
where t' and ω' are random time and frequency coordinates, uniformly distributed within the domain, and C, α_t and α_ω are scaling parameters. The magnitude of the perturbations, C, was chosen to be a fraction of the magnitude of A, and decreased in magnitude as the iterations progressed. The width-control parameters, $\alpha_{t,\omega}$, contracted as the iterations progressed to allow for more localized adjustments. For more discussion of this method see [114].

6.3.2 BFM results

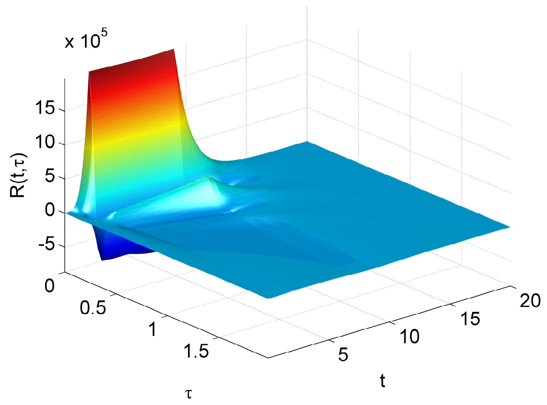
Let us now examine the results from this method by estimating the ES from the ACF for a known example ES. This example is the same as [114]: a Clough-Penzien acceleration spectrum [105] with both frequency and amplitude modulation similar to that used in [82]. The Clough-Penzien acceleration spectrum is used to describe non-stationary seismic ground motion. This spectrum is shown in Figure 6.5(a), and is defined as:

$$S_{CP}^{E}\left(\omega,t\right) = A_{JHT}^{2}\left(t\right)S_0\left(t\right)\underbrace{\left[\frac{1+4\zeta_g^2\left(\frac{\omega}{\omega_g}\right)^2}{\left[1-\left(\frac{\omega}{\omega_g}\right)^2\right]^2+4\zeta_g^2\left(\frac{\omega}{\omega_g}\right)^2}\right]}_{\text{Non-stationary Kanai-Tajimi spectrum}}\times\underbrace{\left[\frac{\left(\frac{\omega}{\omega_f}\right)^4}{\left[1-\left(\frac{\omega}{\omega_f}\right)^2\right]^2+4\zeta_f^2\left(\frac{\omega}{\omega_f}\right)^2}\right]}_{\text{Clough-Penzien correction}} \tag{6.9}$$

where the amplitude modulation is given through the Jennings, Housner, and Tsai modulating function



(a) ES



(b) ACF

Figure 6.5: Clough-Penzien (CP) spectrum with amplitude and frequency modulation

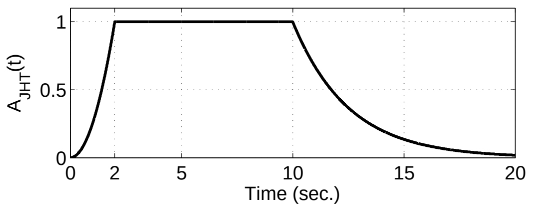


Figure 6.6: Jennings, Housner, and Tsai modulating function [115]

[115]:

$$A_{JHT}(t) = \begin{cases} \left(\frac{t}{t_1}\right)^2 & 0 \leq t \leq t_1 \\ 1 & t_1 \leq t \leq t_2 \\ \exp\left[-c\left(t - t_2\right)\right] & t_2 \leq t \end{cases} \tag{6.10}$$

which is shown in Figure 6.6. The parameter $S_0(t)$ in Eqn. (6.9) is used to describe the intensity of the acceleration in the underlying Kanai-Tajimi spectrum [103, 104]. It is defined as:

$$S_0(t) = \frac{\sigma^2}{\pi\omega_g\left(\frac{1}{2\zeta_g} + 2\zeta_g\right)} \tag{6.11}$$

where σ is the standard deviation, and ω_g and ζ_g in Eqns. (6.9) and (6.11) are the characteristic frequency and damping, respectively, of the ground, which are soil properties. Ellingwood and Batts [116] have suggested values for rock, deep cohesionless soils, and clays and sands, respectively. Finally, ω_f and ζ_f in Eqn. (6.9) are the filtering parameters of the Clough-Penzien spectrum [105], which are typically taken to be:

$$\omega_f = 0.1\omega_g \tag{6.12a}$$

$$\zeta_f = \zeta_g \tag{6.12b}$$

The parameter definitions used in this numerical example are as follows:

$$\omega_g = 30 - 1.25t \tag{6.13a}$$

$$\zeta_g = 0.5 + 0.005t \tag{6.13b}$$

$$\sigma = 100 \tag{6.13c}$$

$$t_1 = 2 \tag{6.13d}$$

$$t_2 = 10 \tag{6.13e}$$

$$c = 0.4 \tag{6.13f}$$

For reference, sample time histories generated from this spectrum (scaled for $\sigma = 1$) were used in the previous chapter, in Figure 5.1(c). The initial guess was a flat spectrum, i.e. $A(t, \omega) = 50 \; \forall (t, \omega)$. Time and frequency were discretized as:

$$t \in (0, 20) \tag{6.14a}$$

$$\tau \in (0, 2) \tag{6.14b}$$

$$\Delta t = 0.025 \tag{6.14c}$$

$$\Delta\tau = 0.025 \tag{6.14d}$$

$$\omega \in (0, 128) \tag{6.14e}$$

$$\Delta\omega = 0.5 \tag{6.14f}$$

The results for the BFM after 1,000,000 iterations are shown in Figure 6.7. The final computed ES is shown in Figure 6.7(a). A "slice" (at constant t) comparing the computed ES with the target ES is shown in Figure 6.7(b). The associated computed ACF is shown in Figure 6.7(c). This has an error of 1.57% when measured according to Eqn. (6.4). Measured in the ES, the error is 6.95%. Visually, both the ACF and ES haven taken at least the general shape of their targets (Figure 6.5). The ACF is noticeably smoother than the ES, and matches the target quite nicely – as evident quantitatively through the error measure, as well. The ES, on the other hand, although quite close the the general shape of the target, is quite 'bumpy'. This is of course due to the nature of the random perturbations used to compute it. Figure 6.7(b) shows that although bumpy/noisy, the general shape is in fact captured. One million iterations is certainly slow (one iteration takes approximately two seconds, depending on programming language, hardware, and parallelization). We can clearly observe that the ES converges slowly towards the target, which is a strong indication that a unique solution exists.

Returning to the primary intentions listed at the beginning of this section, these results show:

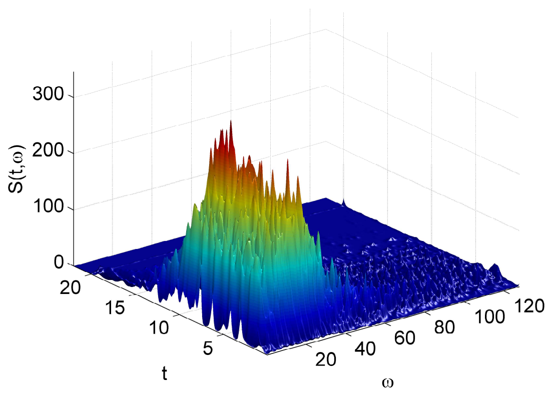
1. clear convergence to a specific ES,

2. convergence to a unique ES,

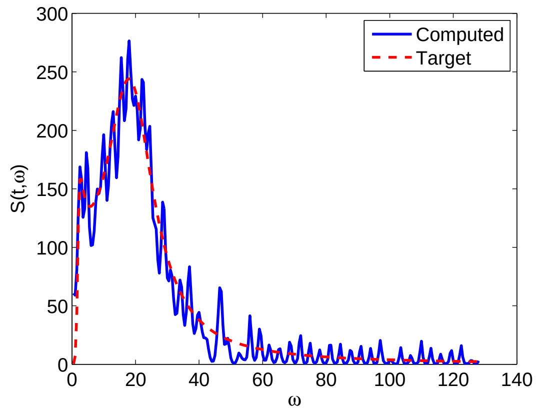
3. and furthermore, convergence to the correct shape.

Specifically, through convergence to a clearly defined shape, and not an arbitrary shape (i.e. complete noise), the results point to the likelihood that a solution exists. The repeated convergence to similar solutions (including similar convergence shown in [114]) then points to the uniqueness of this solution. Note that the convergence path is stochastic, and there is no bias in the algorithm. Thus, if the same minimum is found, and is the only minimum found, it is very likely that this is the only minimum that exists. Finally, recall that this test case was started from a known ES. The clear convergence to the general shape of this target ES points to the likelihood of both existence and uniqueness of the solution to the inversion. To reiterate, although the test case was started from a known ES, this was integrated to the target ACF and then "forgotten." All comparisons in the algorithm are *always* on the ACF, and thus no direct information from the ES makes it way into the convergence path.

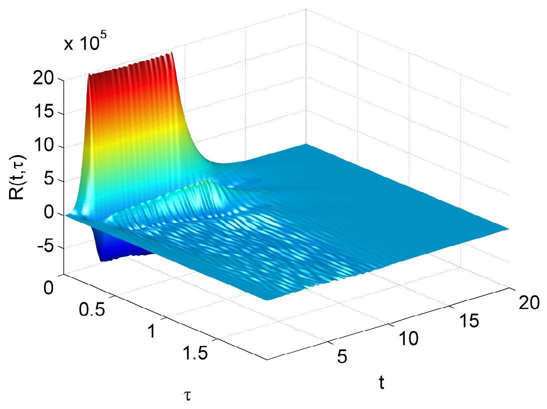
Finally, Figure 6.8 shows sample realizations generated using both the target and estimated spectra. The two samples were generated using the same random phase angles, so as to be able to compare directly. Figure 6.8(a) shows the entire 20 second duration, and Figure 6.8(b) shows two 5 second window zoom-ins. It is clear that the sample generated from the estimated spectrum matches the target quite well. The noise in the estimated spectrum appears to have minimal effect on the generated samples. This points to the conclusion that the general shape of the spectrum is more important than exact replication.



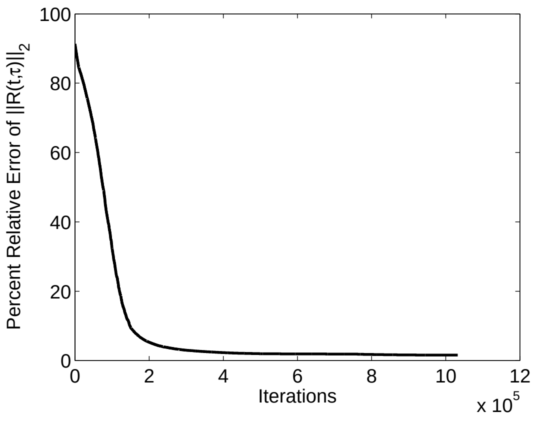
(a) BFM computed ES



(b) BFM sample ES "Slice"

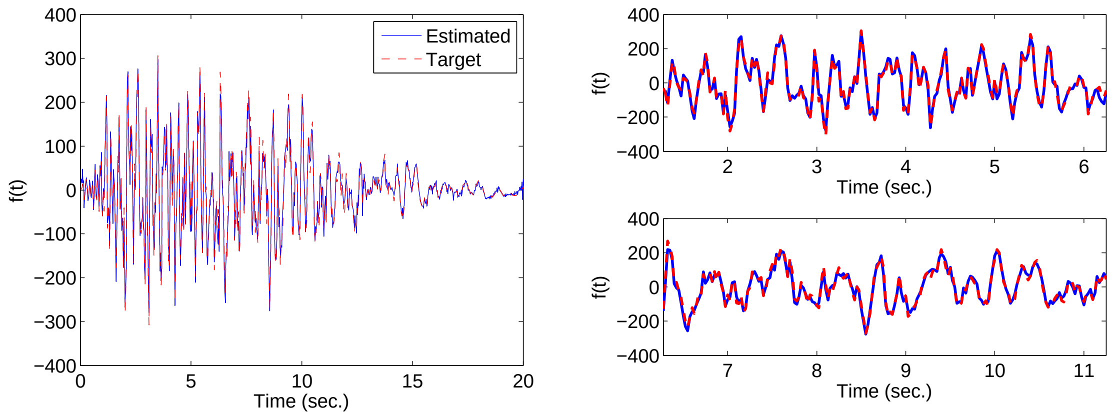


(c) BFM computed ACF



(d) BFM errors vs. iteration

Figure 6.7: BFM convergence results for Clough-Penzien (CP) spectrum with amplitude and frequency modulation. After 1,000,000 iterations of BFM, final error: 6.95% in ES and 1.57% in ACF



(a) Simulated sample realization comparison (b) Zoom in on sample realizations

Figure 6.8: Comparison of sample realizations generated using SRM with target ES (red dash) and estimated ES (solid blue)

6.4 "Multi-Grid" Method (MGM)

6.4.1 Description of algorithm

In the previous section it was shown, though not proved, that a unique solution to the inverse of Eqn. (6.2) likely exists. The BFM described therein was certainly not efficient enough for any practical applications: the computation to sufficient accuracy would take days or weeks. To that end, the remainder of this chapter looks at making this computation more efficient. This section will develop and examine a proposed method, and carry out numerical examples of such. The proposed method will then be compared against existing off-the-shelf optimization techniques. The following section will then propose an efficient initial guess, which in some cases may even be a good enough approximation of the ES. This will be shown through numerical examples, as well.

With that, we will now move on to introduce the "Multi-Grid" Method (MGM). In order to increase the efficiency in convergence, we look to reduce the parameter space of the problem. The original problem for the BFM was on a discrete space of 800×256 'parameters' to be estimated. Therefore, the parameter space was on the order of 200,000 degrees of freedom. With uniformly random sampling for the perturbations, this would require a minimum of 400,000 iterations, on average, for each location to be 'touched' *just once*. Often the domain will be discretized even further. If the original domain is 1024×1024, then the parameter space is over 1,000,000 degrees of freedom. So it is quite clear that the problem grows in complexity quickly. However, by using interpolation and assuming some level of continuity, this space can be significantly reduced. Furthermore, inspired by the multi-grid solvers used in linear systems, it will be demonstrated that it is possible to use the information on the coarse scale grid to improve convergence on the fine scale grid.

The MGM algorithm is shown in Figure 6.9. The MGM algorithm is very similar to the BFM described previously, with the addition of a mesh refinement step. The mesh begins coarse (e.g., for the examples in Section 6.4.2 a 20×10 mesh was used) and gradually increases in density to the finest scale (800×256) over the course of the iterations. When the mesh is transfered from coarse to fine, bi-cubic spline interpolation is used. This assumes a certain degree of continuity, and smoothness, as the bi-cubic splines preserve C^2-continuity. However, splines do not preserve non-negativeness, which is one of the initial assumptions. In order to remedy this, local linear interpolation is used if any negative values are produced by the spline interpolation. Furthermore, although the random perturbations are carried out on the coarse mesh, the errors are always calculated on the fine scale. Otherwise, the norm defined in Eqn. (6.5) would not be consistent between different meshes.

Figure 6.10 shows the progression of the MGM over the first 1,000 iterations. In contrast to the BFM of Figure 6.4, improvements are immediately recognizable. First, because the perturbations are on the coarse scale, each perturbation affects a wider set of points in the (fine scale) domain. Early on in the iterations this is very useful to find the general shape faster, and then fine tune with smaller perturbations afterwards. As such, power begins to build up in the correct areas much sooner in the MGM than the BFM, and by 1,000 iterations the general shape is starting to form – for the BFM this was taking 10,000 or more iterations. Second, the results are much smoother than in the BFM. The bumpy and noisy nature of the BFM results are absent, and in their stead are very smooth curves. Priestley's theory [89] assumes a certain degree of smoothness, so this is an acceptable assumption for the MGM. However, if there were steep gradients that needed to be captured, this could be captured after the initial "broad" iterations, by further iterations on the fine scale.

6.4.2 MGM results

In this section, two numerical examples are carried out for the MGM. First, the CP example from the BFM is revisited. Then, a completely arbitrary and non-physical spectrum is used, to demonstrate the robustness of the algorithm. In the first case, the convergence is compared with some off-the-shelf established optimization techniques.

Clough-Penzien example

In the first numerical example, the CP spectrum from Section 6.3.2 is revisited. The parameters and initial guess are identical to that example. The resulting ES is shown in Figure 6.11(a), and a sample slice is shown in Figure 6.11(b). The associated ACF is shown in Figure 6.11(c). The convergence behavior is compared to the BFM in Figure 6.11(d). After four hundred thousand iterations, the error measured in the ACF is

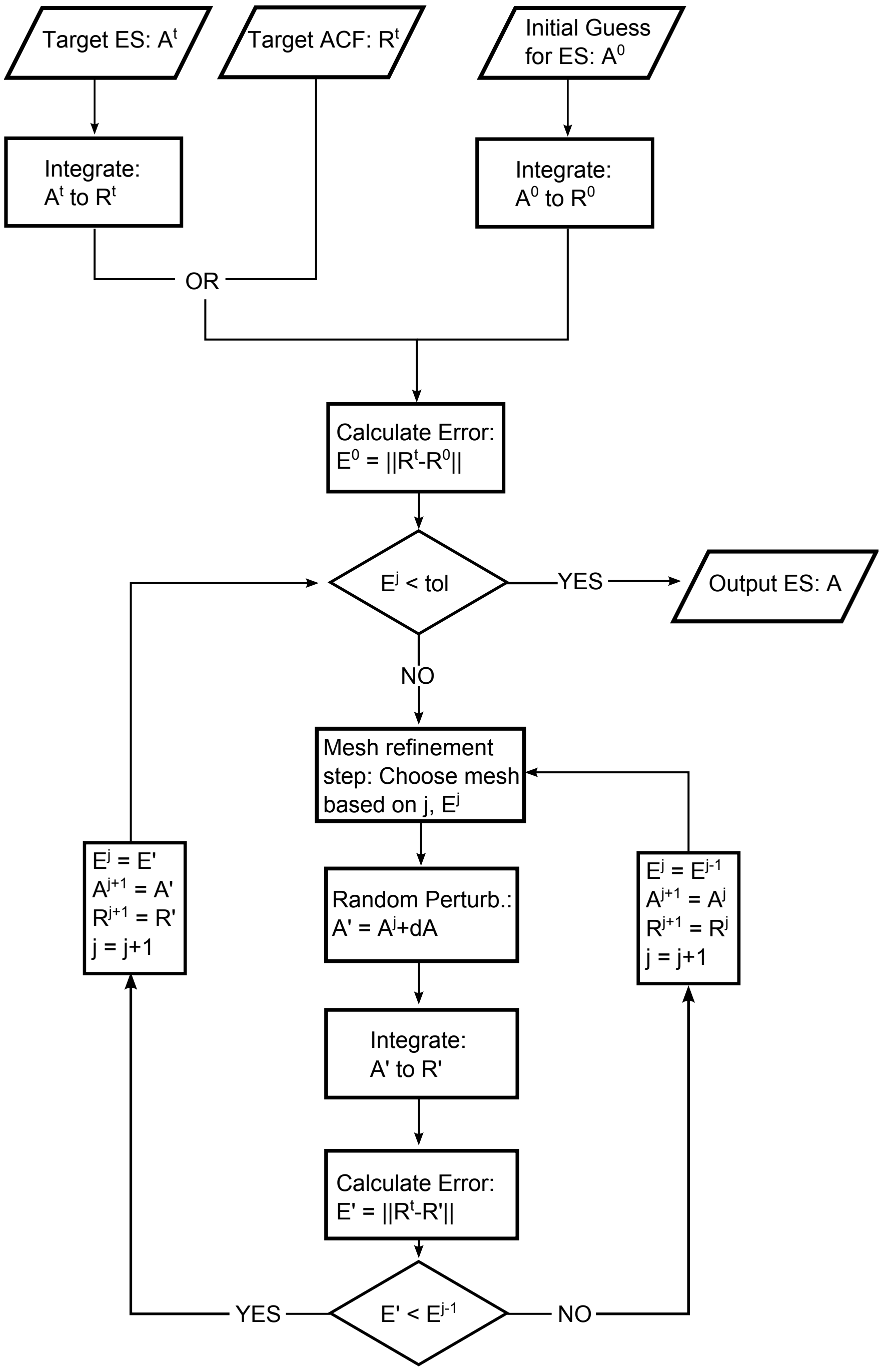
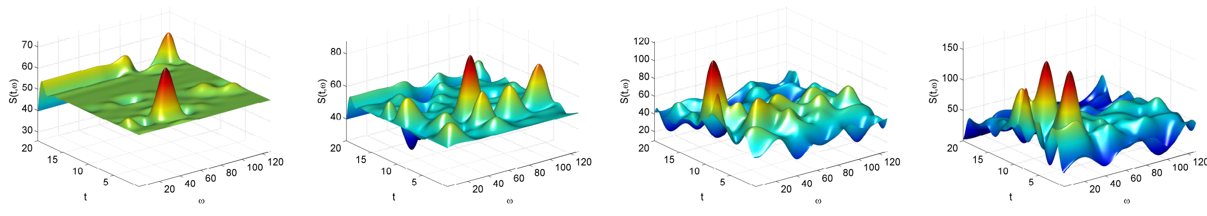

Target ES: A^t
Target ACF: R^t
Initial Guess for ES: A^0
Integrate: A^t to R^t
Integrate: A^0 to R^0
OR
Calculate Error: $E^0 = ||R^t-R^0||$
E^j < tol
YES
Output ES: A
NO
Mesh refinement step: Choose mesh based on j, E^j
Random Perturb.: $A' = A^j+dA$
Integrate: A' to R'
Calculate Error: $E' = ||R^t-R'||$
$E' < E^{j-1}$
YES
NO
$E^j = E'$ $A^{j+1} = A'$ $R^{j+1} = R'$ $j = j+1$
$E^j = E^{j-1}$ $A^{j+1} = A^j$ $R^{j+1} = R^j$ $j = j+1$

Figure 6.9: Multi-Grid Method (MGM) algorithm



(a) 10 iterations (b) 100 iterations (c) 500 iterations (d) 1000 iterations

Figure 6.10: Progression of the MGM over the first 1,000 iterations

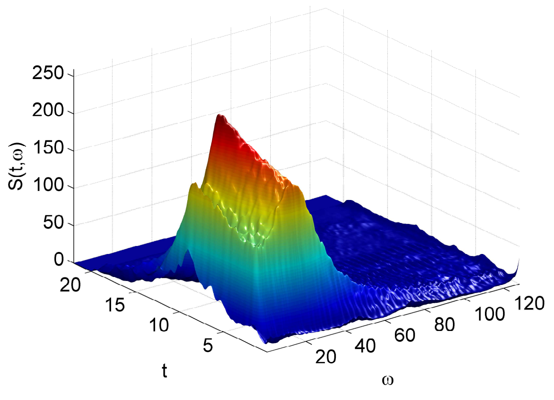
0.6%, and 7.5% measured in the ES. To reach an error level equivalent to the one million iterations of the BFM, we only need to take roughly one hundred thousand iterations - a full order of magnitude reduction in effort.

In addition to efficiency improvements, the MGM also preserves more continuity and smoothness in the ES than the BFM. We can see that the result in Figure 6.11(a) is quite smooth, and qualitatively looks closer to Figure 6.5(a) than the BFM result does (Figure 6.7(a)). Comparing the slices of Figures 6.7(b) and 6.11(b), the noise from the BFM is almost completely eliminated. This is especially true in the regions of high power, and thus higher importance.

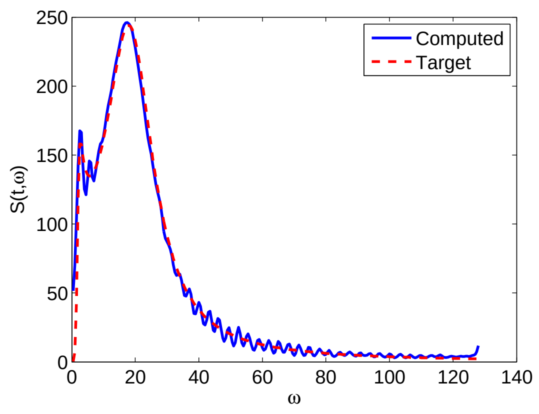
It should be noted that majority of the error in the ES is highly concentrated on the boundaries of the domain. Absolutely no assumptions were made as to "boundary conditions," so as to avoid any biases. Thus, either with constraints or post-processing, this could be cleaned up with some well justified assumptions. For example, it is always assumed that beyond the upper cutoff frequency the power is negligible. Thus, it is fairly safe to assume that the spectrum dies down to zero at $\omega = \omega_u$. Furthermore, depending on the physical process, $S^E(t = 0, \omega)$ and $S^E(t = T, \omega)$ can often be assumed to be zero.

Simulated samples are generated using the estimated and target spectra in Figure 6.12. Once again, the samples were generated using the same random phase angles so as to allow a direct comparison. Figure 6.12(a) shows the samples for the entire period, and Figure 6.12(b) shows two five second window zoom-ins. There is good agreement between the two samples, though the extra high-frequency content is showing up in the estimated sample. As was just described, this could be cleaned up in a post-processing of the spectrum, however.

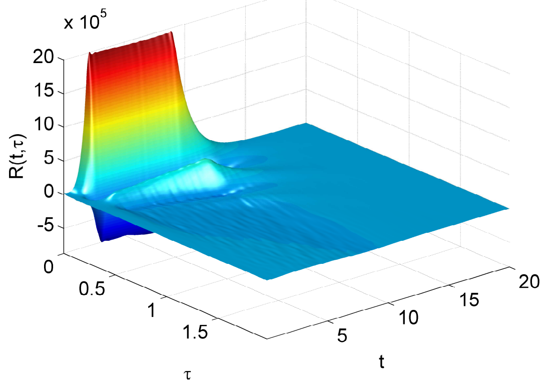
Finally, the MGM is compared with some established off-the-shelf optimization techniques in Figure 6.13. The MGM is clearly more efficient than the off-the-shelf methods. This goes back to the issue of parameter-space that was discussed at the outset of this section. These methods are all severely limited in the number of parameters they can optimize, and are not designed for the scale of this problem. Simulated Annealing very quickly runs out of memory, and Genetic Algorithm and Pattern Search both "bottom-out" very early on. In order to make use of these tools, the parameter space would have to be drastically reduced, and



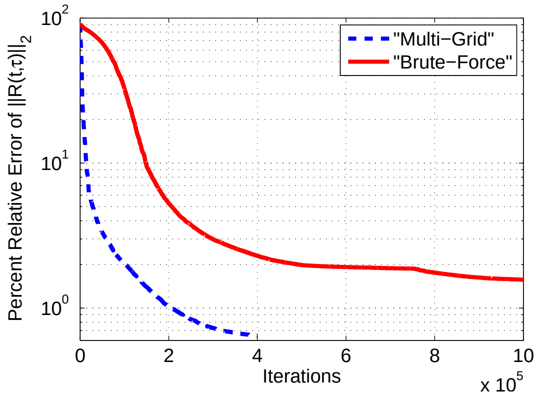
(a) MGM computed ES



(b) MGM sample ES "Slice"

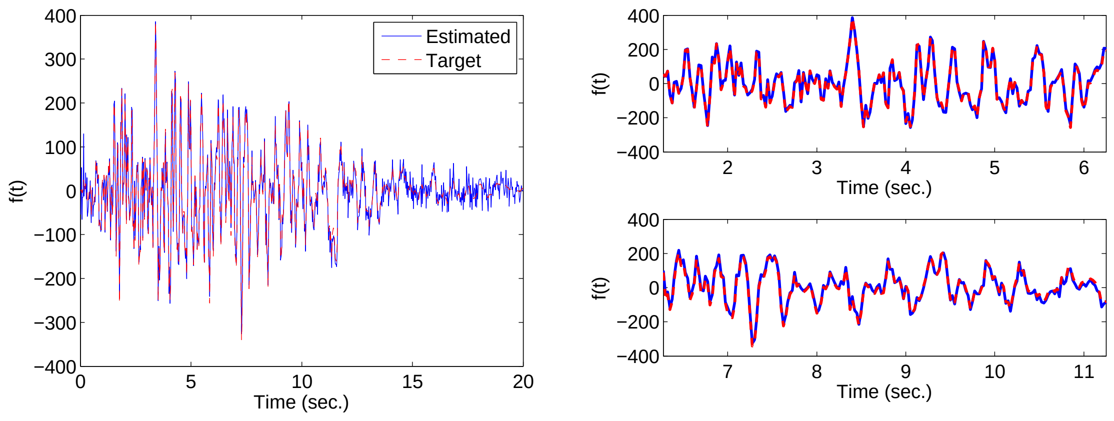


(c) MGM computed ACF



(d) MGM errors vs. iteration

Figure 6.11: Convergence results for Clough-Penzien (CP) spectrum with amplitude and frequency modulation. After 400,000 iterations of MGM, final error: 7.53% in ES and 0.64% in ACF



(a) Simulated sample realization comparison (b) Zoom in on sample realizations

Figure 6.12: Comparison of sample realizations generated using SRM with target ES (red dash) and estimated ES (solid blue)

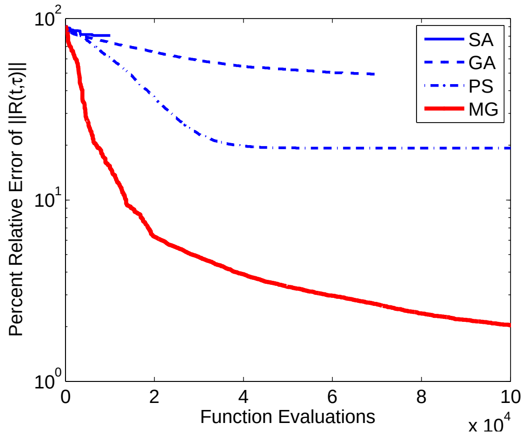


Figure 6.13: Comparison of the MGM with off-the-shelf optimization techniques: (1) Simulated Annealing, (2) Genetic Algorithm, and (3) Pattern Search from the MATLAB [42] Global Optimization Toolbox

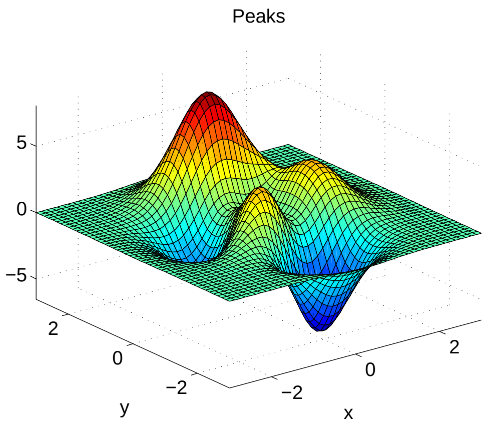


Figure 6.14: MATLAB `peaks` function

in doing so, fine-scale details would be lost. The MGM is clearly superior to these three methods for this problem. That begin said, a coupling approach could be explored to marry the MGM with some established optimization techniques.

MATLAB `peaks`

The second numerical example is non-physical and completely arbitrary. It is chosen only to demonstrate the robustness of the proposed method, and the method's ability to capture an arbitrary shape and number of peaks. The specific target chosen was the `peaks` function in MATLAB [42], which is shown in Figure 6.14, and is expressed as:

$$z = 3\left(1-x\right)^2 e^{-\left(x^2+(y+1)^2\right)} - 10\left(\frac{x}{5} - x^3 - y^5\right) e^{-x^2-y^2} - \frac{1}{3} e^{-\left((x+1)^2+y^2\right)} \tag{6.15}$$

where x and y are each defined between $(-3, 3)$. In order to use this as an example, the domain was scaled and translated in order to match that used in the earlier examples (with the same discretization). In order to preserve the non-negative condition, the absolute value of this function was used. The target ES and associated ACF are shown in Figures 6.15(a) and 6.15(c), respectively. After 200,000 iterations, the computed ES and ACF are shown in Figures 6.15(b) and 6.15(d), respectively. There is very nice agreement between the estimated and target ES and ACFs. The final errors after 200,000 iterations were 4.58% in the ES and 0.68% in the ACF. Once again, majority of the error in the ES was at the boundaries. Two sample slices of the ES are shown in Figures 6.15(e) and 6.15(f). The MGM captures the general shape of the peaks very nicely, and very accurately captures areas of high power. The sources of errors in the ES estimation are clear in Figures 6.15(e) and 6.15(f), as the boundaries and locations where the absolute value created weak discontinuity (just North of $\omega = 40$ in both plots). Aside from these two problem areas, the representation of the ES by the MGM is virtually perfect. In fact, looking at the convergence behavior in Figure 6.16, this

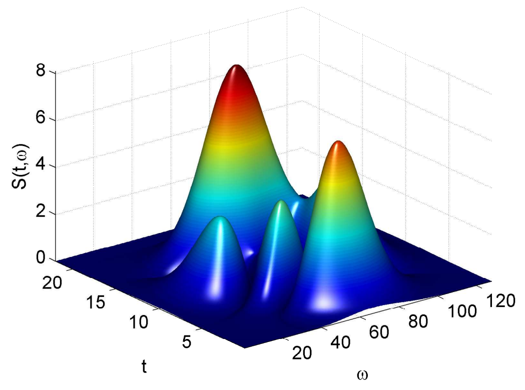
Table 6.1: BFM and MGM efficiency comparison

	BFM		MGM	
	Iteration	1,000,000 iter.	Iteration	400,000 iter.
Fortran serial	10.9870 sec.	18 weeks	11.3329 sec.	7.5 weeks
Fortran 24 core	1.2887 sec.	2 weeks	1.6386 sec.	1 week

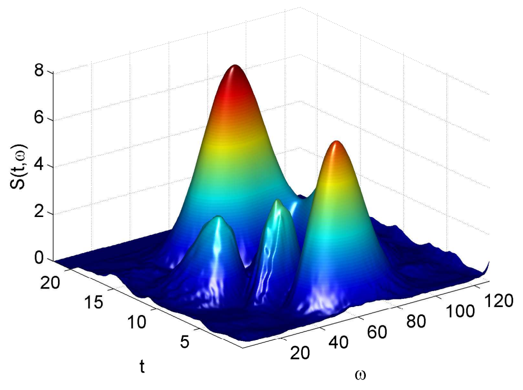
example was faster to converge than the previous. It reached errors in the ACF below 1% before 100,000 iterations. Finally, sample realizations are generated using the target ES (Figure 6.15(a)) and estimated ES (Figure 6.15(b)) and compared in Figure 6.17. Figure 6.17(a) shows the entire period, and Figure 6.17(b) shows two five second window zoom-ins. Once again, the MGM estimated ES produces a sample realization that agrees very nicely with the target sample. The biggest discrepancies are in extra high frequencies, and near the start and end of the duration, as expected due to the boundary errors.

6.4.3 Efficiency comparison

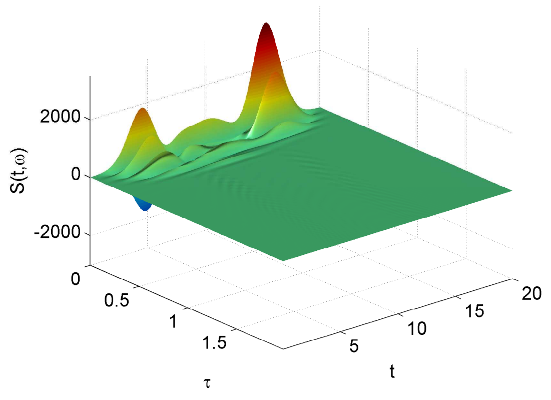
In the previous section, the MGM was shown to be orders of magnitude more efficient than the BFM in terms of iterations. However, it is important to consider the full computational cost, comparing the CPU time for both methods. Table 6.1 shows the average CPU time per iteration for both methods, on a 1024×1024 grid (note: this is finer than the results presented above). The MGM is only slightly slower per iteration. Though, when considering the fact that the BFM takes roughly 1,000,000 iterations to converge, and the MGM 400,000; it is clear that the MGM is still much more efficient than the BFM. In both cases, the advantage of parallelizing is clear - improvements of a full order of magnitude are possible by parallelizing when possible. Unfortunately, the algorithms are not fully parallelizable due to the interdependence of iterations. However, computation of the integral in Eqn. (6.2) is “embarrassingly parallel” in that each time instant is independent.



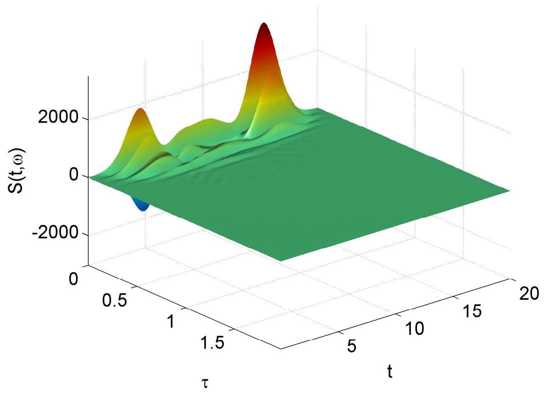
(a) Target ES



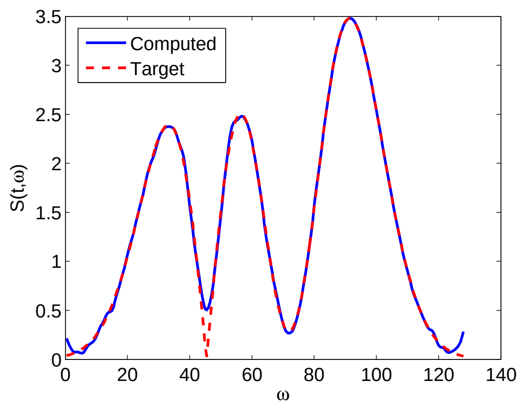
(b) Estimated ES



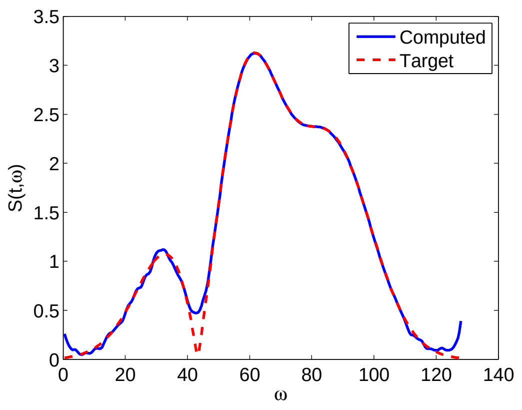
(c) Target ACF



(d) Estimated ACF



(e) Sample ES slice



(f) Samples ES slice

Figure 6.15: MATLAB `peaks` numerical example results. After 200,000 iterations of the MGM, final error: 4.58% in the ES and 0.68% in the ACF

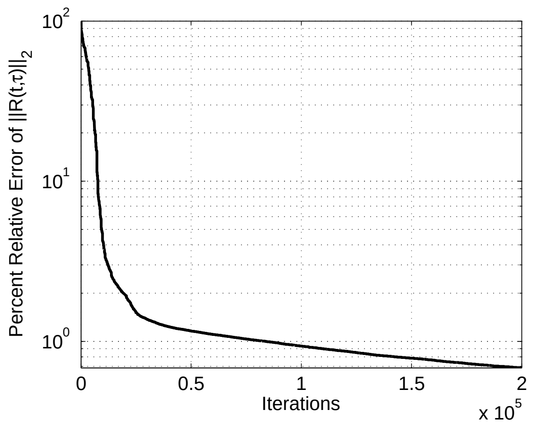
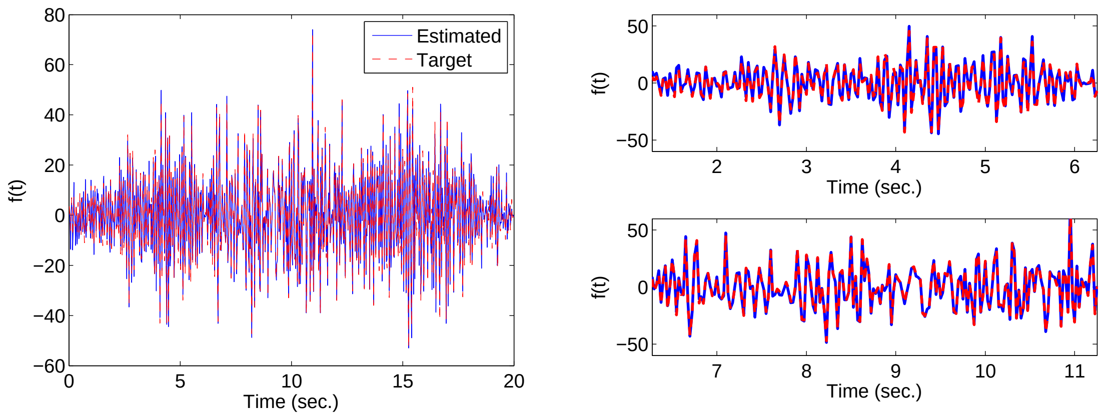


Figure 6.16: MGM MATLAB `peaks` example errors



(a) Simulated sample realization comparison

(b) Zoom in on sample realizations

Figure 6.17: Comparison of sample realizations generated using SRM with target ES (red dash) and estimated ES (solid blue) for `peaks` example

6.5 Preprocessing

6.5.1 Definition of "Pseudo-Spectrum"

Up through this point, the same initial guess was always used, which was simply: $A(t,\omega) = 50\ \forall\,(t,\omega)$. This was useful in demonstrating the robustness of the methods discussed earlier, however it is far from optimal for an initial guess. An intelligent initial guesses will drastically improve convergence, and so it is of the highest importance in these iterative methods. With that in mind, the use of a "Pseudo-Spectrum" is here proposed. This is computed by treating the process as *locally stationary*:

$$\tilde{S}^E(t,\omega) = \frac{1}{2\pi}\int\limits_{-\infty}^{\infty} R(t,\tau)\, e^{-i\omega\tau}\,\mathrm{d}\tau \tag{6.16}$$

It is important to note that in the non-stationary case, $R(t,\tau)$ is not even with respect to τ. So, unlike Eqn. (6.2), this can not be computed in a 'one-sided' fashion (where $A(t,\omega)$ is even with respect to ω), but rather both positive and negative values of τ must be considered to get the most accurate approximation. Furthermore, more careful consideration must be given here towards the discretization, to avoid aliasing effects. Namely, the following requirements must be met:

$$T = \frac{2\pi}{\Delta\omega} \tag{6.17a}$$

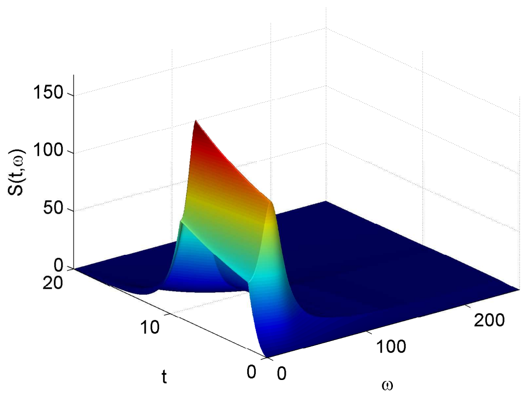
$$\Delta t \leq \frac{2\pi}{2\omega_u} \tag{6.17b}$$

6.5.2 Numerical examples

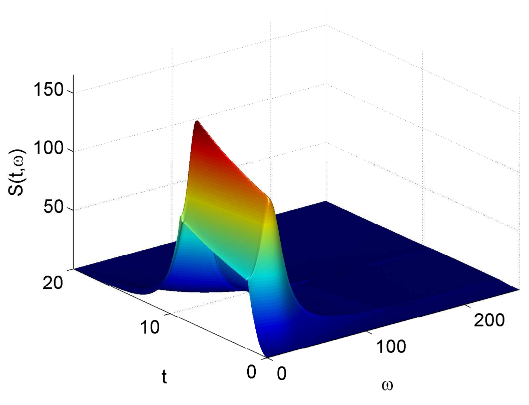
In this section, numerical examples will be carried out using the newly defined pseudo-spectrum. Three examples will be considered, (a) the Kanai-Tajimi spectrum, (b) the Kanai-Tajimi spectrum with Clough-Penzien correction, and (c) a linear chirp process. The first two are defined by Eqn. (6.9), by either ignoring or including the Clough-Penzien term, respectively. All other parameters were kept the same as in earlier examples. The linear chirp process will be defined in its subsection to follow. All examples in this section were carried out on a 1024×1024 mesh in time and frequency, with $0 \leq t \leq 20$ and $0 \leq \omega \leq 256$, unwise otherwise indicated. Thus, $\Delta t = 0.0196$ and $\Delta\omega = 0.2502$. Finally, the two sided lag ranged from $-10 \leq \tau \leq 10$ seconds. In regions where $t \pm \tau$ falls outside of the domain, $R(t,\tau)$ was assumed to be zero.

Kanai-Tajimi example

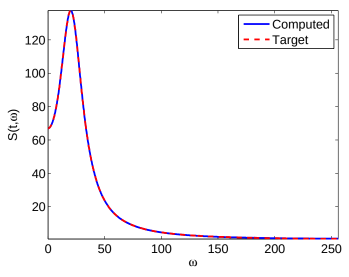
In this example, the Kanai-Tajimi ES from Eqn. (6.9) was used *without* the Clough-Penzien correction. The target ES is shown in Figure 6.18(a). The computed pseudo-spectrum is shown in Figure 6.18(b). For this case, the pseudo-spectrum already matches the target very well. In fact, the error in the ES is only



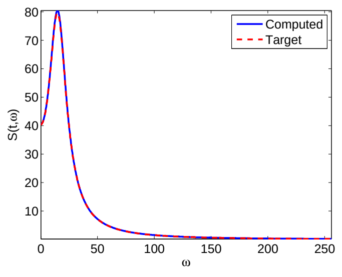
(a) Target Kanai-Tajimi ES



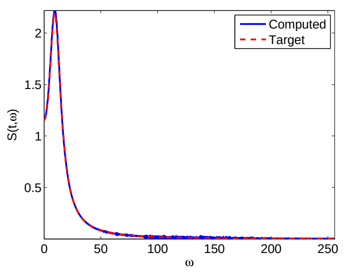
(b) Estimated pseudo-spectrum



(c) $t = 5$ sec.



(d) $t = 10$ sec.



(e) $t = 15$ sec.

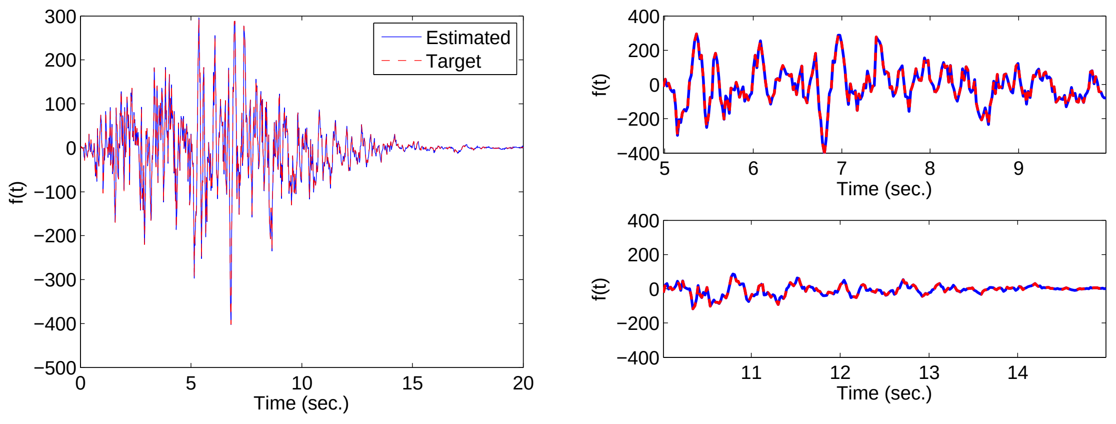
Figure 6.18: Pseudo-spectrum results for Kanai-Tajimi example

0.59%, and the error in the ACF is only 0.69%. Therefore, for this case, the pseudo-spectrum is most likely a good enough approximation of the ES. No perturbations (i.e. MGM) are necessary. Sample slices of the pseudo-spectrum are also shown in Figure 6.18.

Simulated sample realizations are shown in Figure 6.19. Figure 6.19(a) shows the entire duration and Figure 6.19(b) shows two five second window zoom-ins. The simulated samples are virtually indistinguishable. Showing that the pseudo-spectrum is in fact a very good approximation of this ES.

However, recall that the ES in practice would not be known a priori. So, after computing the pseudo-spectrum, only a notion of the error in the ACF would be known. There would be no way of knowing what the error in the ES actually was for an arbitrary ES. Without knowing what the error was after computing the pseudo-spectrum (as we will see shortly, not all cases are as good approximations as this one), MGM perturbations might be carried out regardless in order to ensure that the ES errors were low.

To that end, the MGM was performed with the pseud-spectrum of Figure 6.18(b) as an initial guess. The computed ES is shown in Figure 6.20(a), and the convergence behavior is shown in Figure 6.20(b). The errors in the ES do not lower drastically, but that is because the starting point was so low. In the end, the



(a) Simulated sample realization comparison

(b) Zoom in on sample realizations

Figure 6.19: Comparison of sample realizations generated using SRM with target ES (red dash) and estimated ES (solid blue) for pseudo-spectrum of Kanai-Tajimi example

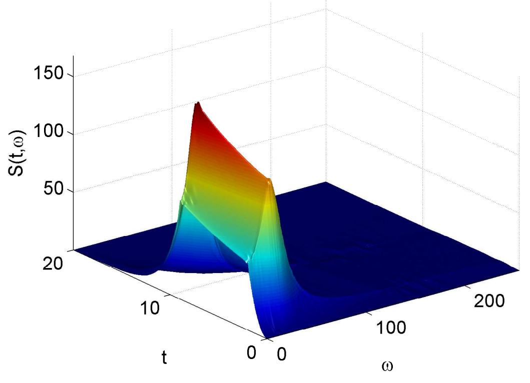
final errors were 0.53% in the ES and 0.38% in the ACF.

Clough-Penzien example

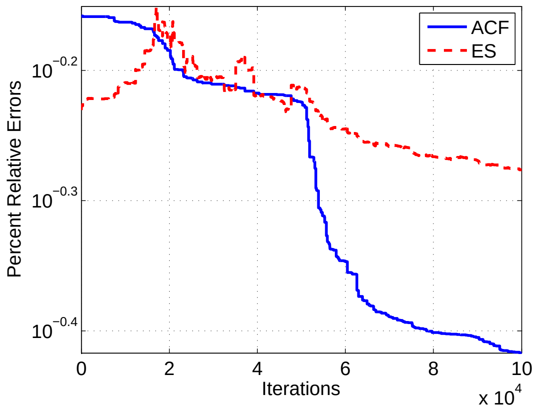
In this example, the Clough-Penzien ES from Eqn. (6.9) is revisited. The target ES is shown in Figure 6.21(a). The computed pseudo-spectrum is shown in Figure 6.21(b). The error in the ES is 2.55%, and the error in the ACF is 0.89%. From Figure 6.21(b), it can be seen that there is some clear difficulty capturing $S^E(t, \omega = 0)$. However, still in this case, the pseudo-spectrum is most likely a good enough approximation of the ES for practical purposes. No perturbations are necessary, once again. Sample slices of the pseudo-spectrum are also shown in Figure 6.21. There is some difficulty capturing the ES in the areas of low power, as evident in Figure 6.21(e).

Simulated sample realizations are shown in Figure 6.22. Figure 6.22(a) shows the entire duration and Figure 6.22(b) shows two five second window zoom-ins. The simulated samples agree nicely, though we can see some extra high frequency content in the tail (between 15 and 20 seconds). Still, this shows that the pseudo-spectrum is in fact a very good approximation of this ES.

Finally, once again, the MGM was performed with the pseudo-spectrum of Figure 6.21(b) as an initial guess. The computed ES is shown in Figure 6.23(a), and the convergence behavior is shown in Figure 6.23(b). The errors in the ES do not lower drastically, but that is because the starting point was so low. In the end, the final errors were 2.4% in the ES and 0.56% in the ACF.

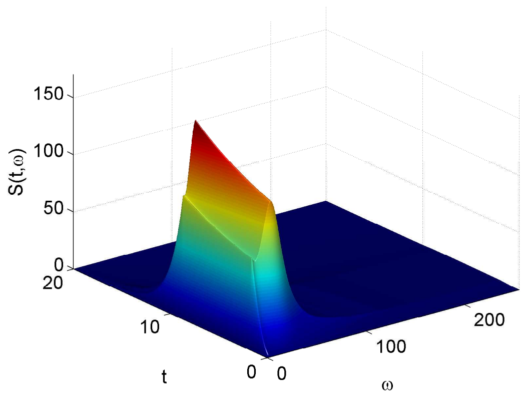


(a) Computed ES

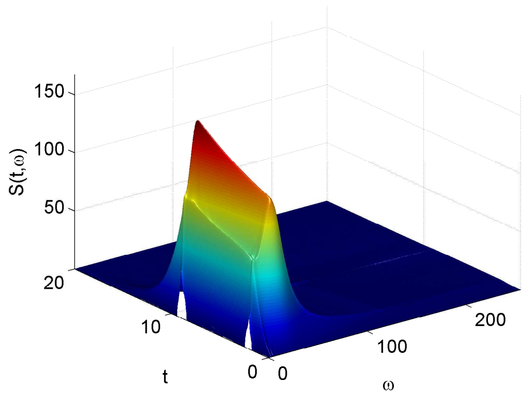


(b) Error vs. iterations

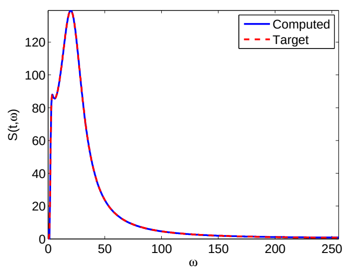
Figure 6.20: MGM with pre-processing for Kanai-Tajimi example. Final errors after 100,000 iterations: 0.53% in ES and 0.38% in ACF



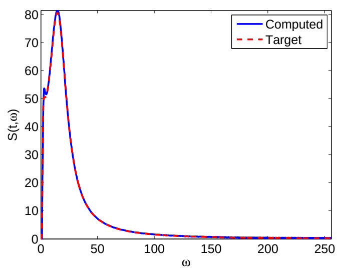
(a) Target Clough-Penzien ES



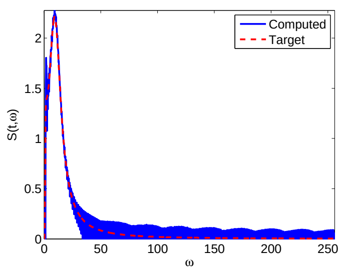
(b) Estimated pseudo-spectrum



(c) $t = 5$ sec.

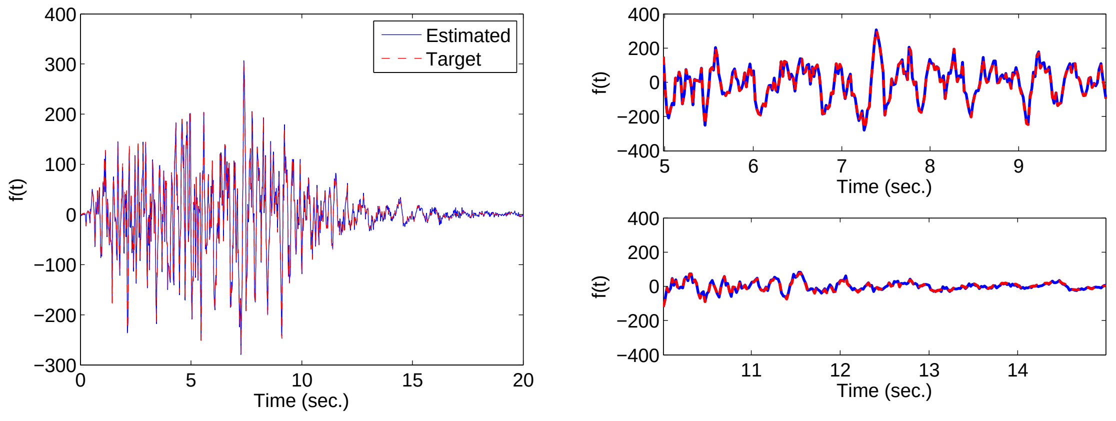


(d) $t = 10$ sec.



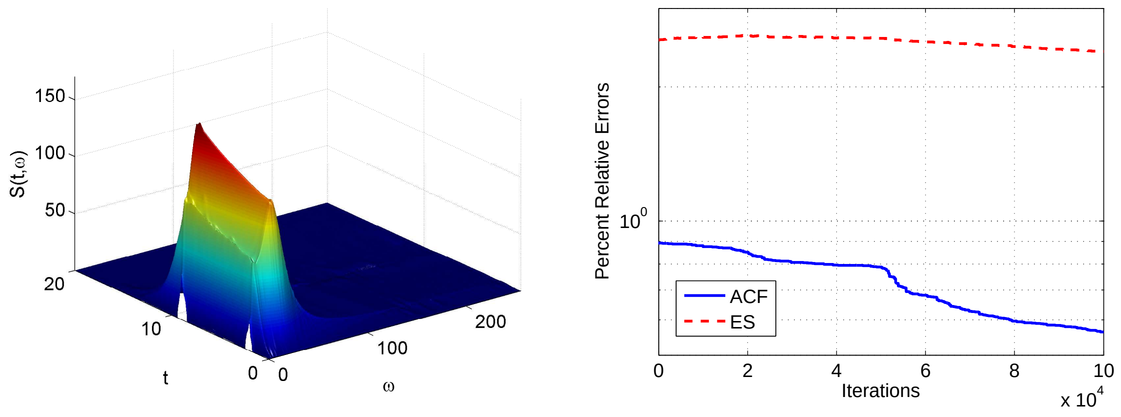
(e) $t = 15$ sec.

Figure 6.21: Pseudo-spectrum results for Clough-Penzien example



(a) Simulated sample realization comparison (b) Zoom in on sample realizations

Figure 6.22: Comparison of sample realizations generated using SRM with target ES (red dash) and estimated ES (solid blue) for pseudo-spectrum of Clough-Penzien example



(a) Computed ES (b) Error vs. iterations

Figure 6.23: MGM with pre-processing for Clough-Penzien example. Final errors after 100,000 iterations: 2.4% in ES and 0.56% in ACF

Linear chirp process

The final numerical example is a linear chirp process, in which the central frequency of a narrow-band process changes linearly with time. This example is considered considerably non-stationary, and thus should pose the most difficult to capture. The chirp ES is defined as:

$$S^E(t,\omega) = \exp\left[\frac{-\left(\omega - \omega_0(t)\right)^2}{4}\right] \tag{6.18}$$

where, in the case of a linear chirp, the frequency shift parameter, $\omega_0(t)$ is defined as:

$$\omega_0(t) = \frac{\alpha_T - \alpha_0}{T}t + \alpha_0 \tag{6.19}$$

where α_0 is the value of $\omega_0(t=0)$, $\alpha_T = \omega_0(t=T)$, and T is the period. In this example, the following values were used:

$$\alpha_0 = \frac{\omega_u}{4} \tag{6.20a}$$

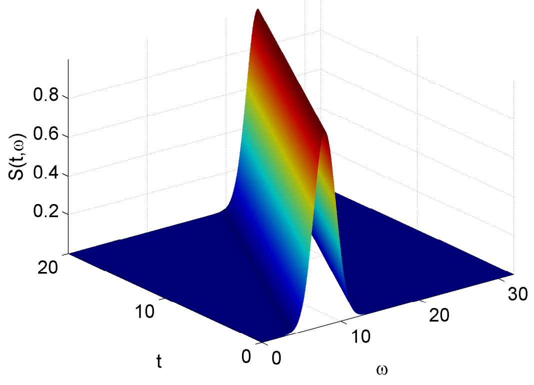
$$\alpha_T = \frac{3\omega_u}{4} \tag{6.20b}$$

$$\therefore \omega_0(t) = \frac{\omega_u}{4}\left(2\frac{t}{T} + 1\right) \tag{6.20c}$$

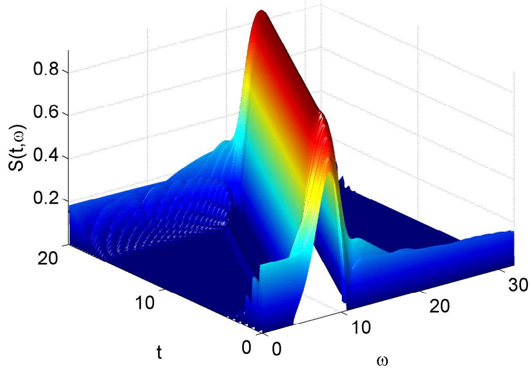
In this example, $\omega_u = 32$ radians per sec. was used. The target spectrum is shown in Figure 6.24(a). The estimated pseudo-spectrum is then shown in Figure 6.24(b). It is clear that the chirp estimation is not as accurate as the first two examples, namely towards $t = 0, T$. In the middle range, an approximate shape is found, and the correct peaks, though there is definite 'frequency smearing,' as evident in Figures 6.24(c), 6.24(d), and 6.24(e). This pseudo-spectrum produces an error of 19.67% in the ES, and 11.18% in the ACF. Thus, for the linear chirp process, there seems to still be a call for perturbations.

The MGM was performed with the pseudo-spectrum of Figure 6.24(b) as an initial guess. The computed ES is shown in Figure 6.25(a), and the convergence behavior is shown in Figure 6.25(b). In the end, the final errors were 9.5% in the ES and 1.3% in the ACF. The ACF error as been reduced drastically, and the ES error has been improved. Though, majority of this error is on the boundaries and could potentially be cleaned up in post-processing.

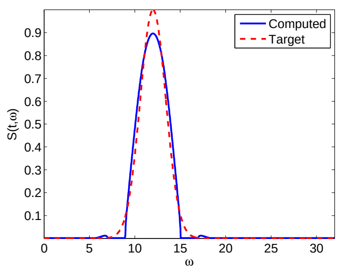
Simulated sample realizations are shown in Figure 6.26. Figure 6.26(a) shows the entire duration and Figure 6.26(b) shows two five second window zoom-ins.



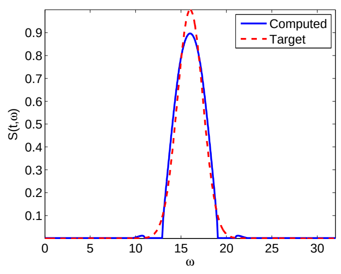
(a) Target linear chirp ES



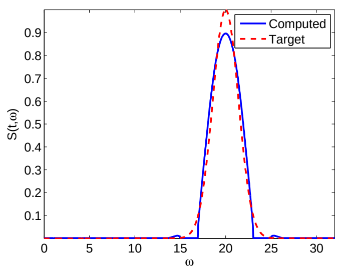
(b) Estimated pseudo-spectrum



(c) $t = 5$ sec.

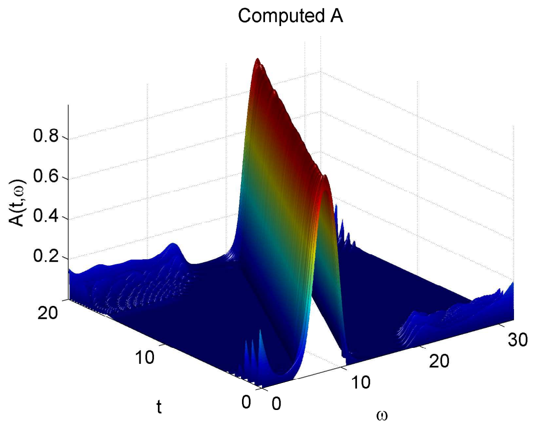


(d) $t = 10$ sec.

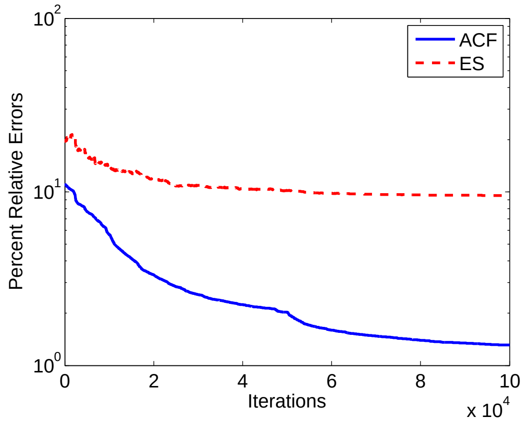


(e) $t = 15$ sec.

Figure 6.24: Pseudo-spectrum results for linear chirp example

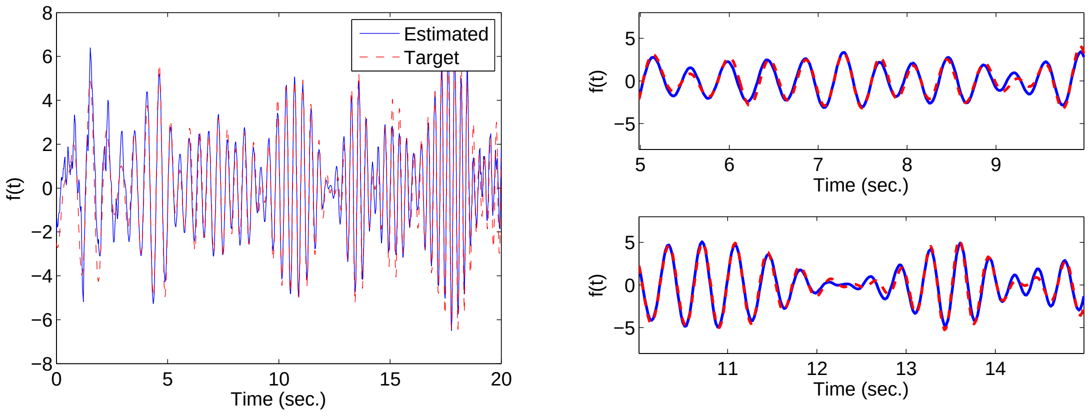


(a) Computed ES



(b) Error vs. iterations

Figure 6.25: MGM with pre-processing for linear chirp example. Final errors after 100,000 iterations: 9.52% in ES and 1.31% in ACF



(a) Simulated sample realization comparison

(b) Zoom in on sample realizations

Figure 6.26: Comparison of sample realizations generated using SRM with target ES (red dash) and estimated ES (solid blue) for MGM with pseudo-spectrum for linear chirp example

Chapter 7

Simulation of non-stationary and non-Gaussian stochastic processes

7.1 Introduction & motivation

So far in Part II, Chapter 5 introduced the Spectral Representation Method (SRM) for simulating various classes of stochastic processes, and Chapter 6 discussed some challenges in the simulation of non-stationary processes. Specifically, the inversion of Evolutionary Spectra (ES) from a prescribed or measured non-stationary Auto-Correlation Function (ACF) was discussed. The topic of simulating non-Guassian processes was briefly touched upon in Section 5.4, both stationary and non-stationary. In this chapter, the simulation of non-stationary, non-Gaussian stochastic processes and fields will be discussed. A novel approach to simulating non-stationary and non-Gaussian stochastic processes through the SRM will be presented, which, aside from one approximate technique proposed by Shields et. al. [99], has so far been undeveloped in the literature.

The importance of considering the non-stationarity of processes, as well as some examples, was already discussed in Sections 5.3 and 6.1. The consideration of the non-Gaussianity of a process, in both the stationary and non-stationary case, is equally important. Take, for example, random media such as the Functionally Graded Material (FGM) shown in Figure 6.2. An engineer may wish to simulate various aspects of this FGM such as its elastic properties (i.e. Young's modulus or Poisson's ratio), the physical distribution of different phases, or perhaps the mechanical strength. In all of these cases, the field to be simulated would be decidedly non-Gaussian, by definition. Elastic properties (aside from some *very* special cases) are always non-negative, and so by definition can not be Gaussian. Similar logic can be applied to the mechanical strength, which also is inherently non-negative. For a two-phase material, such as the Al-HDPE

FGM shown in Figure 6.2, the phase distribution would follow a binary distribution.

Just the same as with non-stationarity, it should be plain to see that careful consideration must be given to the non-Gaussianity when simulating sample realizations. If not, then the simulated realizations will be all but useless for practical purposes, as they will follow a completely incorrect distribution. The notion of a translation process was introduced in Section 5.4. In particular, recall that a stationary and Gaussian process, $Y(t)$, can be "mapped" to a stationary non-Gaussian process, $X(t)$, through the following [92]:

$$X(t) = F_{NG}^{-1}\left\{F_G\left[Y(t)\right]\right\} \tag{7.1}$$

where $F(\cdot)$ denotes the cumulative distribution function (CDF), with subscripts NG and G denoting non-Gaussian and Gaussian, respectively, and superscript -1 denoting the inverse. Recall, also, that this mapping was extended to non-stationary processes by Ferrante et. al. [98], as:

$$X(t) = F_{NG}^{-1}\left\{F_G\left[Y(t), t\right], t\right\} \tag{7.2}$$

in which case the CDF's may vary temporally.

Through the mappings of Eqns. (7.1) and (7.2) the simulation of non-Gaussian processes, either stationary or non-stationary, is straightforward conceptually. Recall, that sample functions generated through the SRM are asymptotically Gaussian due to the central limit theorem (Section 5.2). Therefore, underlying Gaussian samples can be generated with the SRM, and then mapped to a non-Gaussian image through the Eqns. (7.1) and (7.2). Thus, the challenge is in determining the underlying Gaussian process for which to simulate samples. Put differently, the correct Gaussian ES/SDF is needed, such that after the translation mapping the process possesses both the correct distribution *and* spectral characteristics/correlation structure. The literature is rich with methodologies for determining this underlying Gaussian process for the stationary case. In the non-stationary case, there exists an approximate technique by Shields et. al. [99], but otherwise has not been solved up to this point. In this chapter, a robust and "exact" methodology will be developed for determining the underlying non-stationary Gaussian process for simulating non-stationary and non-Gaussian stochastic processes. Robust because it is effective regardless of the degree of non-Gaussianity or non-stationarity, and exact because it makes no approximations. This is distinct from *analytical*, as the procedure is still numerical.

This chapter is organized as follows. Section 7.2 will first review the existing techniques for determining the underlying Gaussian process for simulation of stationary non-Gaussian processes in the SRM. Then, in Section 7.3 a novel methodology will be developed for determining the underlying Gaussian process for non-stationary and non-Gaussian processes. Numerical examples of this methodology will be carried out in Section 7.4.

7.2 Review of methodologies for simulating stationary non-Gaussian processes

It has so far been established that in order to simulate a stationary and non-Gaussian process, the underlying Gaussian process is generated through the SRM and then mapped to the non-Gaussian image through Eqn. (7.1). In order to simulate this underlying Gaussian process, though, the underlying Gaussian Spectral Density Function (SDF) is necessary for use in the SRM. While there is no formal and direct relationship between the non-Gaussian and Gaussian spectra, Grigoriu [92] showed that the ACFs of the two processes are related as:

$$R_{NG}(\tau) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F_{NG}^{-1}\{F_G[x_1]\} \cdot F_{NG}^{-1}\{F_G[x_2]\} \times \Phi\{x_1, x_2; \rho_G(\tau)\}\, \mathrm{d}x_1\mathrm{d}x_2 \tag{7.3}$$

where $\Phi\{x_1, x_2, \rho_G(\tau)\}$ is the joint Gaussian probability density (Eqn. (5.20)), and $\rho(\tau)$ is the normalized Gaussian correlation (Eqn. (5.24)).

This transformation is always possible in a *forward* direction, i.e. from Gaussian to non-Gaussian. For an arbitrary pair of R_{NG} and F_{NG}, the reverse ($NG \to G$) is not always possible. When this reverse transformation is not possible, R_{NG} and F_{NG} are called "incompatible" according to translation theory. In the former case, determining the underlying Gaussian process is trivial, as there is an analytical inversion to Eqn. (7.3). In the latter case, though, the underlying Gaussian process can only be determined numerically. Several approximate techniques exist to determine the underlying Gaussian process [93–97]. All of these methods are iterative techniques which aim to determine the underlying stationary and Gaussian SDF given a stationary and non-Gaussian process (either SDF or ACF, with corresponding CDF). From these methods, there is one distinct difference between [93–96] and [97]. The former four methods all involve generating sample functions and estimating the ensemble SDF, while Shields et. al. [97] proposed a simple and efficient iterative technique making use of Eqn. (7.3) and the Wiener-Khinchin transform (Eqn. (5.2)) to compute the SDF directly without the need to generate any sample functions.

As the algorithm in Shields et. al. [97] will serve as a basis for the algorithm developed in Section 7.3, a brief overview is given here. It is assumed that an incompatible pair of target SDF, $S_{NG}^{Targ}(\omega)$, and non-Gaussian CDF, F_{NG}, are given to start from. The algorithm is shown in Figure 7.1, and works as follows:

1. Make an initial guess for the Gaussian SDF, typically this is taken to be $S_G^{(0)}(\omega) = S_{NG}^{Targ}(\omega)$.
2. From $S_G^{(k)}(\omega)$, compute the associated Gaussian ACF, $R_G^{(k)}(\tau)$, using the Wiener-Khinchin transform of Eqn. (5.2b).

3. From $R_G^{(k)}(\tau)$, compute the associated non-Gaussian ACF, $R_{NG}^{(k)}(\tau)$, using Grigoriu's mapping in Eqn. (7.3).

4. From $R_{NG}^{(k)}(\tau)$, compute the associated non-Gaussian SDF, $S_{NG}^{Comp}(\omega)$, using the Wiener-Khinchin transform of Eqn. (5.2a).

5. If the relative difference between the computed and target SDFs is above the tolerance, then the guess for the Gaussian SDF is updated as:

$$S_G^{(k+1)}(\omega) = \left[\frac{S_{NG}^{Targ}(\omega)}{S_{NG}^{Comp}(\omega)}\right]^{\beta} S_G^{(k)}(\omega) \tag{7.4}$$

   where β is some parameter to control and optimize convergence. It is suggested to use a value in the range $1.3 \leq \beta \leq 1.5$ [97].

6. Repeat steps 2 through 5 until convergence.

In [97] this methodology was shown to be significantly more efficient than the former methods, quite accurate, and straightforward to implement.

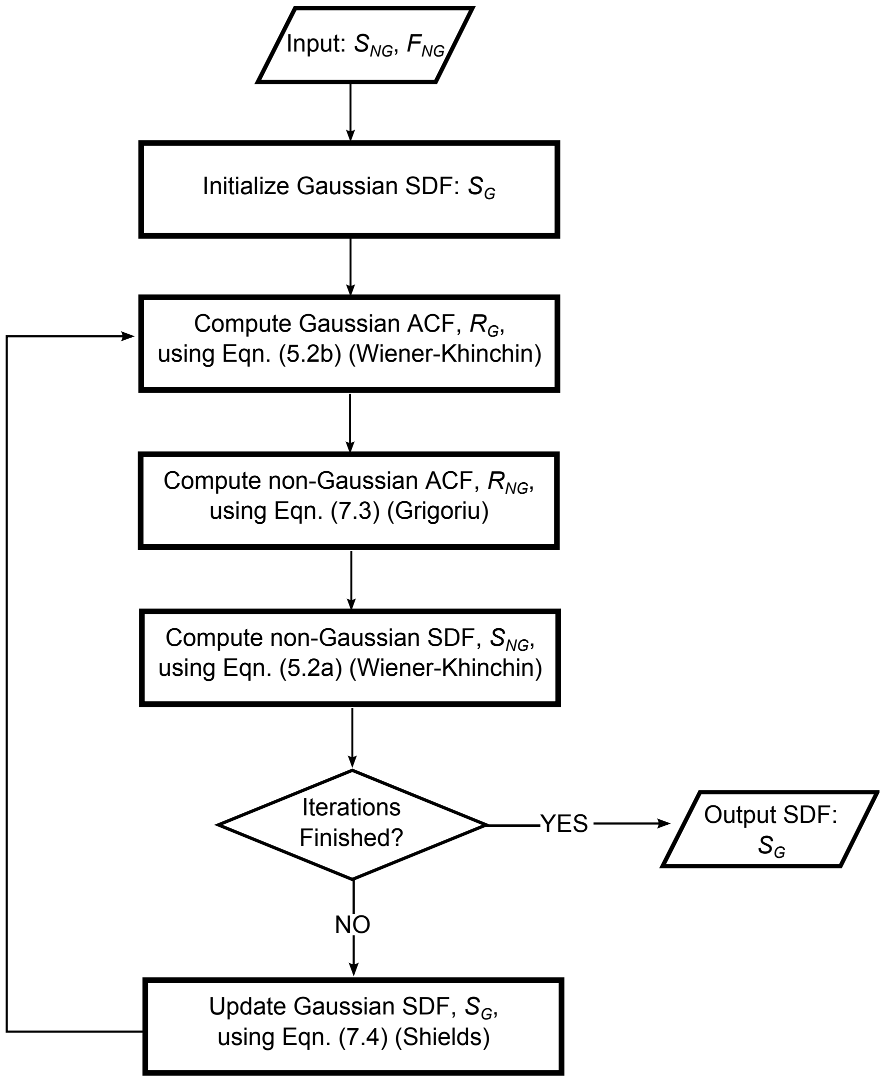

Input: S_{NG}, F_{NG}
Initialize Gaussian SDF: S_G
Compute Gaussian ACF, R_G, using Eqn. (5.2b) (Wiener-Khinchin)
Compute non-Gaussian ACF, R_{NG}, using Eqn. (7.3) (Grigoriu)
Compute non-Gaussian SDF, S_{NG}, using Eqn. (5.2a) (Wiener-Khinchin)
Iterations Finished?
YES
Output SDF: S_G
NO
Update Gaussian SDF, S_G, using Eqn. (7.4) (Shields)

Figure 7.1: Algorithm for determining underlying Gaussian SDF for a stationary and non-Gaussian stochastic process from Shields et. al. [97]

7.3 Proposed methodology for simulating non-stationary and non-Gaussian processes

Although the literature is relatively rich for the stationary case, for simulating non-stationary and non-Gaussian processes with the SRM, there is almost no work published. Generation of sample functions works in much the same way as in the stationary case described above. An underlying Gaussian and non-stationary process is first generated, and then mapped through Eqn. (7.2) to the non-Gaussian image. Once again the issue is determining the underlying Gaussian process.

Recall from Chapter 5 that Ferrante et. al. extended Eqn. (7.3) to account for non-stationarity as [98]:

$$R_{NG}(t,\tau) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F_{NG}^{-1}\{F_G[x_1,t],t\} \cdot F_{NG}^{-1}\{F_G[x_2,t+\tau],t+\tau\} \times \Phi\{x_1,x_2;\rho_G(t,\tau)\}\, \mathrm{d}x_1\mathrm{d}x_2 \tag{7.5}$$

In this case, once again, the inverse transformation is not always possible. However, there exists no robust methodology for determining the underlying Gaussian Evolutionary Spectrum (ES) for a given incompatible pair of non-Gaussian ES, $S_{NG}(t,\omega)$, and CDF, F_{NG}. The only method in the literature currently is an approximate technique developed by Shields and Deodatis [99]. In [99], the non-Gaussian ES is estimated for a given Gaussian ES through a series of approximations. Namely, the process is treated as locally stationary and a so-called pseudo-autocorrelation is found, conceptually similar to the pseudo-spectrum in the previous chapter (in fact, the pseudo-autocorrelation was the inspiration for the pseudo-spectrum). This pseudo-autocorrelation is then translated to the non-Gaussian image, and the non-Gaussian ES is approximated from this ACF. This estimated ES is used to update the guess for the Gaussian ES, similar to Eqn. (7.4). Because of these approximations, this methodology is only accurate for *weakly* non-stationary and *weakly* non-Gaussian processes. It is not accurate in general cases.

It is with this motivation that a method is proposed to determine the underlying Gaussian ES for prescribed incompatible pair of non-stationary and non-Gaussian ES/ACF and CDF. Specifically, due to the methodology presented in the previous chapter allowing for the inversion of a prescribed ACF to its ES, a direct extension of the stationary and non-Gaussian algorithm from [97], and described in Section 7.2, was made possible. This extension is formally outlined below, and numerical examples are carried out in Section 7.4.

The algorithm is shown in Figure 7.2. It is a direct extension of Figure 7.1 to non-stationarity. However, before the developments of Chapter 6, the step in the dashed box was not possible. In this particular step, the non-Gaussian ES is computed from the associated non-Gaussian ACF. As was discussed extensively in Chapter 6, this inversion was not known to exist, let alone possible to compute previously.

The algorithm works as follows, starting from an incompatible pair of target non-Gaussian ES, $S_{NG}^{Targ}(t,\omega)$, and CDF, F_{NG}:

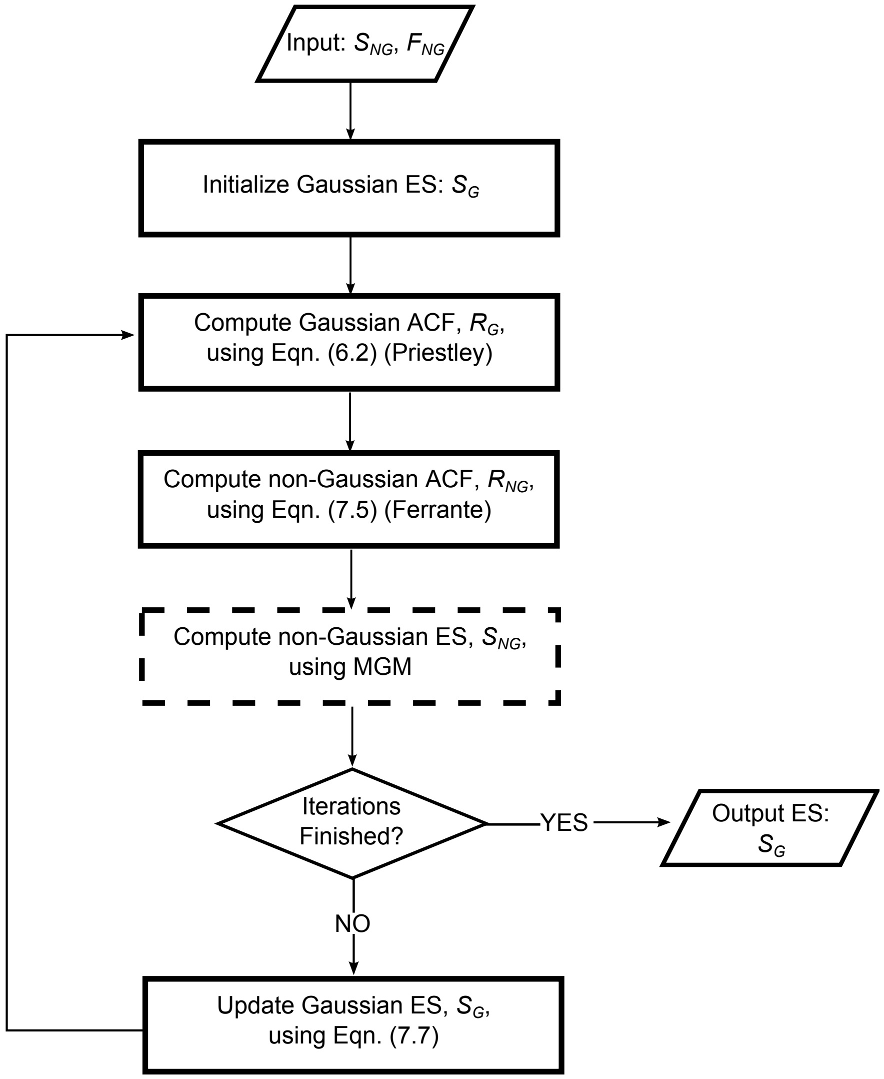

Input: S_{NG}, F_{NG}
Initialize Gaussian ES: S_G
Compute Gaussian ACF, R_G, using Eqn. (6.2) (Priestley)
Compute non-Gaussian ACF, R_{NG}, using Eqn. (7.5) (Ferrante)
Compute non-Gaussian ES, S_{NG}, using MGM
Iterations Finished?
YES
Output ES: S_G
NO
Update Gaussian ES, S_G, using Eqn. (7.7)

Figure 7.2: Proposed methodology for computing the underlying ES for a non-stationary and non-Gaussian process. The MGM of Chapter 6 made the step highlighted by a dashed border possible.

1. Make an initial guess for the Gaussian ES, typically this is taken as: $S_G^{(0)}(t,\omega) = S_{NG}^{Targ}(t,\omega)$.
2. Compute the Gaussian ACF $R_G^{(k)}(t,\tau)$ from the Gaussian ES, $S_G^{(k)}(t,\omega)$ using Eqn. (6.2).
3. Map the Gaussian ACF, $R_G^{(k)}(t,\tau)$ to the non-Gaussian ACF, $R_{NG}^{C}(t,\omega)$ using Eqn. (7.5).
4. Compute the non-Gaussian ES $S_{NG}^{C}(t,\omega)$ from $R_{NG}^{C}(t,\omega)$ using the methods developed and discussed in Chapter 6
5. Calculate the relative difference in the ES as:
$$\varepsilon_k = \frac{\left\| S_{NG}^{Targ} - S_{NG}^{C} \right\|}{\left\| S_{NG}^{Targ} \right\|} \times 100\% \tag{7.6}$$
where the norm $\|\cdot\|$ is defined in Eqn. (6.5).
6. If the difference, ε_k, is greater then some tolerance, update the Gaussian ES as:
$$S_G^{(k+1)}(t,\omega) = \left[\frac{S_{NG}^{Targ}(t,\omega)}{S_{NG}^{C}(t,\omega)} \right]^{\beta} S_G^{(k)}(t,\omega) \tag{7.7}$$
and return to step 2. Otherwise, output $S_G^{(k)}$.

The step that requires the greatest computational effort is # 4, or the dashed box in Figure 7.2, especially if the pseudo-spectrum is not a good enough representation of the ES and perturbations (i.e. the MGM) are required. Furthermore, because of the updating scheme in Eqn. (7.7), small discrepancies in areas of low power have the tendency to be magnified. Take, for example, if at some location (t^*, ω^*) in the domain, the process has essentially died down to zero power. Then, $S_{NG}^{Targ}(t^*, \omega^*) \approx 0$, for all intents and purposes. Numerically, though, the spectra will just be a small number. If the target ES has a value on the order of, say, 10^{-30}, and the computed has a value on the order of, say, 10^{-10}, numerically, they would both be considered "close enough" to zero on their own. However, taken together in Eqn. (7.7), their ratio would be on the order of 10^{20}. So, the value of the Gaussian ES at this point would quickly blow up.

In that regard, it is beneficial to "clean up" some of this noise, particularly in the areas of low power. A smoothing filter can be used to do this. In this work, a Savitzky-Golay [117] smoothing filter was used between steps # 4 and 5. The Savitzky-Golay filter uses local polynomial regression to smooth digital signals. It is advantageous over other techniques because it has the tendency to preserve relative maxima, minima, and widths. The Fortran subroutine from Press' Numerical Recipes text [118], as well as the MATLAB function `sgolayfilt` [42] can be used. In the examples shown in Section 7.4, the former was used.

7.4 Numerical examples

In this section, two numerical examples are carried out for the non-stationary and non-Gaussian algorithm described in the previous section. Specifically, the Kanai-Tajimi spectrum described in Eqn. (6.9) and shown in Figure (6.18(a)) was used. A slight variation was used such that $S(t=0,\omega) \neq 0$. This proved necessary to avoid normalization issues in the ACF, in Eqn. (5.21). Two separate non-Gaussian distributions were then used.

7.4.1 Log-normal distribution

The first distribution used was a log-normal distribution. The log-normal Probability Density Function (PDF) is defined as:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma_G \bar{x}} \exp\left[-\frac{(\ln \bar{x} - \mu_G)^2}{2\sigma_G^2}\right] \tag{7.8a}$$

$$\sigma_G^2 = \ln\left(1 + \frac{\sigma}{\bar{\mu}^2}\right) \tag{7.8b}$$

$$\mu_G = \ln \bar{\mu} - \frac{\sigma_G^2}{2} \tag{7.8c}$$

$$\bar{x} = x - \bar{\mu} \tag{7.8d}$$

where μ_G and σ_G are the mean and standard deviation of the associated Gaussian, or normal, distribution, $\bar{\mu}$ is the temporary mean used to shift the log-normal distribution in order to create a zero-mean process, and σ is the standard deviation of the log-normal distribution. This log-normal distribution has a support region of $x \in (-\bar{\mu}, \infty)$, and the following properties:

- Mean:

$$\bar{\mu} = e^{\mu_G + \frac{\sigma_G^2}{2}} \tag{7.9}$$

- Variance:

$$\text{Var} = \sigma^2 = \left(e^{\sigma_G^2} - 1\right) e^{2\mu_G + \sigma_G^2} \tag{7.10}$$

- Skewness:

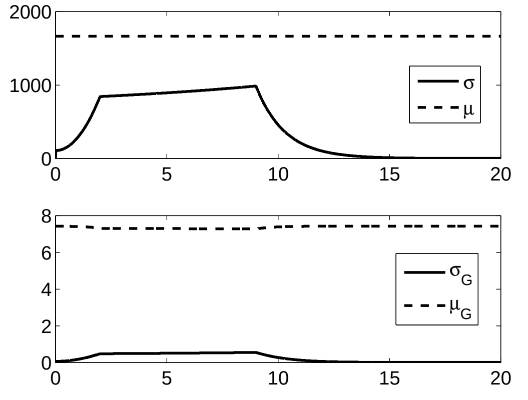
$$\gamma = \left(e^{\sigma_G^2} + 2\right)\sqrt{e^{\sigma_G^2} - 1} \tag{7.11}$$

- Kurtosis:

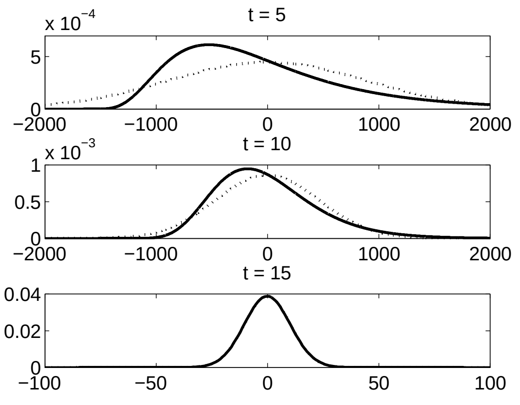
$$\text{Kurt} = e^{4\sigma_G^2} + 2e^{3\sigma_G^2} + 3e^{2\sigma_G^2} - 6 \tag{7.12}$$

The CDF of the log-normal distribution is:

$$F(x) = \Phi\left(\frac{\ln x - \mu_G}{\sigma_G}\right) \tag{7.13}$$



(a) Temporal variation of log-normal distribution parameters. The top plot depicts the actual standard deviation of the log-normal process, σ and the shifted mean, $\bar{\mu}$. The bottom plot depicts the parameters μ_G and σ_G, which are the mean and standard deviation of the associated normal distribution.



(b) Comparison of normal and log-normal distributions. Both distributions have identical variances, and zero mean. The dashed line in each plot shows the normal distribution, and the solid line shows the log-normal. Three time instances showing different variances are shown.

Figure 7.3: Log-normal distribution

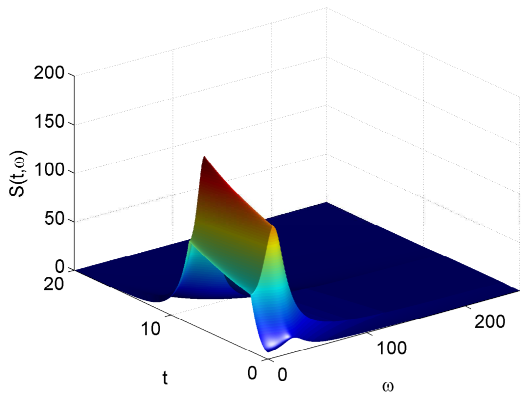
where $\Phi(\cdot)$ is the standard normal distribution. The CDF and inverse CDF for log-normal distributions were calculated using the `PROB` library for Fortran [119].

Of course, for a non-stationary process, all of these parameters can vary in time. In the example considered here, $\bar{\mu}$ was held constant, but the variance, σ^2, varied in accordance with the prescribed ACF. For a zero-mean process, $R(t, \tau = 0) = \sigma^2(t)$. The 'shift,' $\bar{\mu}$ was chosen to be 4 times the temporal average of the standard deviation, or:

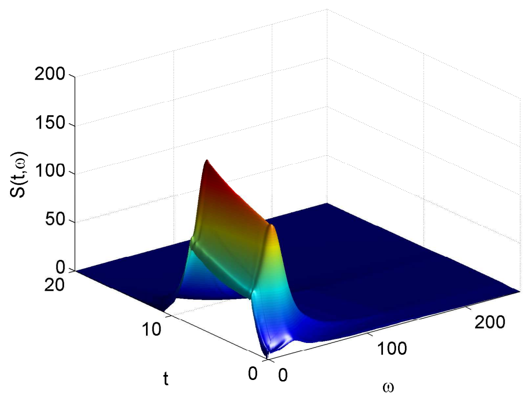
$$\bar{\mu} = 4 \times \frac{1}{T} \int_0^T \sigma(t)\ \mathrm{d}t = 4 \times \frac{1}{N_t} \sum_{j=1}^{N_t} \sigma_j \tag{7.14}$$

where, $\sigma_j = \sigma(t_j)$. The remaining parameters were therefore calculated according to Eqn. (7.8). All of the parameters describing the log-normal distribution are plotted in Figure 7.3(a). Namely, $\bar{\mu}$, σ, σ_G, and μ_G are plotted. Figure 7.3(b) then shows comparisons of the log-normal distribution and its comparable Gaussian distribution at three time instances. Note, that this comparable Gaussian is not $N(\mu_G, \sigma_G)$, but rather $N(0, \sigma)$ such that the two distributions have identical mean and variance.

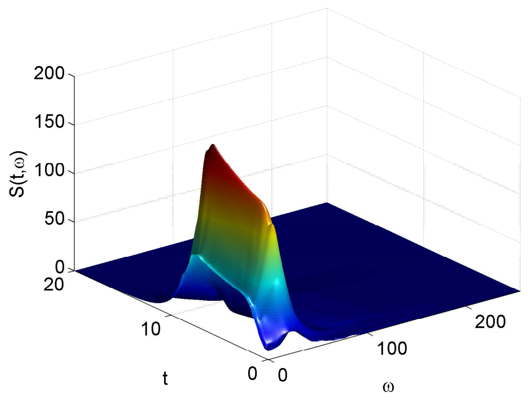
The results for the Kanai-Tajimi ES with this log-normal distribution are shown in Figures 7.4 and 7.5. After 20 iterations, the difference between the compatible and incompatible ES is just under 8%. Recall, though, that these two should not match each other perfectly. The target ES in this case was so-called *incompatible* according to translation theory. Thus, we were looking for the underlying Gaussian ES which



(a) Incompatible non-Gaussian ES



(b) Compatible non-Gaussian ES

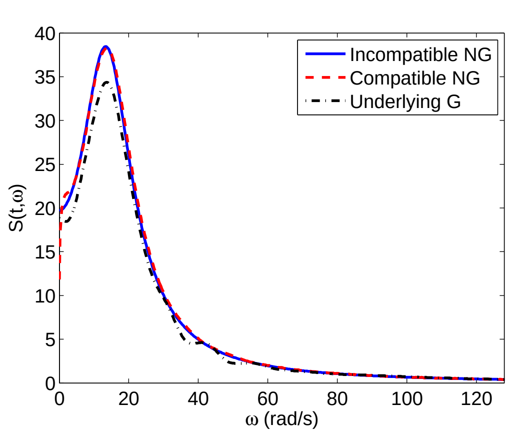


(c) Underlying Gaussian ES

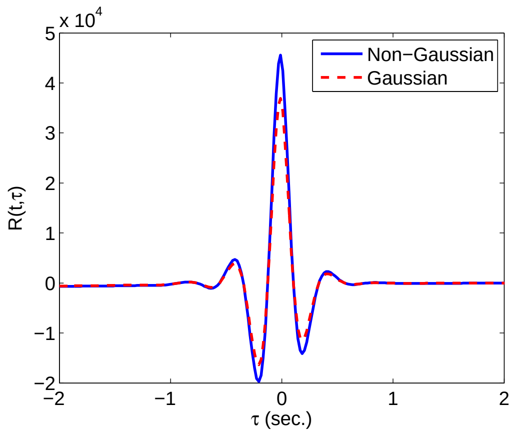
Figure 7.4: Log-normal Kanai-Tajimi process: evolutionary spectra

when mapped to the non-Gaussian image, matches the target as closely as possible. They will not match exactly, though, since the target was incompatible to begin with. The target, incompatible ES is shown in Figure 7.4(a). The computed, compatible non-Gaussian ES and underlying Gaussian are shown in Figures 7.4(b) and 7.4(c), respectively. A sample slice comparing the three ES is shown in Figure 7.5(a), and a sample slice showing the translated ACF is shown in Figure 7.5(b). Finally, a plot of the convergence of the algorithm is shown in Figure 7.5(c).

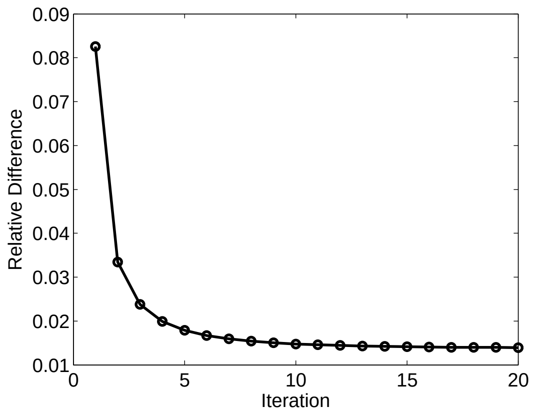
Some small differences are evident in the spectra, but there is definitely clear distortion in the correlation structure - due to the translation mapping of Eqn. (7.5). If we plot the (normalized) Gaussian ACF vs. the (normalized) log-normal ACF, we can quantify this distortion. This is plotted in Figure (7.6). The red dashed line depicts the "no distortion" case, where $\rho_G = \rho_{NG}$. The points plotted are the computed values of (ρ_{NG}, ρ_G). Deviations from the dashed line indicates correlation distortion.



(a) Sample slice of ES; $t = 10$ s.



(b) Sample slice of ACF; $t = 10$ s.



(c) Differences vs. iterations

Figure 7.5: Log-normal Kanai-Tajimi convergence

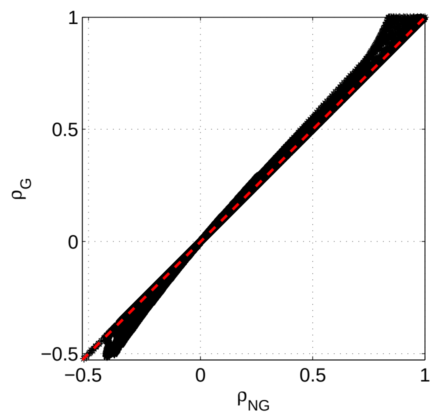


Figure 7.6: Distortion of the correlation structure for the log-normal Kanai-Tajimi process. The red dashed line depicts the “no distortion” case, where $\rho_G = \rho_{NG}$. The points plotted are the computed values of (ρ_{NG}, ρ_G).

7.4.2 Uniform distribution

The second example considered uses the same ES, but with a uniform distribution instead. The uniform PDF is defined as:

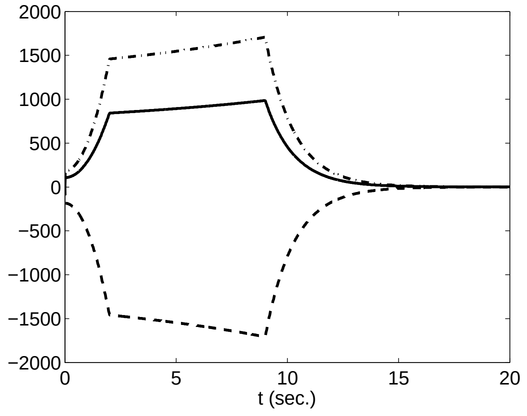
$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \le x \le b \\ 0 & \text{for } x < a \text{ or } x > b \end{cases} \tag{7.15}$$

where a and b are the lower and upper bounds of the domain. The uniform distribution has the following properties:

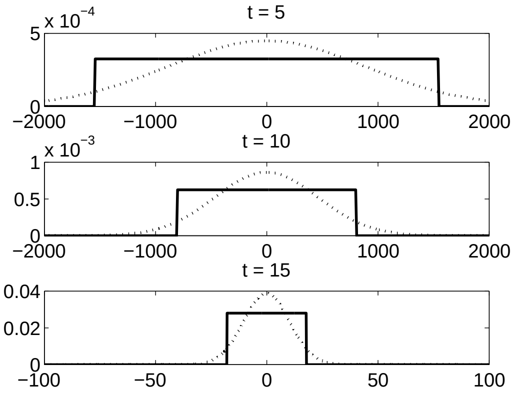
- Mean:
$$\mu = \frac{1}{2}(a+b) \tag{7.16}$$
  Thus, for a zero-mean uniform distribution, $a = -b$.
- Variance:
$$\sigma^2 = \frac{1}{12}(b-a)^2 \tag{7.17}$$
- Skewness: $\gamma = 0$
- Kurtosis: $-\frac{6}{5}$

The CDF for the uniform distribution is:

$$F(x) = \begin{cases} 0 & \text{for } x < a \\ \frac{x-a}{b-a} & \text{for } a \le x < b \\ 1 & \text{for } x \ge b \end{cases} \tag{7.18}$$



(a) Temporal variation of uniform distribution parameters. The plot shows the variation with respect to time of the standard deviation, σ, of the uniform distribution, as well as its upper, b, and lower, a, limits.



(b) Comparison of normal and uniform distributions. Both distributions have identical variances, and zero mean. The dashed line in each plot shows the normal distribution, and the solid line shows the uniform. Three time instances showing different variances are shown.

Figure 7.7: Uniform distribution

Unlike the log-normal distribution, this is quite straightforward to calculate numerically. Likewise, the inverse CDF is simply:

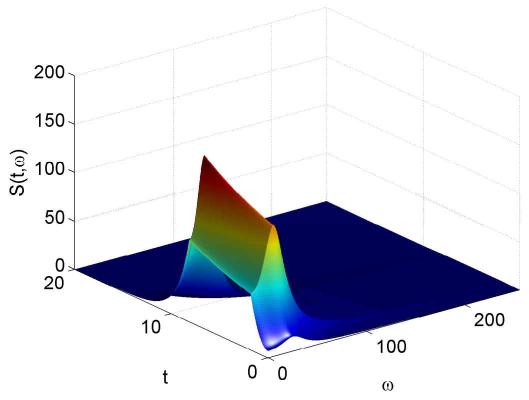
$$x = F(x)(b - a) + a \tag{7.19}$$

Though, this is only defined for $0 \leq F(x) \leq 1$.

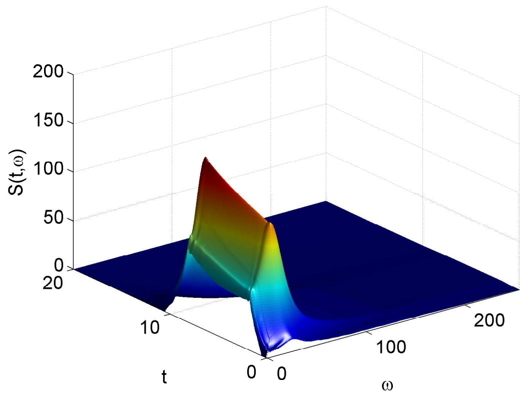
From the above equations, and the restriction that we have a zero-mean process, it is clear there is only one undefined parameter: $a = -b$. This parameter varies in time, and was chosen such that the variance matches the prescribed variance. Therefore, $b = \sqrt{3}\sigma$ and $a = -\sqrt{3}\sigma$. These parameters are plotted along with the standard deviation in Figure 7.7(a). Figure 7.7(b) then shows comparisons of the uniform distribution and its analogous Gaussian distribution at three time instances.

The results for the Kanai-Tajimi ES with this uniform distribution are shown in Figures 7.8 and 7.9. After 20 iterations, the difference between the compatible and incompatible ES is about 4.75%. The target, incompatible ES is shown in Figure 7.8(a). The computed, compatible non-Gaussian ES and underlying Gaussian are shown in Figures 7.8(b) and 7.8(c), respectively. A sample slice comparing the three ES is shown in Figure 7.9(a), and a sample slice showing the translated ACF is shown in Figure 7.9(b). Finally, a plot of the convergence of the algorithm is shown in Figure 7.9(c).

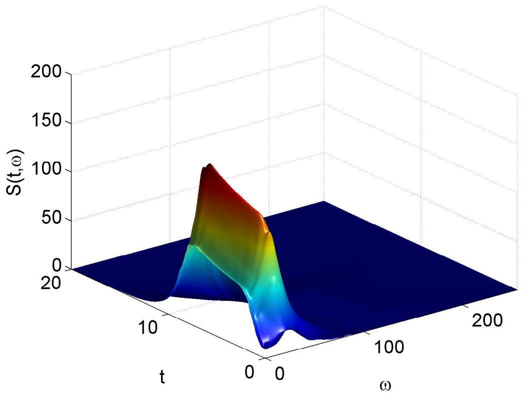
In contrast to the log-normal case, the uniform distribution appears to create very little distortion. We once again plot the correlation distortion in Figure 7.10. As would be expected from Figure 7.9(b), the correlation distortion plot of Figure 7.10 shows very little distortion. Even though the uniform distributions



(a) Incompatible non-Gaussian ES

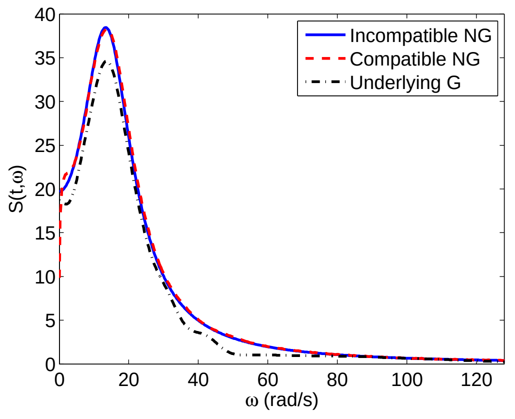


(b) Compatible non-Gaussian ES

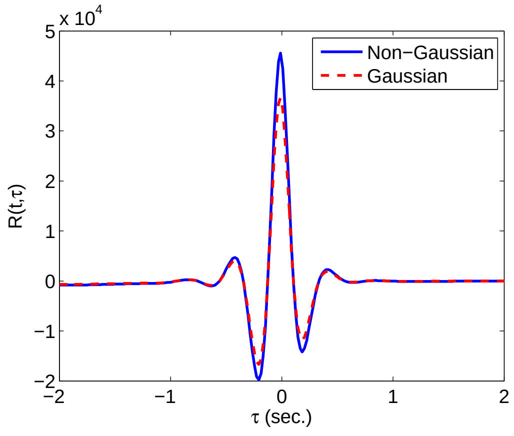


(c) Underlying Gaussian ES

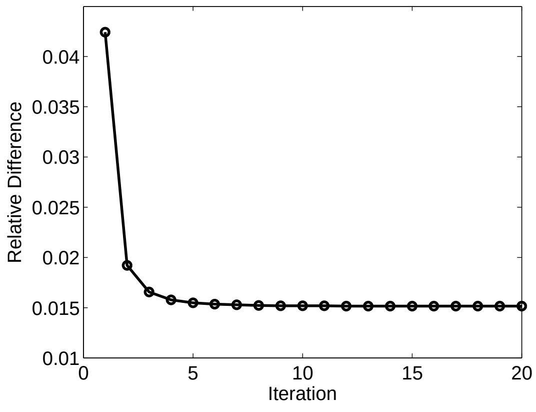
Figure 7.8: Uniform Kanai-Tajimi process: evolutionary spectra



(a) Sample slice of ES; $t = 10$ s.



(b) Sample slice of ACF; $t = 10$ s.



(c) Differences vs. iterations

Figure 7.9: Uniform Kanai-Tajimi convergence

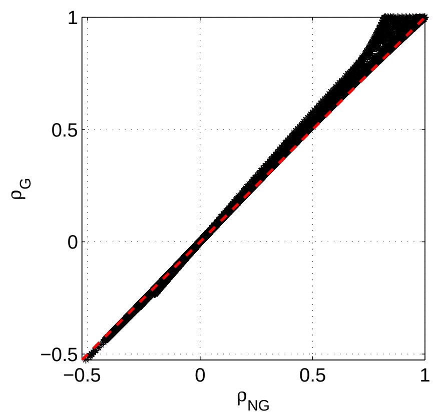


Figure 7.10: Distortion of the correlation structure for the uniformly distributed Kanai-Tajimi process. The red dashed line depicts the "no distortion" case, where $\rho_G = \rho_{NG}$. The points plotted are the computed values of (ρ_{NG}, ρ_G).

shown in Figure 7.7(b) are $s$trongly non-Gaussian, there is still little distortion to the correlation structure when mapped as a translation process.

Chapter 8

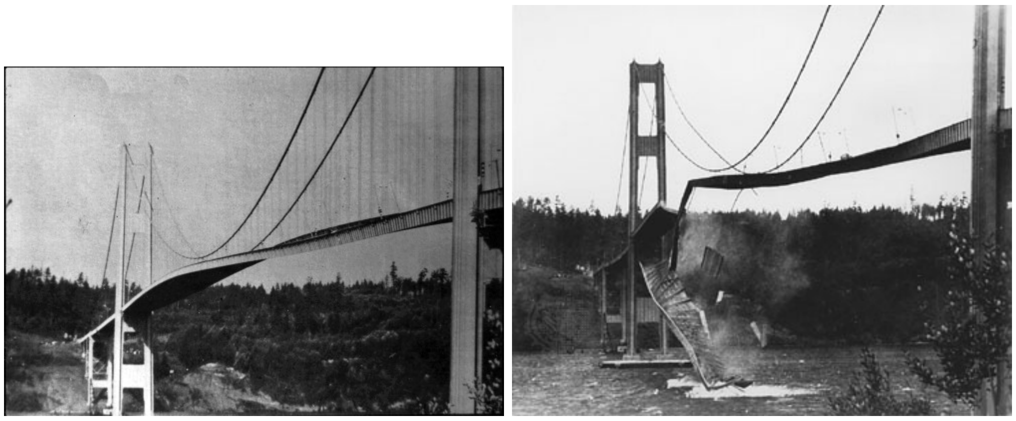
Simulation of wind velocities along long-span structures

8.1 Introduction & motivation

For tall buildings and long bridges alike, wind loads are often the governing design loads. Especially in the case of long-span bridges, the dynamic behavior under wind loading is likely to be the governing design condition. Consider the well known case of the Tacoma-Narrows bridge, which failed spectacularly under dynamic wind loading (Figure 8.1) - the aeroelastic flutter that eventually lead to its failure [120] would never be accounted for in a static analysis. In order to account for the randomness in the wind loading, this is often considered probabilistically [121]. To that end, Monte-Carlo simulation is typically used in practice. Sample realizations of wind velocity fields are generated, and the response of the bridge is calculated. Ensemble statistics can then be calculated from the different responses.

The most widely used method, and standard practice for generating wind velocities along the length of a bridge is to model the wind velocities as a multi-variate (m-v) process. Simulation of sample realizations of m-v processes within the Spectral Representation Method (SRM) was discussed in Chapter 5, Section 5.5. Recall, that the Cross-Spectral Density Matrix (CSDM), $\mathbf{S}(\omega)$ is decomposed into $\mathbf{S}(\omega) = \mathbf{H}(\omega)\mathbf{H}^{T*}(\omega)$ using Cholesky's decomposition [63, 82, 108]. Alternatively, a modal decomposition can be used [122–124]. It is well known, however, that for both modal and Cholesky decompositions, that there becomes an issue when the number of components in the vector process become large (on the order of 20-30 points). Due to the fact that neighboring points are highly correlated with each other, as the CSDM grows, it becomes increasingly closer to singular, and these decompositions break down.

To that end, recent research efforts have centered on approximate techniques to simplify this decom-



(a) Vibration of Tacoma Narrows bridge (b) Eventual collapse of Tacoma Narrows bridge

Figure 8.1: The Tacoma Narrows bridge, a well known and often used example of resonance failure due to wind loading. In fact, the failure was due to aeroelastic flutter [120]. In any case, dynamic analysis of the wind loading is necessary to avoid such failures. Images sources: (a) `http://en.wikipedia.org/wiki/File:Image-Tacoma_Narrows_Bridge1.gif` (b) `http://media-1.web.britannica.com/eb-media/21/59921-004-9B396A7B.jpg`

position. Some examples of this include the following non-exhaustive list. Yang, Chang, and Chang [125] developed an explicit form for the Cholesky decomposition under certain assumptions. In [126], Cao et. al. developed algebraic expressions to explicitly express the Cholesky decomposition. Li et. al. [127] proposed a simplified formulation, which breaks the $n-$variate vector process, with n correlated components, into smaller sized independent processes. These smaller vector processes (or even univariate processes) avoid the issues of large dimensionality, and are more computationally efficient. In [128], Ding et. al. reduce the computational effort by limiting the number of Cholesky decompositions necessary, however, the decomposition itself is not circumvented. For seismic applications, rather than wind, Gao et. al. [129] proposed the use of the lagged coherency matrix rather than the CSDM, since it is smoothly varying in frequency, and can then be coupled with interpolation methods to reduce the need for Cholesky decompositions. All of these methods, however, involve approximations to some degree. Furthermore, majority of these methods remain quite computationally intensive, with many decompositions necessary for each simulation.

In this chapter, an alternative formulation will be presented in which a virtually infinite number of spatial points can be simulated *exactly*. This is achieved by modeling the wind velocities as a continuous 'wave' in space-time rather than a discrete vector in space and continuous in time. Futhermore, the sample generation is comparatively efficient due to use of the Fast Fourier Transform (FFT). The chapter is organized as follows. Section 8.2 derives the frequency wavenumber spectrum used to describe the stochastic wave, and its relationship to the conventional m-v description of wind velocities. Section 8.3 will then expand upon the discussion of Section 5.6.2 and give a detailed discussion of the numerical implementation of simulating stochastic waves, specifically for the wind velocity problem. Finally, Section 8.4 will carry out a numerical example to show the proposed formulation's validity, effectiveness, and accuracy. All of the theory and examples in this chapter assume stationarity and homogeneity.

8.2 Modeling wind as a stochastic wave

8.2.1 Derivation of frequency-wavenumber (F-K) spectrum

In this section, we propose the use of the frequency-wavenumber (F-K) spectrum, $S(\omega, \kappa)$, to model the wind velocities as a stochastic "wave," continuous in both space and time. This allows the wind velocities to be modeled at a virtualy infinite number of points along the length of the structure. The Cross-Spectral Density Matrix (CSDM), or auto-spectrum and coherence model, can be transformed to the F-K spectrum through the following integral expression [100].

$$S(\omega, \kappa) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) \cdot \gamma(\xi, \omega) e^{i\kappa\xi} \mathrm{d}\xi \tag{8.1}$$

where $S(\omega)$ is the auto-spectrum, $\gamma(\xi, \omega)$ is the coherence model, ξ is the spatial separation distance, ω is angular frequency, and κ is wavenumber. When modeling wind velocities, the Kaimal spectrum [101] is typically chosen for the auto-spectrum, $S(\omega)$, and the Davenport model [102] is typically used for the coherence function, $\gamma(\xi, \omega)$. The Kaimal spectrum is defined (using parameter definitions from [130]) as [101]:

$$S_K(\omega) = \frac{1}{2} \cdot \frac{200}{2\pi} \cdot u_*^2 \cdot \frac{z}{U(z)} \cdot \frac{1}{\left[1 + 50\frac{\omega z}{2\pi U(z)}\right]^{\frac{5}{3}}} \tag{8.2}$$

where z is the height above ground, $U(z)$ is the average wind velocity at height z, and u_* is the velocity friction component defined as:

$$u_* = \frac{kU(z)}{\ln\left(\frac{z}{z_0}\right)} \tag{8.3}$$

where k is Von Karman's constant ($k = 0.4$), and z_0 is a parameter describing the ground roughness. The Davenport coherence function is defined as [102]:

$$\gamma(\xi, \omega) = e^{-\frac{\lambda\omega\xi}{2\pi U(z)}} \tag{8.4}$$

Combining Eqns. (8.1), (8.2), and (8.4), the F-K spectrum, $S(\omega, \kappa)$, can be expressed as:

$$S(\omega, \kappa) = \underbrace{\frac{1}{2} \cdot \frac{200}{2\pi} \cdot u_*^2 \cdot \frac{z}{U(z)} \cdot \frac{1}{\left[1 + 50\frac{\omega z}{2\pi U(z)}\right]^{\frac{5}{3}}}}_{\text{Kaimal spectrum}} \times \underbrace{\left[\frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-\frac{\lambda\omega\xi}{2\pi U(z)}} e^{i\kappa\xi} d\xi\right]}_{\text{Fourier integral of Davenport coherence}} \tag{8.5}$$

The Fourier integral in the second term has an known analytical solution [131]. Therefore, Eqn. (8.5) simplifies to:

$$S(\omega,\kappa) = \frac{1}{2}\cdot\frac{200}{2\pi}\cdot u_*^2\cdot\frac{z}{U(z)}\cdot\frac{1}{\left[1+50\frac{\omega z}{2\pi U(z)}\right]^{\frac{5}{3}}}\times\left[\frac{\left(\frac{\lambda\omega}{\pi U(z)}\right)}{\kappa^2+\left(\frac{\lambda\omega}{2\pi U(z)}\right)^2}\right] \tag{8.6}$$

The F-K spectrum in Eqn. (8.6) is a generalized, closed form, expression for modeling wind velocity fields in space-time as a stochastic wave. Sample realizations can be generated using the SRM formulation in Section 5.6.2, and that which will be discussed later in Section 8.3.

8.2.2 Verification of F-K spectrum

In order for the theory developed above, and the F-K spectrum of Eqn. (8.6), to be valid, the spectral density $S(\omega)$ should be the originally prescribed Kaimal spectrum, $S_K(\omega)$, at any point in the spatial domain. Newland [132] showed that for a multi-dimensional process, any dimension can be integrated out to give the uni-variate spectral density in the remaining dimension(s). Thus, for the two dimensional case of the wave being considered in this chapter:

$$S(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S(\omega,\kappa)\ \mathrm{d}\kappa \tag{8.7}$$

Here, the scaling of $\frac{1}{2\pi}$ is added to be consistent with definitions used throughout this thesis. For simplicity, it is convenient to define a quantity $\rho(\omega)$, such the F-K spectrum of Eqn. (8.6) can be rewritten as:

$$S(\omega,\kappa) = S_K(\omega)\times\left[\frac{\rho(\omega)}{\kappa^2+\left(\frac{\rho(\omega)}{2}\right)^2}\right] \tag{8.8}$$

where the function $\rho(\omega)$ is defined as:

$$\rho(\omega) = \frac{\lambda\omega}{\pi U(z)} \tag{8.9}$$

Substituting Eqn. (8.8) into Eqn. (8.7), and solving:

$$S(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_K(\omega)\times\left[\frac{\rho(\omega)}{\kappa^2+\left(\frac{\rho(\omega)}{2}\right)^2}\right]\ \mathrm{d}\kappa \tag{8.10a}$$

$$= \frac{1}{2\pi}S_K(\omega)\,\rho(\omega)\int_{-\infty}^{\infty}\frac{1}{\kappa^2+\left(\frac{\rho(\omega)}{2}\right)^2}\ \mathrm{d}\kappa \tag{8.10b}$$

$$= \frac{1}{2\pi}S_K(\omega)\,\rho(\omega)\times 2\pi\sqrt{\frac{1}{\rho^2(\omega)}} \tag{8.10c}$$

$$= S_K(\omega) \tag{8.10d}$$

Therefore, we can see that at any point in the domain the auto-spectral characteristics are intact, and still reflect the prescribed Kaimal spectrum.

It is also required that the coherence is retained throughout the spatial domain, however this considerably non-trivial to explore analytically. In Section 8.4, the auto-spectral and coherence characeteristics of generated samples will be verified through numerical tests.

8.3 Simulation of wind velocities as a wave

In order to simulate sample realizations of a stochastic wave the Spectral Representation Method (SRM) is used once again. The basic formulation for simulating stochastic waves in the SRM was given in Chapter 5, Section 5.6.2. However, much better results were found if a frequency shifting method was adopted, similar to that described by Zerva [79]. In the standard SRM, the frequencies are discretized such that:

$$\omega_j = j\Delta\omega \tag{8.11}$$

However, Zerva showed it was advantageous to shift the frequencies such that:

$$\omega_j = \left(j + \frac{1}{2}\right)\Delta\omega \tag{8.12}$$

There are several benefits to this formulation, as discussed in [79]. Worth noting are its (a) increased ensemble convergence rate, (b) simplified adaption to FFT, and (c) preservation of ergodicity properties. Specific to this example, note that there is a singularity at $S\left(\omega = 0, \kappa = 0\right)$ in Eqn. (8.6). This singularity is avoided by using this discretization, as the value for $S\left(\omega = \frac{\Delta\omega}{2}, \kappa = \frac{\Delta\kappa}{2}\right)$ is finite. In the case of a wave, both frequency, ω, and wave-number, κ, are discretized in this matter. Using this discretization, the SRM of Eqn. (5.35) can be used directly.

Use of the FFT can dramatically increase efficiency of simulation. A comparison between FFT and direct-summation is carried out in Appendix C. In order to make use of the FFT, Eqn. (5.35) is rearranged as:

$$X\left(x,t\right) = \mathrm{Re}\left[\sum_{l=0}^{N_\kappa-1}\sum_{m=0}^{N_\omega-1}\left\{B_{lm}^{(1)}\exp\left[i\omega_m t + i\kappa_l x\right] + B_{lm}^{(2)}\exp\left[-i\omega_m t + i\kappa_l x\right]\right\}\right] \tag{8.13}$$

where, $\mathrm{Re}\left[\cdot\right]$ denotes the real part, and:

$$B_{lm}^{(n)} = 2\sqrt{S\left(\omega_m, \kappa_n\right)\cdot\Delta\omega\Delta\kappa}\,e^{i\phi_{lm}^{(n)}} \tag{8.14}$$

Using the notation from Chapter 5, Eqn. (8.13) can be re-written as a series of FFTs:

$$X\left(x,t\right) = \mathrm{Re}\left\{\mathrm{FFT}_\kappa\left[\mathrm{FFT}_\omega\left(\mathbf{B}^{(1)}\right)\right] + \mathrm{FFT}_\kappa\left[\mathrm{IFFT}_\omega\left(\mathbf{B}^{(2)}\right)\right]\right\} \tag{8.15}$$

where $\mathrm{FFT}\left(\cdot\right)$ and $\mathrm{IFFT}\left(\cdot\right)$ denote the Fast Fourier Transform and Inverse Fast Fourier Transform, respectively, and subscripts ω and κ denote along which dimension the (I)FFT is computed. Appendix C gives

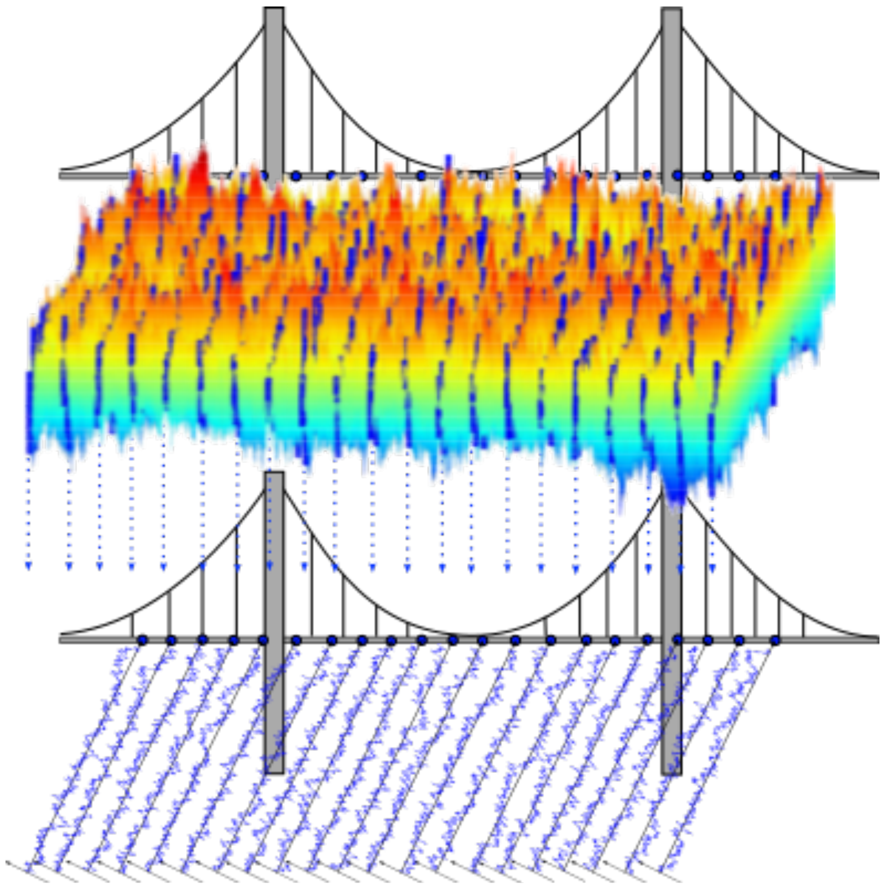


Figure 8.2: Simulation of wind velocities as a wave: The top image shows the simulated wave sample, and the bottom shows the individual "slices" that make up the time histories at each point of interest along the bridge

a more detailed explanation for the implementation of this simulation, as well as some example MATLAB code.

Once the wave is simulated, individual points in space can be picked out, as necessary or desired. As a conceptual example, consider a bridge with a 1,000 meter main span. The designer builds a FEM model of the bridge with 10 meter segments representing the deck, and thus would like wind velocity time histories for each of the 100 nodes along the span. Using the conventional multi-variate methods, this would be impossible. In fact, those methods break down after about 20-30 points. Thus, the wind 'field' is modeled as a continuous wave as described above. With the continuous wave sample, the individual time histories at the 100 locations are simply the "slices" where x is held constant, for each of the desired x_i. This is illustrated in Figure 8.2. The continuous wave is shown on top, with slices highlighted in blue. These slices are then shown below, as individual time histories.

8.4 Numerical example

8.4.1 Problem definition

In this section a numerical example is considered, and the generated samples are tested to verify that their spectral characteristics match the desired auto-spectrum and coherence function. In other words, it is ensured that the samples simulated via the wave method and multi-variate method are *statistically* equivalent. The example considered is a bridge with a 1600 meter long main span. The other parameter definitions are as follows:

- $L = 1600$ m
- $z = 50.0$ m
- $z_0 = 0.03$ m
- $U(z) = 40.0$ m/s

Although the wind velocities are being modeled as a continuous wave, the wave is still discretized for numerical simulation. The frequency-wavenumber discretization used is:

- $\omega_u = 8\pi$ rad/s
- $N_\omega = 1,024$
- $\Delta\omega = \frac{\omega_u}{1023} \approx 0.0246$ rad/s
- $N_\kappa = 16,368$
- $\kappa_u = \frac{2(N_\kappa - 1)}{L}\pi = \frac{16367}{800}\pi \approx 64.2731$ 1/m
- $\Delta\kappa = \frac{2\pi}{L} = \frac{\pi}{800} \approx 0.0039$ 1/m

Though, due to the frequency shifting paradigm of [79], as described above, the upper-cutoff frequency and wavenumber are actually shifted to:

- $\omega_u' = \omega_u + \frac{\Delta\omega}{2} \approx 25.1450$ rad/s
- $\kappa_u' = \kappa_u + \frac{\Delta\kappa}{2} \approx 64.2750$ 1/m

The other discretization parameters were kept the same (i.e. number of points and step sizes in both frequency and wavenumber). Time and space are therefore discretized as:

- $N_t = 2N_\omega = 2,048$

- $T = \frac{2\pi}{\Delta\omega} \approx 255.6251$ s
- $\Delta t = \frac{T}{N_t} = \frac{\left(\frac{2\pi}{\Delta\omega}\right)}{N_t} \approx 0.1249$
- $N_x = 2N_\kappa = 32,736$
- $\Delta x = \frac{L}{N_x} \approx 0.0489$ m

Before getting into the results, there are some items in the discretization worth noting. First, note that $N_t = 2N_\omega$ and $N_x = 2N_\kappa$. This is because when using the FFT, the frequency is actually doubled and then padded with zeros. This is in accordance with the formulation in [112]. Also note, that $N_x = 32,736$. This high N_x is chosen (a) to demonstrate the ability of this method to simulate a very high number of points in space, and (b) it was found through trial and error that this combination of k_u and $\Delta\kappa$ produced the best results - in terms of avoiding aliasing as well as closeness of fit to target coherence. All N's are chosen as powers of 2 to optimize the FFT efficiency. Finally, the specific relationship between frequency and time, and that of space and wavenumber, are in accordance with the MATLAB FFT routines, which were used in this chapter.

8.4.2 Results

In this numerical example, 5,000 samples are generated from the F-K spectrum. At the start of the simulation, an index was chosen at random. In the results shown here, this index was $j = 476$, for which $x_j = 23.2160$ m. The auto-spectrum was estimated at this location in space for each of the 5,000 samples, and the ensemble average of these auto-spectra is shown in Figure 8.3 compared with the prescribed Kaimal spectrum. For sample function 'slice' $X(t, x_j)$, the auto-spectrum can be estimated as:

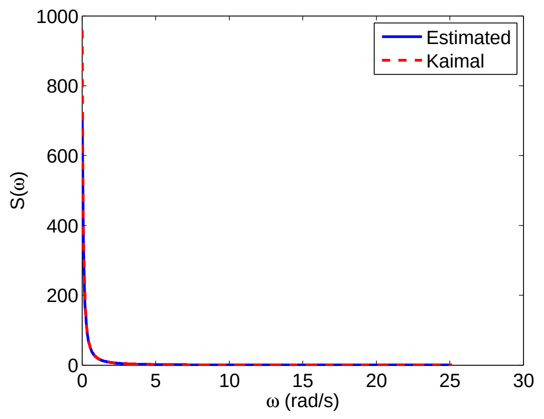
$$S_{jj}(\omega) = \frac{\left|\int_0^T X(t, x_j)\, e^{-i\omega t}\, \mathrm{d}t\right|^2}{2\pi T} \tag{8.16a}$$

$$= \frac{\left|\mathrm{FFT}\left(\mathbf{X}^{(j)}\right)\right|^2}{\Delta\omega M_\omega^2} \tag{8.16b}$$

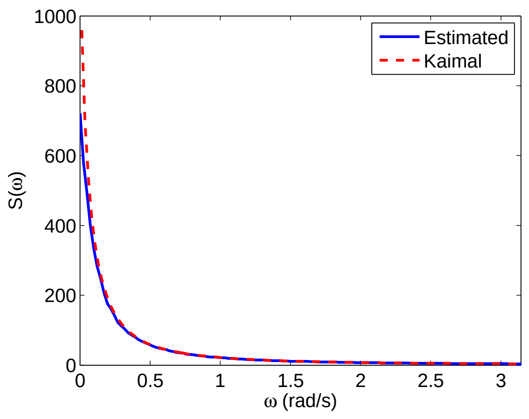
where $\mathrm{FFT}(\cdot)$ is using the notation defined earlier, $\mathbf{X}^{(j)}$ is the vector such that $X_k^{(j)} = X(t_k, x_j)$, and $M_\omega = 2N_\omega$ is the number of points used for the FFT as described above.

The estimated auto-spectrum matches the target very nicely. Aside from very close to $\omega = 0$, the two are indistinguishable. Thus, as was shown in Eqn. (8.10), the auto-spectral characteristics are retained in the wave model proposed.

Additionally, the cross-spectra are estimated for $S_{j(j+50)}(\omega)$, $S_{j(j+100)}(\omega)$, and $S_{j(j+200)}(\omega)$. In the results shown here, these indices correspond to $x_{526} = 25.6598$ m, $x_{576} = 28.1036$ m, $x_{676} = 32.9912$ m.



(a) Kaimal vs. estimated spectra



(b) Zoom-in detail

Figure 8.3: Comparison of the estimated spectra from 5000 samples generated from the wave model vs. the prescribed Kaimal auto-spectrum

Therefore, the separation distances are: $\xi_{476,526} = 2.4438$ m, $\xi_{476,576} = 4.8876$ m, $\xi_{476,676} = 9.7752$ m. For sample function slices $X(t, x_j)$ and $X(t, x_k)$, the cross-spectra are estimated as:

$$S_{jk}(\omega) = \frac{\int_0^T X(t, x_j)\, e^{i\omega t}\, \mathrm{d}t \int_0^T X(t, x_k)\, e^{-i\omega t}\, \mathrm{d}t}{2\pi T} \tag{8.17a}$$

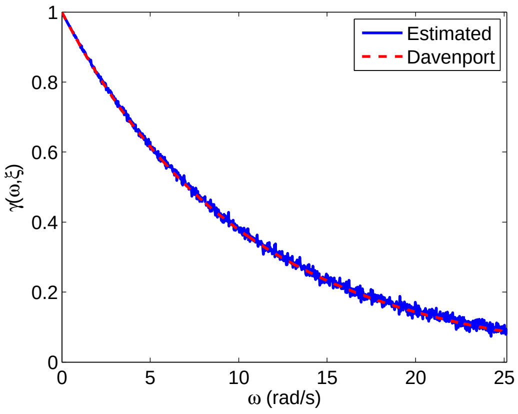
$$= \frac{\mathrm{IFFT}\left(\mathbf{X}^{(j)}\right) \mathrm{FFT}\left(\mathbf{X}^{(k)}\right)}{\Delta\omega M_\omega^2} \tag{8.17b}$$

From these cross-spectra, the coherence functions are estimated as:

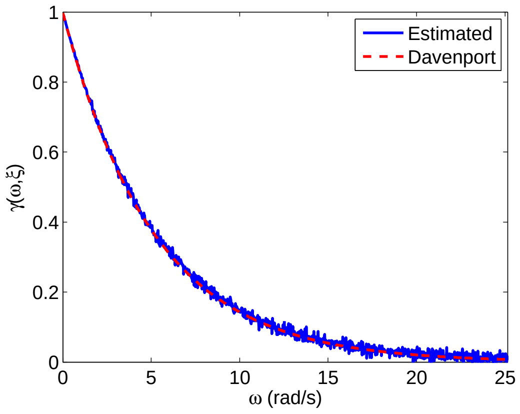
$$\gamma(\omega, \xi_{jk}) = \frac{S_{jk}(\omega)}{S_{jj}(\omega)\, S_{kk}(\omega)} \tag{8.18}$$

These three estimated coherence functions are plotted against the target Davenport coherence models for those specified separation distances in Figure 8.4. For all three, there is clear agreement between the target and the estimated. The first two are in fact perfectly aligned with the target. The third, for $\xi_{476,676} = 9.7752$ shown in Figure 8.4(c), though, is quite noisy. This is inescapable, unfortunately. Though, it is not an issue with the simulated sample, but rather the estimation of the coherence and cross-spectra. It is well known that the coherence is very difficult to capture, and is very sensitive [100], especially as the separation distance grows. These results are therefore acceptable.

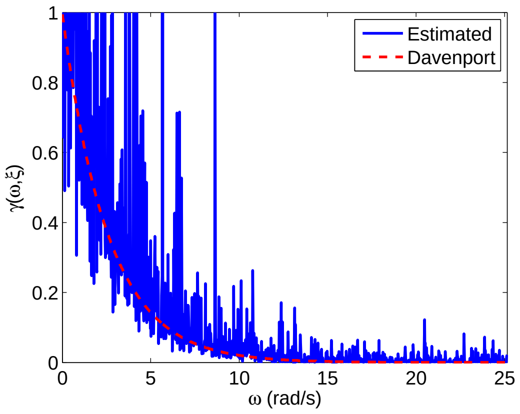
Finally, a sample realization of the simulated wave is shown in Figure 8.5. Figure 8.5(a) shows the entire wave sample. Figure 8.5(b) then shows sample time histories taken at $x = 0$ m, $x = 800$ m, and $x = 1600$ m. Lastly, Figure 8.5(c) shows zoom-in time histories at $x = 23.2160$ m and $x = 25.6598$ m, over-layed on top of each other, to show the spatial variation between neighboring points in space.



(a) $\gamma\,(\omega, \xi = 2.4438)$

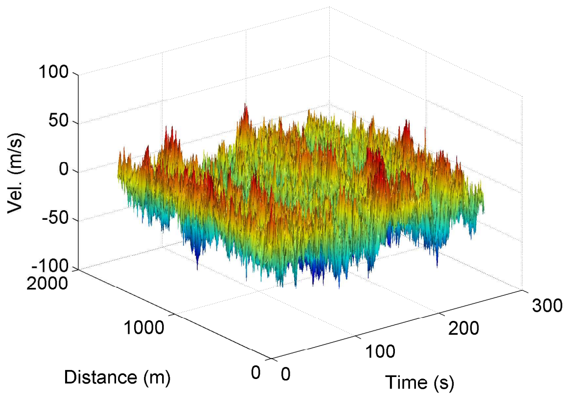


(b) $\gamma\,(\omega, \xi = 4.8876)$

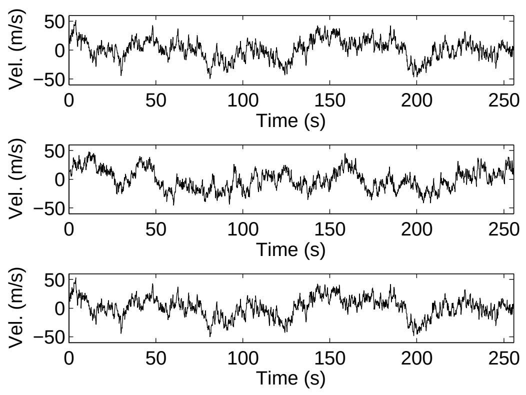


(c) $\gamma\,(\omega, \xi = 9.7752)$

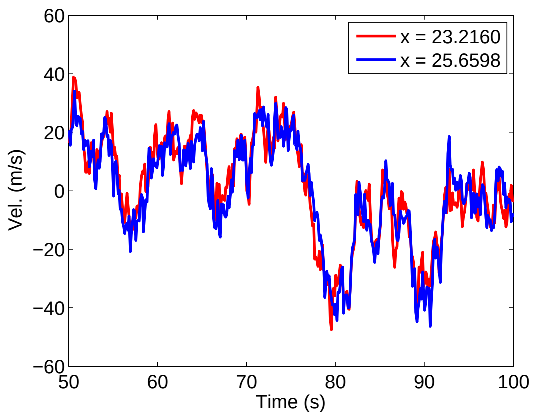
Figure 8.4: Comparison of the estimated coherence functions from 5000 samples generated from the wave model vs. the prescribed Davenport coherence function



(a) Generated sample wave



(b) Generated time histories. Top: $x = 0$ m. Middle: $x = 800$ m. Bottom: $x = 1600$ m.



(c) Spatial variation between neighboring points

Figure 8.5: Generated sample realization of wind velocities modeled as a stochastic wave

Chapter 9

Conclusions

In conclusion, the main contributions of the work presented in this thesis are summarized below in Section 9.1. Some plans for future work are discussed in Section 9.2.

9.1 Main contributions

9.1.1 Part I: Modeling heterogeneous fields

In Part I of this thesis, modeling heterogeneous materials with arbitrarily shaped inclusions within the eXtended Finite Element Method was discussed. A novel enrichment function formulation was presented, which models arbitrarily shaped inclusions and holes using cubic spline interpolation of discrete boundary points. The formulation was shown to be accurate and efficient for modeling arbitrary inclusions. The formulation was then applied to nonlinear finite deformation problems in hyperelasticity. The ability of the proposed formulation to track the evolving boundary geometry through large deformation was demonstrated.

Applications for the proposed formulation are numerous. Consider, for example, problems in which internal geometry changes with deformation, like the example problems of Chapter 4. The spline method presented allows for these problems to be modeled without the need for complicated meshes, and without the need for re-meshing as the geometry deforms. Applications in which re-meshing is to be avoided are the biggest applications for this method, and the XFEM, such as optimization problems, or Monte-Carlo methods. In applications such as these, the problem will need to be solved repeatedly, for differing geometries, and the proposed method will allow for the same mesh (a regular grid) to be used for each solution.

9.1.2 Part II: Simulating random procresses & fields

In part II of this thesis, simulation of random processes and fields was discussed. Chapter 6 first discussed the simulation of non-stationary processes and inhomogeneous fields. Namely, the inversion of the evolutionary spectrum from non-stationary auto-correlation functions was examined. The existence and uniqueness of such an inversion was shown to, likely, exist. This work was among the first such indications of an existence of the inverse. An efficient methodology for computing this inverse was then presented. The need for this inversion for simulating non-stationary processes was discussed. More importantly, the necessity of this inversion within the framework of a translation based simulation of non-stationary and non-Gaussian processes was shown in Chapter 7. In this chapter, an algorithm for determining the underlying Gaussian evolutionary spectrum for a prescribed non-stationary and non-Gaussian process and distribution was proposed. Aside from one approximate technique, the work presented in this chapter was the first algorithm for simulating non-stationary and non-Gaussian stochastic processes in the spectral representation method. It is the first robust and theoretically exact method.

Finally, in Chapter 8, a novel approach to simulating wind velocities along long-span structures was presented. In contrast to the existing methodologies of modeling these wind velocities as stochastic vector processes, it was proposed to model them as a stochastic wave which is continuous in space and time. This allows the simulation of wind velocities at a virtually infinite number of points in space, a feat which was impossible up to this point. Furthermore, the method was shown to be accurate, robust, and more efficient than previous methodologies.

9.2 Future work

There is a wealth of potential future work, either in extensions of the formulations and methodologies presented in this thesis, or in applications to particular studies. Some examples are listed below for each part of the thesis.

9.2.1 Part I: Modeling heterogeneous fields

The spline based enrichment function presented in this part of the thesis has several interesting applications, and room for future work. For one, extension to 3-dimensions is currently being explored. Fortunately, the parametric spline formulation that was presented lends itself very naturally to extensions to additional dimensions. The formulation was shown to work well in finite deformation problems in which internal geometries deform significantly - this could then be extended to additional constitutive laws, such as plasticity, for example.

In the author's opinion, one of the most interesting applications is in optimization problems. Be it for material design, flaw detection, or any other problem in which the internal geometry needs to be optimized given some error function. The spline formulation was shown to be very accurate for very few number of discrete boundary points - this can be likened to very few parameters in an optimization scheme. By minimizing the number of parameters necessary to solve for, the optimization techniques should prove to be more efficient.

Another particular route, which the author wishes to explore further, is the tie between XFEM and uncertainty quantification, namely Monte-Carlo or other random sampling techniques. The alleviation of the need to generate complicated meshes at each independent solution in the stochastic space makes XFEM a very attractive "black box" solver for these methods.

9.2.2 Part II: Simulating random procresses & fields

Similarly, there is a plethora of future work in applications and extensions to the work presented in Part II. First and foremost, a rigorous mathematical proof of existence and uniqueness for the inversion of the evolutionary spectrum from a prescribed auto-correlation remains to be developed, if even possible. This should be explored further, without question. Additionally, perhaps a closed form analytical inversion is possible in the future. If not, then more efficient numerical inversion techniques may be possible - or improvements to the pseudo-spectrum proposed in Chapter 6.

Improvements to the methodologies presented in Chapter 6 will automatically improve the efficiency and/or accuracy of the non-stationary and non-Gaussian methodology presented in Chapter 7. Furthermore, there are several interesting applications that the author hopes to explore. Namely, the simulation and characterization of functionally graded materials. These functionally graded materials are at the forefront of materials science, currently, and solid probabilistic understanding of their characteristics will strengthen their design.

Finally, the methodology presented in Chapter 8 for simulating wind velocities for long-span structures could potentially be extended to other cases. The formulation in this thesis was for one-dimensional, homogeneous in space, and stationary in time, velocity fields. Alternatively, each of these 'categories' could be extended to multi-dimensionality, inhomogeneity, and non-stationarity, respectively. Consideration of these additional complexities in the velocity fields would make for stronger simulation and understanding of the physical processes.

Bibliography

[1] N. Sukumar, D. L. Chopp, N. Moës, and T. Belytschko. Modeling holes and inclusions by level sets in the extended finite-element method. *Computer Methods in Applied Mechanics and Engineering*, 190 (46-47):6183–6200, 2001.

[2] B. Hiriyur, H. Waisman, and G. Deodatis. Uncertainty quantification in homogenization of heterogeneous microstructures modeled by XFEM. *International Journal for Numerical Methods in Engineering*, 88(3):257–278, October 2011.

[3] T. Belytschko and T. Black. Elastic crack growth in finite elements with minimal remeshing. *International Journal for Numerical Methods in Engineering*, 45(July 1998):601–620, 1999.

[4] N. Moës, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, 46(1):131–150, 1999.

[5] N. Moës, M. Cloirec, P. Cartraud, and J.-F. Remacle. A computational approach to handle complex microstructure geometries. *Computer Methods in Applied Mechanics and Engineering*, 192(28-30): 3163–3177, July 2003.

[6] T.-P. Fries. A corrected XFEM approximation without problems in blending elements. *International Journal for Numerical Methods in Engineering*, 75(5):503–532, 2008.

[7] T.-P. Fries and T. Belytschko. The extended/generalized finite element method: An overview of the method and its applications. *International Journal for Numerical Methods in Engineering*, 84(3): 253–304, August 2010.

[8] H. Waisman and T. Belytschko. Parametric enrichment adaptivity by the extended finite element method. *International Journal for Numerical Methods in Engineering*, 73(12):1671–1692, March 2008.

[9] A. Menk and S. P. A. Bordas. Numerically determined enrichment functions for the extended finite element method and applications to bi-material anisotropic fracture and polycrystals. *International Journal for Numerical Methods in Engineering*, 83(7):805–828, 2010.

[10] C. W. Liu and E. Taciroglu. Enriched reproducing kernel particle method for piezoelectric structures with arbitrary interfaces. *International Journal for Numerical Methods in Engineering*, 67(11):1565–1586, September 2006.

[11] C. W. Liu and E. Taciroglu. Shape optimization of piezoelectric devices using an enriched meshfree method. *International Journal for Numerical Methods in Engineering*, 78(2):151–171, April 2009.

[12] E. Chahine, P. Laborde, and Y. Renard. Spider XFEM, an extended finite element variant for partially unknown crack-tip displacement. *Revue européenne de mécanique numérique*, 17(5-6-7):625–636, October 2008.

[13] W. Aquino, J. C. Brigham, C. J. Earls, and N. Sukumar. Generalized finite element method using proper orthogonal decomposition. *International Journal for Numerical Methods in Engineering*, 79(7): 887–906, August 2009.

[14] S. Abbas, A. Alizada, and T.-p. Fries. The XFEM for high-gradient solutions in convection-dominated problems. *International Journal for Numerical Methods in Engineering*, 82(8):1044–1072, December 2009.

[15] M. Kästner, S. Müller, J. Goldmann, C. Spieler, J. Brummund, and V. Ulbricht. Higher-order extended FEM for weak discontinuities - level set representation, quadrature and application to magneto-mechanical problems. *International Journal for Numerical Methods in Engineering*, December 2012.

[16] S. Osher and J. a. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, November 1988.

[17] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, 93(4):1591–1595, February 1996.

[18] J. A. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Cambridge University Press, 1999.

[19] M. Stolarska, D. L. Chopp, N. Moës, and T. Belytschko. Modelling crack growth by level sets in the extended finite element method. *International Journal for Numerical Methods in Engineering*, 51(8): 943–960, July 2001.

[20] N. Sukumar, D. Chopp, and B. Moran. Extended finite element method and fast marching method for three-dimensional fatigue crack propagation. *Engineering Fracture Mechanics*, 70(1):29–48, January 2003.

[21] R. Gracie, G. Ventura, and T. Belytschko. A new fast finite element method for dislocations based on interior discontinuities. *International Journal for Numerical Methods in Engineering*, 69(2):423–441, January 2007.

[22] V. S. Rao. On modelling thermal oxidation of Silicon II: numerical aspects. *International Journal for Numerical Methods in Engineering*, 47:359–377, 2000.

[23] R. Duddu, S. Bordas, D. Chopp, and B. Moran. A combined extended finite element and level set method for biofilm growth. *International Journal for Numerical Methods in Engineering*, 74(5):848–870, April 2008.

[24] R. Duddu, D. L. Chopp, P. Voorhees, and B. Moran. Diffusional evolution of precipitates in elastic media using the extended finite element and the level set methods. *Journal of Computational Physics*, 230(4):1249–1264, February 2011.

[25] D. Rabinovich, D. Givoli, and S. Vigdergauz. XFEM-based crack detection scheme using a genetic algorithm. *International Journal for Numerical Methods in Engineering*, 71(9):1051–1080, August 2007.

[26] H. Waisman, E. N. Chatzi, and A. W. Smyth. Detection and quantification of flaws in structures by the extended finite element method and genetic algorithms. *International Journal for Numerical Methods in Engineering*, 82(3):303–328, 2009.

[27] E. N. Chatzi, B. Hiriyur, H. Waisman, and A. W. Smyth. Experimental application and enhancement of the XFEMGA algorithm for the detection of flaws in structures. *Computers & Structures*, 89(7-8): 556–570, April 2011.

[28] J. Fish and T. Belytschko. *A First Course in Finite Elements*. John Wiley & Sons, Ltd., 2007.

[29] Y. Abdelaziz and A. Hamouine. A survey of the extended finite element. *Computers & Structures*, 86 (11-12):1141–1151, June 2008.

[30] T. Belytschko, R. Gracie, and G. Ventura. A review of extended/generalized finite element methods for material modeling. *Modelling and Simulation in Materials Science and Engineering*, 17(4):043001, June 2009.

[31] S. P. A. Bordas, P. V. Nguyen, C. Dunant, A. Guidoum, and H. Nguyen-Dang. An extended finite element library. *International Journal for Numerical Methods in Engineering*, 71(6):703–732, August 2007.

[32] S. Mohammadi. *Extended Finite Element Method for Fracture Analysis of Structures*. Wiley-Blackwell, 2008.

[33] T.-P. Fries and T. Belytschko. The intrinsic XFEM: a method for arbitrary discontinuities without additional unknowns. *International Journal for Numerical Methods in Engineering*, 68(13):1358–1385, December 2006.

[34] T.-P. Fries and T. Belytschko. New shape functions for arbitrary discontinuities without additional unknowns. In *Meshfree methods for partial differential equations III*, pages 87–103. Springer, 2007.

[35] U. Ascher and C. Greif. *A First Course in Numerical Methods*. SIAM, 2011.

[36] T. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39-41): 4135–4195, October 2005.

[37] J. Cottrell, T. Hughes, and Y. Bazilevs. *Isogeometric Analysis. Toward Integration of CAD and FEA*. Wiley, 2009.

[38] C. de Boor. *A practical guide to splines*. Springer-Verlag, 1978.

[39] G. D. Knott. *Interpolating cubic splines*. Birkh\"{a}user, 1999.

[40] E. V. Shikin and A. I. Plis. *Handbook on Splines for the User*. CRC, 1995.

[41] D. L. Chopp. Numerical Methods for Moving Interfaces. In *Lecture Notes*, 2011.

[42] MATLAB. *version 8.0.0.783 (R2012b)*. The MathWorks Inc., Natick, Massachusetts, 2012.

[43] S. Natarajan, D. Mahapatra, and S. P. A. Bordas. Integrating strong and weak discontinuities without integration subcells and example applications in an XFEM/GFEM framework. *International Journal for Numerical Methods in Engineering*, 83(3):269–294, 2010.

[44] T. Elguedj, A. Gravouil, and A. Combescure. A mixed augmented Lagrangian-extended finite element method for modelling elasticplastic fatigue crack growth with unilateral contact. *International Journal for Numerical Methods in Engineering*, 71(13):1569–1597, September 2007.

[45] R. Gracie, J. Oswald, and T. Belytschko. On a new extended finite element method for dislocations: Core enrichment and nonlinear formulation. *Journal of the Mechanics and Physics of Solids*, 56(1): 200–214, January 2008.

[46] G. Legrain, N. Moës, and E. Verron. Fracture with large deformation using X-FEM. In *European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS 2004*, 2004.

[47] G. Legrain, N. Moës, and E. Verron. Stress analysis around crack tips in finite strain problems using the eXtended finite element method. *International Journal for Numerical Methods in Engineering*, 63 (2):290–314, May 2005.

[48] S. Loehnert, D. S. Mueller-Hoeppe, and P. Wriggers. 3D corrected XFEM approach and extension to finite deformation theory. *International Journal for Numerical Methods in Engineering*, 86(4-5): 431–452, April 2011.

[49] I. Nistor, M. L. E. Guiton, P. Massin, N. Moës, and S. Géniaut. An X-FEM approach for large sliding contact along discontinuities. *International Journal for Numerical Methods in Engineering*, 78(12): 1407–1435, June 2009.

[50] C. Dubois, S. Le Corre, M. Zarroug, P. Rozycki, and N. Moës. Impact on highly compressible media in explicit dynamics using the X-FEM. *Computational Mechanics*, 46(2):329–348, May 2010.

[51] P. Rozycki, N. Moës, E. Bechet, and C. Dubois. X-FEM explicit dynamics for constant strain elements to alleviate mesh constraints on internal or external boundaries. *Computer Methods in Applied Mechanics and Engineering*, 197(5):349–363, January 2008.

[52] T. Hettich and E. Ramm. Interface material failure modeled by the extended finite-element method and level sets. *Computer Methods in Applied Mechanics and Engineering*, 195(37-40):4753–4767, July 2006.

[53] T. Hettich, A. Hund, and E. Ramm. Modeling of failure in composites by X-FEM and level sets within a multiscale framework. *Computer Methods in Applied Mechanics and Engineering*, 197(5):414–424, January 2008.

[54] A. Khoei, A. Shamloo, and A. Azami. Extended finite element method in plasticity forming of powder compaction with contact friction. *International Journal of Solids and Structures*, 43(18-19):5421–5448, September 2006.

[55] M. Anahid and A. Khoei. New development in extended finite element modeling of large elasto-plastic deformations. *International Journal for Numerical Methods in Engineering*, 75(10):1133–1171, September 2008.

[56] M. Kästner, G. Haasemann, and V. Ulbricht. Multiscale XFEM-modelling and simulation of the inelastic material behaviour of textile-reinforced polymers. *International Journal for Numerical Methods in Engineering*, 86(4-5):477–498, April 2011.

[57] A. Khoei, S. Biabanaki, and M. Anahid. Extended finite element method for three-dimensional large plasticity deformations on arbitrary interfaces. *Computer Methods in Applied Mechanics and Engineering*, 197(9-12):1100–1114, February 2008.

[58] A. Khoei, S. Biabanaki, and M. Anahid. A Lagrangian-extended finite-element method in modeling large-plasticity deformations and contact problems. *International Journal of Mechanical Sciences*, 51 (5):384–401, May 2009.

[59] T. Belytschko, W. Liu, and B. Moran. *Nonlinear finite elements for continua and structures*, volume 36. Wiley, 2000.

[60] P. Wriggers. *Nonlinear finite element methods*, volume 9. Springer, January 2008.

[61] L. E. Malvern. *Introduction to the mechanics of a continuous medium.* Prentice-Hall, Englewood Cliffs, N.J., 1969.

[62] C. Daux, N. Moës, J. Dolbow, N. Sukumar, and T. Belytschko. Arbitrary branched and intersecting cracks with the extended nite element method. *International Journal for Numerical Methods in Engineering*, 48:1741–1760, 2000.

[63] M. Shinozuka and C. Jan. Digital simulation of random processes and its applications. *Journal of Sound and Vibrationon*, 25(1):111–128, 1972.

[64] W. Gersch and J. Yonemoto. Synthesis of multivariate random vibration systems: a two-stage least squares AR-MA model approach. *Journal of Sound and Vibration*, 52(4):553–565, 1977.

[65] P. Spanos and J. Hansen. Linear prediction theory for digital simulation of sea waves. *Journal of Energy Resources Technology*, 103:243–249, 1981.

[66] F. Kozin and F. Nakajima. The order determination problem for linear time-varying AR models. *IEEE Transactions on Automatic Control*, AC-25(2):250–257, 1980.

[67] P. Spanos. ARMA algorithms for ocean wave modeling. *Journal of Energy Resources Technology*, 105: 300–309, 1983.

[68] P. Spanos and G. Solomos. Markov approximation to transient vibration. *Journal of Engineering Mechanics*, 109(4):1134–1150, 1983.

[69] E. Samaras, M. Shinzuka, and A. Tsurui. ARMA representation of random processes. *Journal of Engineering Mechanics*, 111(3):449–461, 1985.

[70] K. A. R. I. Karhunen. Uber lineare Methoden in dew Wahrscheinlichkeitsrechnung. *Ann. Acad. Sci. Fennicae.*, 37:3–79, 1947.

[71] M. Kac and A. Siegert. An explicit representation of a stationary Gaussian process. *The Annals of Mathematical Statistics*, 18(3):438–442, 1947.

[72] M. Loeve. Fonctions aléatoires du second ordre, A note in P. *Levi's Processus stochastiques et mouvement Brownien, Gauthier-Villars, Paris*, 1948.

[73] R. G. Ghanem and P. D. Spanos. *Stochastic Finite Elements: A Spectral Approach.* Dover, revised edition, 2012.

[74] N. Wiener. Generalized harmonic analysis. *Acta Mathematica*, 55:117–258, 1930.

[75] A. Khintchine. Korrelationstheorie der stationären stochastischen Prozesse. *Mathematische Annalen*, 109(1):604–615, December 1934.

[76] M. Priestley. *Spectral Analysis and Times Series Volume 1: Univariate Series.* Academic Press, 1981.

[77] A. Papoulis. *Probability, Random Variables, and Stochastic Processes.* WCB/McGraw-Hill, 3rd edition, 1991.

[78] M. Shinozuka and G. Deodatis. Simulation of stochastic processes by spectral representation. *Applied Mechanics Reviews*, 44(4), 1991.

[79] A. Zerva. Seismic ground motion simulations from a class of spatial variability models. *Earthquake Engineering & Structural Dynamics*, 21:351–361, 1992.

[80] A. Zerva and V. Zervas. Spatial variation of seismic ground motions: An overview. *Applied Mechanics Reviews*, 55(3):271, 2002.

[81] J.-N. Yang. Simulation of random envelope processes. *Journal of Sound and Vibration*, 21(1):73–85, March 1972.

[82] G. Deodatis. Non-stationary stochastic vector processes: seismic ground motion applications. *Probabilistic Engineering Mechanics*, 11(3):149–167, July 1996.

[83] S. Amada, T. Munekata, Y. Nagase, Y. Ichikawa, A. Kirigai, and Y. Zhifei. The Mechanical Structures of Bamboos in Viewpoint of Functionally Gradient and Composite Materials. *Journal of Composite Materials*, 30(7):800–819, May 1996.

[84] S. Amada, Y. Ichikawa, T. Munekata, Y. Nagase, and H. Shimizu. Fiber texture and mechanical graded structure of bamboo. *Composites Part B: Engineering*, 28(1-2):13–20, January 1997.

[85] C. H. Page. Instantaneous Power Spectra. *Journal of Applied Physics*, 23(1):103–106, 1952.

[86] W. Martin and P. Flandrin. Wigner-Ville spectral analysis of nonstationary processes. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(6):1461–1470, December 1985.

[87] J. Hammond and R. Harrison. Wigner-Ville and evolutionary spectra for covariance equivalent nonstationary random processes. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 10:1025–1028, 1985.

[88] J. Hammond, J. Lee, and R. Harrison. The relationship between Wigner-Ville and evolutionary spectra for frequency modulated random processes. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 2327–2330, 1986.

[89] M. Priestley. *Non-linear and Non-stationary Time Series Analysis.* Academic Press, 1988.

[90] J. Liang, S. R. Chaudhuri, and M. Shinozuka. Simulation of Nonstationary Stochastic Processes by Spectral Representation. *Journal of Engineering Mechanics*, 133(6):616–627, June 2007.

[91] Y. Li and A. Kareem. Simulation of Multivariate Nonstationary Random Processes by FFT. *Journal of Engineering Mechanics*, 117(5):1037–1058, May 1991.

[92] M. Grigoriu. *Applied non-Gaussian processes: examples, theory, simulation, linear random vibration, and MATLAB solutions.* PTR Prentice Hall, Englewood Cliffs, N.J., 1995.

[93] F. Yamazaki and M. Shinozuka. Digital generation of non-Gaussian stochastic fields. *Journal of Engineering Mechanics*, 114(7):1183–1197, 1988.

[94] G. Deodatis and R. Micaletti. Simulation of highly skewed non-Gaussian stochastic processes. *Journal of Engineering Mechanics*, 127(12):1284–1295, 2001.

[95] Y. Shi, G. Deodatis, and S. Koutsourelakis. A novel approach for simulation of non-Gaussian fields: Application in estimating wire strengths from experimental data. In *9th ASCE Joint Special Conference on Probabilistic Mechanics and Structural Reliability*, Albuquerque, New Mexico, July 2004.

[96] P. Bocchini and G. Deodatis. Critical review and latest developments of a class of simulation algorithms for strongly non-Gaussian random fields. *Probabilistic Engineering Mechanics*, 23(4):393–407, October 2008.

[97] M. Shields, G. Deodatis, and P. Bocchini. A simple and efficient methodology to approximate a general non-Gaussian stationary stochastic process by a translation process. *Probabilistic Engineering Mechanics*, 26(4):511–519, October 2011.

[98] F. Ferrante, S. Arwade, and L. Grahambrady. A translation model for non-stationary, non-Gaussian random processes. *Probabilistic Engineering Mechanics*, 20(3):215–228, July 2005.

[99] M. D. Shields and G. Deodatis. Estimation of evolutionary spectra for simulation of non-stationary and non-Gaussian stochastic processes. *Computers & Structures*, In Press, 2012.

[100] A. Zerva. *Spatial Variation of Seismic Ground Motions: Modeling and Engineering Applications.* CRC Press, Taylor & Francis Group, Boca Raton, FL, 2009.

[101] J. C. Kaimal, J. C. Wyngaard, Y. Izumi, and O. R. Coté. Spectral characteristics of surface-layer turbulence. *Quarterly Journal of the Royal Meteorological Society*, 98(417):563–589, July 1972.

[102] A. Davenport. The dependence of wind loads on meteorological parameters. *Proceedings of the International Research Seminar on Wind Effects on Buildings and Structures*, 1:19–82, 1967.

[103] K. Kanai. Seismic-empirical formula for the seismic characteristics of the ground. *Bulletin of Earthquake Research Institute*, 35:309–325, 1957.

[104] H. Tajimi. A statistical method of determining the maximum response of a building structure during an earthquake. In *Proceedings of the 2nd world conference on earthquake engineering*, pages 781–798, 1960.

[105] R. Clough and J. Penzien. *Dynamics of Structures.* McGraw Hill, 1975.

[106] R. Harichandran and E. Vanmarcke. Stochastic variation of earthquake ground motion in space and time. *Journal of Engineering Mechanics*, 112(2):154–174, 1986.

[107] J. Luco and H. Wong. Response of a rigid foundation to a spatially random ground motion. *Earthquake Engineering & Structural Dynamics*, 14:891–908, 1986.

[108] G. Deodatis. Simulation of ergodic multivariate stochastic processes. *Journal of Engineering Mechanics*, 122(8):778–787, 1996.

[109] M. Shinozuka, M. Kamata, and C.-B. Yun. Simulation of earthquake ground motion as multi-variate stochastic process. Technical report, Department of Civil Engineering and Operations Research, Princeton University, 1989.

[110] M. Shinozuka and G. Deodatis. Simulation of multi-dimensional Gaussian stochastic fields by spectral representation. *Applied Mechanics Reviews*, 49(1):29–53, 1996.

[111] G. Deodatis and M. Shinozuka. Simulation of seismic ground motion using stochastic waves. *Journal of engineering mechanics*, 115(12):2723–2737, 1989.

[112] M. Shinozuka and G. Deodatis. Stochastic wave models for stationary and homogeneous seismic ground motion. *Structural Safety*, 10:235–246, 1991.

[113] Pacific-Earthquake-Engineering-Research-Center. PEER Ground Motion Database, 2013. URL `http://peer.berkeley.edu/products/strong_ground_motion_db.html`.

[114] M. D. Shields. *Simulation of Stochastic Processes: Applications in Civil Engineering.* PhD thesis, Columbia University, New York, NY, 2010.

[115] P. Jennings, G. Housner, and N. Tsai. Simulated earthquake motions. Technical report, Earthquake Engineering Research Laboratory, California Institute of Technology, Pasadena, California, 1968.

[116] B. R. Ellingwood and M. E. Batts. Characterization of earthquake forces for probability-based design of nuclear structures. Technical report, Center for Building Technology, National Bureau of Standards, Washington, DC (USA), 1982.

[117] A. Savitzky and M. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[118] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in Fortran The Art of Scientific Computing.* Cambridge University Press, 2nd edition, 1992.

[119] J. Burkardt. PROB, 2012. URL `http://people.sc.fsu.edu/~jburkardt/f_src/prob/prob.html`.

[120] K. Y. Billal and R. H. Scanlan. Resonance, Tacoma Narrows bridge failure, and undergraduate physics textbooks. *American Journal of Physics*, 59(2):118–124, 1991.

[121] A. Davenport. The application of statistical concepts to the wind loading of structures. *ICE Proceedings*, (6480):449–472, 1961.

[122] M. Shinozuka, C.-B. Yun, and H. Seya. Stochastic methods in wind engineering. *Journal of Wind Engineering and Industrial Aerodynamics*, 36:829–843, January 1990.

[123] M. Di Paola. Digital simulation of wind field velocity. *Journal of Wind Engineering and Industrial Aerodynamics*, 74-76:91–109, April 1998.

[124] M. Di Paola and I. Gullo. Digital generation of multivariate wind field processes. *Probabilistic Engineering Mechanics*, 16(1):1–10, January 2001.

[125] W. Yang, T. Chang, and C. Chang. An efficient wind field simulation technique for bridges. *Journal of Wind Engineering and Industrial Aerodynamics*, 67-68:697–708, April 1997.

[126] Y. Cao, H. Xiang, and Y. Zhou. Simulation of stochastic wind velocity field on long-span bridges. *Journal of Engineering Mechanics*, 126(1):1–6, 2000.

[127] Y. Li, H. Liao, and S. Qiang. Simplifying the simulation of stochastic wind velocity fields for long cable-stayed bridges. *Computers & Structures*, 82(20-21):1591–1598, August 2004.

[128] Q. Ding, L. Zhu, and H. Xiang. An efficient ergodic simulation of multivariate stochastic processes with spectral representation. *Probabilistic Engineering Mechanics*, 26(2):350–356, April 2011.

[129] Y. Gao, Y. Wu, D. Li, H. Liu, and N. Zhang. An improved approximation for the spectral representation method in the simulation of spatially varying ground motions. *Probabilistic Engineering Mechanics*, 29:7–15, July 2012.

[130] E. Simiu and R. H. Scanlan. *Wind Effects on Structures: An Introduction to Wind Engineering*. John Wiley & Sons, Inc., 1978.

[131] R. Haberman. *Applied Partial Differential Equations with Fourier Series and Boundary Value Problems*. Pearson Prentice Hall, Upper Saddle River, New Jersey, 4th edition, 2004.

[132] D. Newland. *An Introduction to Random Vibrations, Spectral & Wavelet Analysis*. Longman Scientific & Technical, Essex, England, 3rd edition, 1993.

Appendix A

Closest point on spline curve derivation

The parametric spline curve is defined as:

$$\mathbf{P}(t) = \begin{bmatrix} x_\Gamma(t) \\ y_\Gamma(t) \end{bmatrix} = \begin{bmatrix} a_x t^3 + b_x t^2 + c_x t + d_x \\ a_y t^3 + b_y t^2 + c_y t + d_y \end{bmatrix} \tag{A.1}$$

The distance from any point (x, y) to a point on the curve, as a function of the curve parameter t, is given as:

$$d(t) = \sqrt{(x - x_\Gamma(t))^2 + (y - y_\Gamma(t))^2} \tag{A.2}$$

To find the shortest distance from the point to the curve, we must find the closest point on the curve. In other words, we must find t which minimizes $d(t)$. We start by substituting the expressions for x_Γ and y_Γ into the expression for d. This gives:

$$\begin{aligned} d^2 \;=\; & x^2 - 2xd_x + a_x^2 t^6 + b_x^2 t^4 + c_x^2 t^2 + 2a_x t^5 b_x + \\ & 2a_x t^4 c_x + 2b_x t^3 c_x - 2xa_x t^3 - 2xb_x t^2 - \\ & 2xc_x t + 2a_x t^3 d_x + 2b_x t^2 d_x + 2c_x t d_x + \\ & 2a_y t^5 b_y + 2a_y t^4 c_y + 2b_y t^3 c_y - 2ya_y t^3 - 2yb_y t^2 - \\ & 2yc_y t + 2a_y t^3 d_y + 2b_y t^2 d_y + 2c_y t d_y + d_x^2 + y^2 - \\ & 2yd_y + a_y^2 t^6 + b_y^2 t^4 + c_y^2 t^2 + d_y^2 \end{aligned} \tag{A.3}$$

In order to find the minimum, we differentiate with respect to t and set to zero. Differentiating the above and setting it equal to zero yields the following quintic equation:

$$\begin{aligned} 0 \quad = \quad & \left(3a_x^2 + 3a_y^2\right) t^5 + \left(5a_y b_y + 5b_x a_x\right) t^4 + \\ & \left(2b_x^2 + 2b_y^2 + 4a_y c_y + 4c_x a_x\right) t^3 + \\ & \left(-3y a_y - 3a_x x + 3b_y c_y + 3b_x c_x + 3a_y d_y + 3d_x a_x\right) t^2 + \\ & \left(-2y b_y + c_y^2 + 2b_y d_y - 2x b_x + c_x^2 + 2b_x d_x\right) t - \\ & y c_y + c_y d_y - x c_x + c_x d_x \end{aligned} \tag{A.4}$$

or,

$$0 = C_1 t^5 + C_2 t^4 + C_3 t^3 + C_4 t^2 + C_5 t + C_6 \tag{A.5}$$

where:

$$\begin{aligned} C_1 &= 3a_x^2 + 3a_y^2 \\ C_2 &= 5a_y b_y + 5b_x a_x \\ C_3 &= 2b_x^2 + 2b_y^2 + 4a_y c_y + 4c_x a_x \\ C_4 &= -3y a_y - 3a_x x + 3b_y c_y + 3b_x c_x + 3a_y d_y + 3d_x a_x \\ C_5 &= -2y b_y + c_y^2 + 2b_y d_y - 2x b_x + c_x^2 + 2b_x d_x \\ C_6 &= -y c_y + c_y d_y - x c_x + c_x d_x \end{aligned} \tag{A.6}$$

Appendix B

Quadrature for C^0 continuous functions

B.1 Introduction

In this Appendix, a brief convergence analysis is carried out for numerical quadrature rules on a C^0 continuous function in two dimensions. The convergence of these rules are well established for smooth and continuous functions, but not necessarily for the weakly discontinuous functions used in Part I of this thesis. The function, f, examined is the signed distance function for a circular inclusion/hole:

$$f\left(x, y\right) = \left|\sqrt{x^2 + y^2} - r\right| \tag{B.1}$$

where, x and y are the Cartesian coordinates, and r is the radius of the circle. The domain considered is square with sides of length $2a$. The geometry considered is diagrammed in Figure B.1. The objective is to integrate Eqn. (B.1) over the entire domain. An analytical solution to this integral is first found, and then the numerical quadrature rules are used to integrate the function, and the results are compared.

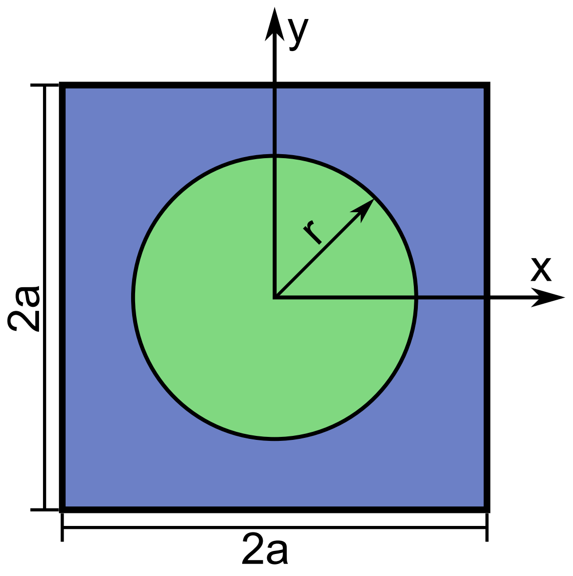


Figure B.1: Quadrature test problem geometry

B.2 Analytical reference solution

Due to the symmetry of the problem, the domain can be reduced to just a single quadrant, and therefore:

$$\int_{-a}^{a}\int_{-a}^{a} f(x,y)\ \mathrm{d}x\mathrm{d}y = 4\times\int_{0}^{a}\int_{0}^{a} f(x,y)\ \mathrm{d}x\mathrm{d}y \tag{B.2}$$

In order to account for the discontinuity at the internal boundary, the integral is subdivided at this boundary. Thus, Eqn. (B.2) is rewritten as:

$$\int_{-a}^{a}\int_{-a}^{a} f(x,y)\ \mathrm{d}x\mathrm{d}y = 4\times\left[\int_{0}^{a}\int_{0}^{\sqrt{r-x^2}}\left(r-\sqrt{x^2+y^2}\right)\ \mathrm{d}x\mathrm{d}y + \int_{0}^{a}\int_{\sqrt{r-x^2}}^{a}\left(\sqrt{x^2+y^2}-r\right)\ \mathrm{d}x\mathrm{d}y\right] \tag{B.3}$$

A switch from the Cartesian coordinates, (x,y), to the polar coordinates, (ρ,θ) is employed. With this coordinate transformation, the integral can be further simplified as:

$$\int_{-a}^{a}\int_{-a}^{a} f(x,y)\ \mathrm{d}x\mathrm{d}y = 4\times\left[\int_{0}^{r}\int_{0}^{\frac{\pi}{2}}(r-\rho)\,\rho\ \mathrm{d}\theta\mathrm{d}\rho + \int_{0}^{\frac{\pi}{4}}\int_{r}^{\frac{a}{\cos\theta}}(\rho-r)\,\rho\ \mathrm{d}\rho\mathrm{d}\theta + \int_{\frac{\pi}{4}}^{\frac{\pi}{2}}\int_{r}^{\frac{a}{\sin\theta}}(\rho-r)\,\rho\ \mathrm{d}\rho\mathrm{d}\theta\right] \tag{B.4}$$

Solving Eqn. (B.4) yields the solution:

$$\int_{-a}^{a}\int_{-a}^{a} f(x,y)\ \mathrm{d}x\mathrm{d}y = 4\times\left[\frac{\pi r^3}{12} - ra^2 + \frac{a^3}{3}\left(\frac{1}{\sqrt{2}} + \tanh^{-1}\left(\tan\left(\frac{\pi}{8}\right)\right)\right) + \frac{a^3}{6}\left(\sqrt{2} + \ln\left(\cot\left(\frac{\pi}{8}\right)\right)\right)\right] \tag{B.5}$$

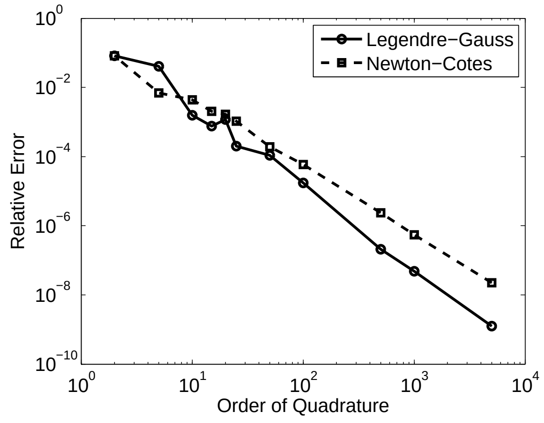


Figure B.2: Quadrature convergence

B.3 Quadrature convergence

Two numerical quadrature rules are now compared with the analytical solution shown in Eqn. (B.5). Namely, Legendre-Gauss and Newton-Cotes quadrature rules are examined - the former being the Gaussian quadrature used in Part I of this thesis. In both cases, the order of the quadrature was varied, and Eqn. (B.1) was integrated over the domain $\{x, y \in [-a, a]\}$ with $a = 1$ and $r = 0.25$. The relative error in the numerical integrals are plotted in Figure B.2. It is clear that both quadrature rules converge despite the function being only C^0 continuous.

Appendix C

MATLAB codes for the Spectral Representation Method, and FFT implementations

C.1 Introduction

In this Appendix, sample MATLAB functions are presented for simulating stochastic processes in the SRM. A brief comparison between direct summation and FFT implementations is examined, as well. This Appendix will only deal with Gaussian processes, as it is only focusing on the simulation of sample realizations in the SRM. Section C.2 will present sample codes for simulation of stationary processes, and examine the efficiency of using the FFT versus direct summation. Section C.3 will present a sample code for generating non-stationary processes. Section C.4 will present a sample code for generating the stochastic waves discussed in Chapter 8.

C.2 Stationary processes

C.2.1 MATLAB codes

Direct summation

Below is a sample MATLAB function which generates sample realizations of a stationary random process through the SRM. This is a direct implementation of Eqn. (5.3). The code takes as input a vector of frequency values, `w`, a vector of time values, `t`, and a vector of SDF values, `S`. The vectors `S` and `w` should be of the same length, and S_j should correspond to $S(\omega_j)$. The function outputs the vector, `y`, representing the sample realization $y_j = y(t_j)$.

```
function [ y ] = statSimDS( w, t, S )

    dw  = w(2)-w(1);          % frequency step size
    N   = length(S);          % number of points in the frequency domain
    phi = rand(1,N)*2*pi;     % N random phase angles between (0,2\pi)

    % Generate sample realization:
    %   Loop over time and sum over frequency
    y = zeros(1,length(t));       % initialize sample function
    for i = 1:length(t)
        y(i) = sum( 2*sqrt( S*dw ) .* cos(w*t(i) + phi) );
    end
end
```

FFT

Below is another sample MATLAB function which generates sample realizations of a stationary random process through the SRM. Though, in this case the FFT formulation of Eqn. (5.8) is used. The input and output variables are the same as in `statSimDS`, above, aside from the fact that the time vector, `t`, is now an output. The time points where FFT outputs values is automatically chosen based off of the input frequency values, hence the time vector is now an output of the function.

```
function [ y , t ] = statSimFFT( w, S )

    % Original domain:
    dw      = w(2)-w(1);          % frequency step size
    N       = length(S);          % number of points in the frequency domain

    % New domain
    M       = 2^nextpow2(2*N);    % M = 2*N, or next power of 2
    S2      = zeros(1,M);         % pad new spectrum with zeros
    S2(1:N) = S;
    t       = 2*pi/dw*linspace(0,1,M+1); t(end)=[]; % time vector
    phi     = rand(1,M)*2*pi;     % M random phase angles between (0,2\pi)

    % Generate sample realization:
    B       = 2*sqrt(S2*dw).*exp(1i*phi);    % B matrix
    f       = M*ifft(B,M);                   % take FFT
```

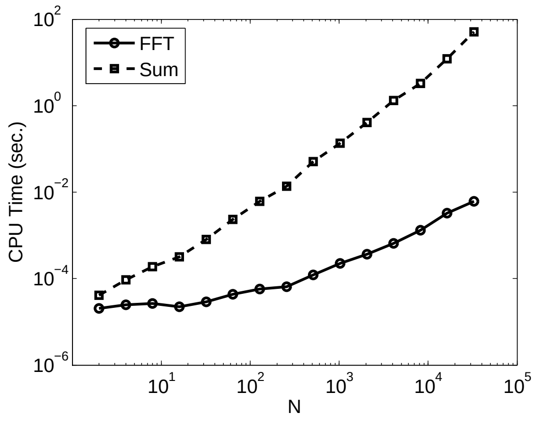


Figure C.1: Comparison of computational efficiency between using the direct implementation of Eqn. (5.3) and the FFT implementation of Eqn. (5.8).

```
    y        = real(f);                          % only want the real part
end
```

C.2.2 Efficiency comparison

The computational efficiency of the two SRM functions just presented, `statSimDS` and `statSimFFT`, are now compared. The number of points in the domain, N, is varied from 2^1 through 2^{15}. Note, that it was ensured that the same number of points were used in both cases, accounting for the padding in the FFT formulation. Thus, for any given N, $M = 2N$ was used in *both* cases, with the padded spectrum. The CPU time was averaged over 100 simulations for each N, and the average CPU time is plotted in Figure C.1. The advantage of using the FFT formulation is quite clear - it is multiple orders of magnitude more efficient. As N increases, this is even more evident.

C.3 Non-stationary processes

Below is a sample MATLAB function which generates sample realizations of a non-stationary random process through the SRM. Unfortunately in this case, there is no exact method which leverages the FFT. Thus, the code presented here is very similar to the stationary code, `statSimDS`, above, and is a direct implementation of Eqn. (5.15). The inputs and outputs are identical to `statSimDS`, above. The only discrepancy being that the array `S` is now two dimensional, such that $S_{jk} = S\left(t_j, \omega_k\right)$, and in line 11, where the ES must be indexed in time.

```
function [ y ] = nonStatSim( w, t, S )

    dw  = w(2)-w(1);         % frequency step size
    N   = length(S);         % number of points in the frequency domain
    phi = rand(1,N)*2*pi;    % N random phase angles between (0,2\pi)

    % Generate sample realization:
    %   Loop over time and sum over frequency
    y   = zeros(1,length(t));   % initialize sample function
    for i = 1:length(t)
        y(i) = sum( 2*sqrt( S(i,:)*dw ) .* cos(w*t(i) + phi) );
    end
end
```

C.4 Stochastic waves

C.4.1 MATLAB codes

Direct Summation

Below is a sample MATLAB function which generates sample realizations of a stochastic wave through the SRM. This is a direct implementation of Eqn. (5.35). The same as earlier examples, this code takes as input the spectrum (in this case the 2D F-K spectrum of Chapter 8), as well as time and space vectors. It then gives as output the 2D sample function in space and time.

```
function [ y ] = waveSimDS( w, k, x, t, S )

    % Original domain:
    dw      = w(2)-w(1);    % frequency step size
    Nw      = length(w);    % number of points in orig. freq. domain
    dk      = k(2)-k(1);    % wavenumber step size
    Nk      = length(k);    % number of points in orig. w-n domain
    [K,W]   = meshgrid(k,w);% mesh w and k to vectorize operations

    % Random phase angles:
    phi1             = rand(Nk,Nw)*2*pi;
    phi2             = rand(Nk,Nw)*2*pi;

    % Generate sample realization:
    %   Loop over space & time and sum over frequency & wave-number
    y                  = zeros(length(t),length(x)); % initialize sample func
        .
    for ii = 1:length(t)
        for jj = 1:length(x)
            y(ii,jj) = 2*sum(sum(sqrt(S2*dw*dk).*...
                       ( cos(W*t(ii)+K*x(jj)+phi1) +...
                       cos(-W*t(ii)+K*x(jj)+phi2))));
        end
    end
end
```

FFT

Below is a sample MATLAB function which generates sample realizations of a stochastic wave through the SRM. This function makes use of the FFT formulation, which is necessary in the multi-dimensional/wave case. The function presented represents Eqn. (8.13). Just like `statSimFFT`, above, the time and space vectors are outputs as they are dependent on the FFT. The need for two phase angle arrays, and the `ifft` vs. `fft` is consistent with Eqn. (8.13).

```
function [ y , x, t ] = waveSimFFT( w, k, S )

    % Original domain:
    dw       = w(2)-w(1);     % frequency step size
    Nw       = length(w);     % number of points in orig. freq. domain
    dk       = k(2)-k(1);     % wavenumber step size
    Nk       = length(k);     % number of points in orig. w-n domain

    % New domain:
    Mw                 = 2^nextpow2(2*Nw); % M = 2*N, or next power of 2
    Mk                 = 2^nextpow2(2*Nk);
    S2                 = zeros(Mk,Mw);     % pad new spectrum with zeros
    S2(1:Nk,1:Nw)      = S;
    t                  = 2*pi/dw*linspace(0,1,Mw+1); t(end)=[]; % time vector
    Mt                 = length(t);        % number of time points
    x                  = 2*pi/dk*linspace(0,1,Mk+1); x(end)=[]; % space vec.
    Mx                 = length(x);        % number of spatial points

    % Random phase angles:
    phi1               = rand(Mk,Mw)*2*pi;
    phi2               = rand(Mk,Mw)*2*pi;
    % Integrands for FFT:
    B1                 = 2*sqrt(S*dk*dw).*exp(1i*phi1);
```

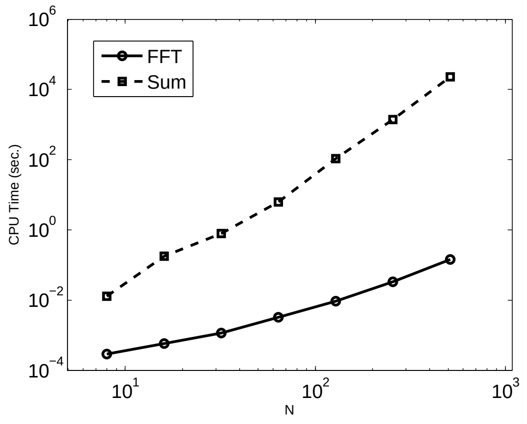


Figure C.2: Comparison of computational efficiency between using the direct implementation of Eqn. (5.35) and the FFT implementation of Eqn. (8.13).

```
    B2                = 2*sqrt(S*dk*dw).*exp(1i*phi2);
    % FFT along wavenumber first:
    F1                = Mk*ifft(B1,Mx,1);
    F2                = Mk*ifft(B2,Mx,1);
    % FFT along frequency next:
    F1                = Mw*ifft(F1,Mt,2);
    F2                = fft(F2,Mt,2);
    % Put it all together, and take the real part:
    y                 = real(F1+F2);

end
```

C.4.2 Efficiency comparison

The computational efficiency of the two wave-SRM functions just presented, `waveSimDS` and `waveSimFFT`, are now compared. The number of points in the domain, N, is varied from 2^3 through 2^9. Note, (1) that $N_\omega = N_\kappa = N$ for this, now, 2D domain; and (2) that it was ensured that the same number of points were used in both cases, accounting for the padding in the FFT formulation. Thus, for any given N, $M = 2N$ was used in *both* cases, with the padded spectrum. The CPU time was averaged over 3 simulations for each N, and the average CPU time is plotted in Figure C.2. The advantage of using the FFT formulation is quite clear - it is multiple orders of magnitude more efficient. As N increases, this is even more evident.