# EFFICIENT ALGORITHMS AND HIGH-PERFORMANCE SIMULATIONS FOR A CLASS OF DETERMINISTIC AND STOCHASTIC MODELS

by

Brandon C. Reyes

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Applied Mathematics and Statistics).

Golden, Colorado

Date _____

Signed: _____
Brandon C. Reyes

Signed: _____
Dr. Mahadevan Ganesh
Thesis Advisor

Signed: _____
Dr. Vladislav Petyuk
Thesis Co-Advisor

Golden, Colorado

Date _____

Signed: _____
Dr. Greg Fasshauer
Professor and Head
Department of Applied Mathematics and Statistics

# ABSTRACT

Efficient numerical algorithms and high-performance computing (HPC) simulations are crucial for the understanding and discovery of complex biological and physical processes. The main focus of this thesis is on the development and HPC implementation of algorithms to address key challenges associated with these processes. These include numerical approaches for the identification of deterministic biological signaling pathways capable of switch-like behavior; efficient finite element method (FEM) based uncertainty quantification (UQ) algorithms for the reduction in computational resources to simulate stochastic-space-time physical models; and neural network (NN) based approximations of deterministic counterparts of the space-time models.

To showcase our efficient FEM constrained high-order multilevel UQ algorithmic developments for space-time physical processes with input uncertainties (in a class of phase-separation and quantum systems) and marked advantages on CPU-based HPC environments with message passing interface (MPI) implementations, we simulate high-stochastic dimensional systems governed by the Allen-Cahn (A-C) and Schrödinger equations, respectively defined on stationary and time-dependent domains. Using a physics constrained NN (PCNN) algorithm we simulate a data-driven inverse problem for parameters in a deterministic A-C model, and also demonstrate accuracy and GPU computational costs associated with our PCNN approach for the simulation of a class of oscillatory processes in moving domains, governed by the Schrödinger equation in three spatial dimensions. For the biological processes, we demonstrate the effectiveness of our optimization-based method for general detection of switch-like behavior on models describing a biterminal futile signaling cycle and a biterminal prion/double phosphorylation motif.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

# ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisors Professor Mahadevan Ganesh and Dr. Vlad Petyuk for their constant guidance and support. Without their help, my introduction into the world of research would have surely been a painful and arduous process. I know that the knowledge both of you have given me will continue to aid me for the rest of my life. The support of the Colorado Golden Energy Computing Organization (GECO) is gratefully acknowledged. Lastly, I would like to extend a gigantic thank you to my family and friends. When it comes down to it, all of you are the only reason I can make it through the day and continue to push myself. In particular, I would like to thank my immediate family (Mom, Dad, Kristin, Megan, Rusty, Laney, Gavin), the man you think I am, is the man I strive to be every day.

# CHAPTER 1

# INTRODUCTION

Efficient numerical algorithms and high-performance computing (HPC) simulations are crucial for the understanding and discovery of complex biological and physical processes. A key reason for this is that most real-world applications are governed by complex time- and space-dependent interactions. The complexity of a process' time-evolutions and/or spatial-interactions can be mathematically modeled by a continuous system of Ordinary Differential Equations (ODEs) in time, and/or a Partial Differential Equation (PDE) in space and time, in conjunction with initial and boundary conditions. Such continuous systems cannot, in general, be solved analytically. However, their counterpart approximate discrete systems can be simulated, allowing one to understand the processes.

Setting up the approximate high-dimensional discrete systems requires robust model-based algorithms, and their HPC implementations facilitate efficient and accurate simulation of the processes, using the model input information. The lack of precise knowledge of the input data such as parameters in a chosen model, leads to stochastic versions of the model. The stochastic-dimension of the model is dictated by the number of uncertain input parameters in the model. This typically high stochastic-dimension introduces additional complexity, which requires efficient stochastic sampling-based approximation methods. Consequently, determined by each sample realization, a large number of simulations of the associated deterministic model demand parallel implementation of the hybrid deterministic-stochastic algorithm. The main focus of this thesis is to consider a class of such biological and physical processes that are motivated by continuous deterministic and stochastic models on stationary and moving domains. For each of these models, we develop efficient numerical algorithms and simulate the processes using HPC implementations of the algorithms.

We begin our journey into this research space by first considering a biochemical process modeled by a system of nonlinear ODEs that characterize signaling pathways. Signaling pathways capable of switching between two states are ubiquitous within living organisms. They provide the cells with the means to produce reversible or irreversible decisions. Switch-like behavior of biological systems is realized through biochemical reaction networks capable of having two or more distinct steady states, which are dependent on initial conditions. This is an important area of research as it has been shown that distortion of these signaling pathways with switch-like behavior manifests in no switching at all, switching at incorrect input signals, irreversible switching, and other phenomena. These malfunctions result in incorrect cell decisions and may be one of the underlying causes of developmental disorders such as cancer, diabetes, and presumably a number of other pathologies [1–3]. Investigation of whether or not a signaling pathway can confer bistability and switch-like behavior (without knowledge of specific kinetic rate constant values) is a mathematically and computationally challenging problem.

For this complex and challenging model, a number of techniques have been developed to investigate signaling pathways. One well-established theoretical framework to preclude multistationarity (and therefore bistability) in a network motif following mass action kinetics was developed by Feinberg, Horn, and Jackson [4, 5]. This theory, aptly named Chemical Reaction Network Theory (CRNT) uses the underlying structure of the reactions in the network to identify key properties. CRNT includes results such as the Deficiency Zero and One Theorems, which preclude bistability for certain network structures, irregardless of the kinetic constant values [4]. Although these theorems are very powerful, it is often the case that more complex networks found in cell signaling do not meet the deficiency requirement of these theorems [6]. To consider networks with higher deficiency, the Deficiency One Algorithm and Advanced Deficiency Algorithm were developed [6, 7]. These algorithms use the structure of the network to construct a system of equalities and inequalities. If these systems are solvable by either linear or nonlinear programming, they can state the existence of multiple positive steady states. However, these methods do not provide any conclusions about switchability between the alternative steady states.

To further understand how bistability is characterized, algebraic methods have been developed. These methods often utilize the Gröbner basis approach [8, 9], cylindrical algebraic decomposition [10] or sign pattern analysis of the Jacobian [11] to obtain a restricted analytical solution of the system. Although useful in chemical reaction networks that do not consist of too many reactions, these methods often fail due to their symbolic complexity. Recently, the optimization-based approach of [12] has gained attention as it provides an efficient procedure to search for bistability with respect to the conservation laws of a mass action system and is capable of handling large reaction networks. This particular approach attempts to find bistability by constructing an efficient optimization problem to find a saddle-node and then uses numerical continuation to confirm if the saddle-node is a saddle-node bifurcation (in general for the signaling pathways under study bistability occurs via saddle-node bifurcations).

This newer approach is of considerable interest because it allows 1) evaluation of fairly large pathways and 2) directly imposing bounds on values of the parameters of the network, such as the species' concentrations and kinetic constants. Although this method has proven to be useful in addressing real-world signaling pathways, it assumes that the network is of a particular structure. More specifically, that the graph representation of the network is uniterminal (as defined in Chapter 2). To further this very promising approach and address its restrictions, in Chapter 2 we construct an optimization-based approach that provides an efficient procedure for the detection of switch-like bistability in reaction networks governed by mass action kinetics for any network structure.

For rest of this thesis, we switch our focus to space-time PDE models and develop efficient algorithms and HPC implementation of the models. In particular, in Chapters 4 and 5, we consider a class of stochastic-PDE models, respectively, in stationary and moving spatial domains. In these two chapters, respectively, we use the stationary- and moving-mesh finite element methods (FEMs) for deterministic counterparts of the models in space-time variables; and, for stochastic variables, we use single- and multi-level Monte Carlo (MC) and quasi-MC (QMC) approximations. To this end, we first describe the stochastic approximation techniques in

Chapter 3 and demonstrate these using standard examples.

The uncertain nature of parameters in our models in Chapters 4 and 5, lead to solutions being stochastic process quantities of interest (QoI). The uncertainty quantification (UQ) of the QoI, using statistical moments metrics (such as mean and variance), facilitates understanding the behavior of the processes that are modeled by the stochastic-PDE models. Stochastic computational models with UQ have been established for several systems such as those modeling diffusion, financial, and wave propagation processes. See for example [13–22]. Due to the inherent input data uncertainty in our stationary- and moving domain stochastic PDE models, for the associated output quantities, the fundamental first- and second-moments (mean and variance) are crucial for understanding the QoI in a probability space. Each of these output stochastic process moments are defined in continuous form by a high-dimensional integral with respect to the probability measure. The MC and QMC methods provide, respectively, low- and high-order N-term summation approximations of the integral using equal weights and N-samples in the probability space [13, 14, 23–25]. In our stochastic-PDE models, for each term in the summation we need to solve the associated deterministic space-time PDE models using the FEM approximations. Thanks to the advancement in multi-core parallel computing hardware, such MC/QMC simulations have become possible through parallel scientific computing programming techniques such as the message passing interface (MPI) in conjunction with multi-core processing hardware.

Although the MC-based simulation is the standard choice to discretize high-dimensional integrals [14], the MC simulation cost becomes computationally very expensive when the QoI is obtained by the need to solve space-time complex models using FEM-type discretization methods. This is due to the fact that the low-order MC approximation convergence rate is only $O(N^{-1/2})$, where we recall that $N$ is the number of samples in the probability space. The Quasi-Monte Carlo (QMC) methods provide an approach to improve the convergence rate using high-order approximations through a carefully chosen set of quasi-random sampling points. These deterministic points take on various forms and are often generated using widely used

Sobol' sequences or the general class of lattice rules or digital-nets [26–28]. Although many of these sampling based QMC methods are important for applied models, depending on the stochastic regularity of the QoI, the interlaced digital-nets based approaches provide a way to further increase convergence rates of the polynomial lattice-rules QMC sampling methods with $\mathcal{O}(N^{1-\epsilon})$ convergence rate (for some $\epsilon > 0.5$). An extensive amount of material has been dedicated to this subject as provided by [26, 29–34] and references therein.

Mathematical analysis of QMC-based algorithms for stochastic-PDE simulations, using the PDE-specific designed QMC points, have recently been an active area of research, see the recent article [35] and references therein. Establishing a similar analysis for Sobol' sequences based stochastic-PDE simulations is an open problem. However, the black-box nature of generating the Sobol' QMC points facilitates general user-friendly framework for applications including interlaced counterparts of such points for high-order approximations and has been demonstrated to perform similar to the PDE-specific designed QMC points, see [19]. In this thesis, we use interlaced Sobol' QMC points for our single- and multi-level stochastic PDE simulation algorithms.

Although MC simulations can be improved upon if QMC is used, both of these methods can still consume a large amount of HPC resources because the order constant in the convergence rate also depends on the variance of the QoI. Additionally, the stochastic-variable regularity of the QoI for the QMC case is tied to how much improvement is observed in the convergence rate. To reduce the computational complexity of MC and QMC, we will conclude our exploration of UQ tools by considering Multilevel Monte Carlo (MLMC) and Multilevel Quasi-Monte Carlo (MLQMC) simulations. The multi-level (ML) approach was first proposed in the context of MC approximations, see [17] and references therein. The MLMC and MLQMC approximations are obtained by an appropriate trade-off between the MC/QMC simulations and the FEM approximations. The trade-off reduces the computational complexity of MC and QMC by allowing one to perform a few more, say, $L$-levels of MC/QMC simulations, with QoI integrands having smaller variances, as we increase from level 0 to level $L$. The reduction in the

computational cost can be achieved using certain trade-offs between finer and coarser realizations of the numerical grids being considered. In Chapter 3, we review all of these methods and provide a clear structure to their implementation that we will utilize in later chapters.

In Chapter 4 we consider uncertain nature of a physical process on a stationary domain described by the Allen-Cahn (A-C) system, which is a celebrated phenomenological nonlinear mathematical model. The A-C model describes processes such as the phase separation in the mixing and creation of alloys and was proposed about four decades ago by Allen and Cahn [36]. The model is composed of a nonlinear space-time PDE, initial state, and boundary conditions [37, Chapter 6]. The deterministic A-C system has been widely investigated in literature for its relation to several physical processes such as crystal growth [38], population dynamics [39], image segmentation [40], and motion by mean curvature [41–43]. One of the key features of the A-C model is on the description of the evolution of complex initial structures of the order field to a simple desired state. Numerical simulations of such key features from known deterministic initial states have been established in several articles, see for example [38, 40, 42, 44] and the extensive references therein.

It is also known that the assumption of complete knowledge of the A-C system is not appropriate. As a consequence, stochastic counterparts of the A-C equation have been proposed, see for example [45–47] and references therein for theoretical investigations of various mathematical properties associated with such models. However, numerical investigations to quantify uncertainties in stochastic A-C systems are yet to be investigated in the literature. In Chapter 4, we develop and implement a hybrid high-order multi-level stochastic A-C computational model (using elements from Chapter 3), in conjunction with finite element in space-time and efficient sampling of the high-dimensional stochastic space, to compute statistical moments of a QoI induced by the A-C output field. The input uncertainty in the model occurs both in the A-C PDE and in the initial state that facilitates the stochastic evolutionary process. Our hybrid deterministic-stochastic approximation framework is based on a space-time finite element method in two and three spatial dimensions, high-order approximations in high-dimensional stochastic

variables, and fast evaluations using multi-level and parallel computations.

In Chapter 5 we consider our second stochastic model, however, this time concerned with a physical process on a moving domain with the underlying process governed by the space-time Schrödinger equation. The Schrödinger equation is a celebrated linear PDE originally proposed about a century ago to describe the wave function of a quantum-mechanical system. Our research is concerned with efficient simulation of statistical moments of a moving domain induced stochastic quantum transients. The theory and applications of transients have been a widely studied field, for an in-depth review of this material, we refer to [48] and a large number of references therein. One particularly interesting application of quantum transients was introduced in the seminal 1952 paper by Moshinsky [49], entitled *Diffraction in Time* (DiT). Even several decades after introduction of the DiT model, there continues to be substantial interest in understanding more practical variants and solvability of the model [50], many of which consider a quantum matter transient wave phenomena related to that in optics.

As described in the review article [48, Page 2], the transient models investigated mainly in the (Physics) literature are those for which analytical solutions can be constructed using classical functions (such as Fresnel integrals). Such analytic constructions impose restrictions on the model such as requiring stationary shutters, deterministic initial conditions, or no boundary conditions (to apply Fourier/Laplace transforms). These include the seminal DiT model by Moshinsky [49] and the recently investigated exactly solvable variant DiT model [50] and references therein. The model in [50] admits a time-dependent aperture/shutter function $\chi(t)$ and represents the resulting unknown particle wave function $\Psi(x,t)$ as an improper-spatial-integral on $(-\infty,\infty)$, with its integrand depending on a deterministic initial state and a fundamental solution Green's kernel, satisfying the time-dependent one-space dimension Schrödinger equation. In this simplified case, the improper integral can be evaluated only for certain special forms of $\chi(t)$ and the approach in [50] also assumes that the Schrödinger Green's kernel can be evaluated exactly. In general, the latter is not possible when the one-dimensional Schrödinger equation is considered in conjunction with time-dependent spatial-boundedness of the moving shutter (as the beam of

the particles impinge on the shutter), associated boundary conditions, and a non-deterministic continuous initial state.

The main focus of Chapter 5 is developing and demonstrating an algorithm to compute statistical moments of a QoI, determined by a stochastic quantum density profile induced by a random continuous initial state. The stochastic quantum density profile integrand is determined by the evolution of the unknown wave function satisfying a stochastic Schrödinger model on a moving domain, which is determined by a random moving shutter. In the deterministic case of this model (and a discontinuous initial state), the model reduces to the stationary shutter DiT model introduced by Moshinsky [49], which models the deterministic quantum dynamics of a suddenly released beam of independent particles. In addition to applying efficient methods for computing statistical moments of the QoI (using techniques established in Chapter 3), we also develop an efficient moving-mesh FEM for solving the associated deterministic version of the stochastic Schrödinger model on a moving domain.

In Chapters 4 and 5 of this thesis, we demonstrate that the MLMC and MLQMC methods provide substantial advantages compared to the standard MC and QMC methods for solving stochastic PDEs. Such advantages are possible because of efficient simulations of the associated deterministic PDEs using the classical spline-approximations based FEM. Although these methods are preferred for the time being, it is anticipated that the current and future decades will be dominated by data-driven neural network (NN) based approximations. Thus, in conjunction with PDE systems (similar to spline-based approximations for FEM), machine learning methods (MLM) will require that the NN-approximations they produce approximately satisfy the physics based PDE systems. In recent years, such Physics Informed NN (PINN) approaches have been explored in the literature for PDE systems defined on stationary domains, see for example [51–53] and references therein.

To this end, in Chapter 6 we first explore NN approximations for data driven problems and then PINNs for classical PDE models. In particular, we consider the PDE model that played a crucial role for developing an efficient algorithm for the deterministic A-C system in Chapter 4. Based on

our own investigations of these PINN approximations, we then find that unlike the FEM, the PINN approach does not satisfy the boundary and initial conditions to high accuracy. Hence, subsequent time evolution processes are not very efficient and often fail to achieve less than the scientifically appropriate 0.1% accuracy. For this reason, we consider an improvement upon the PINN framework by using a method that more accurately describes the given boundary and initial data. This method provides the physics constrained NN (PCNN) framework, which yields over a 100% gain in accuracy compared to the PINN-based approximations for space-time models. We demonstrate the efficiency of the PCNN approach for an inverse parameter model associated with the A-C system on a stationary domain. The main purpose of this exploration is to develop NN approximations for a time-dependent domain model governed by the Schrödinger equation with oscillatory solution behavior. The developed PCNN approach is advantageous as boundary and initial conditions are *a priori* satisfied for our models and incorporated into the NN-based approximations. To our knowledge, this is the first time NN approximations have been considered for PDEs on moving domains and its performance and advantages compared with the moving-mesh FEM based approximations.

This thesis is organized as follows: In Chapter 2 (and Appendix A) we consider the biological application of detecting switch-like bistability in chemical reaction networks. We accomplish this by considering basic terminology and the review of past techniques for the detection of switch-like bistability. We then construct a set of new procedures that allow for the general detection of said bistability for any chemical reaction network governed by mass action kinetics. In Chapter 3 we recall known theory and algorithms necessary for the efficient simulation of key metrics for a QoI dictated by a stochastic-PDE model and demonstrate such methods using basic models. In Chapter 4 we utilize the methods developed in Chapter 3 to simulate the uncertain nature of the physical process described by two-/three-spatial and higher-stochastic dimensional nonlinear A-C systems; where our algorithms include an efficient operator-splitting based FEM in space-time and single- and multi-level stochastic algorithms. In Chapter 5 we again consider a physical process in conjunction with basic ideas developed in Chapter 3, but this time we

investigate how the stochastic time-dependent Schrödinger model evolves on a moving domain, utilizing our efficient moving-mesh FEM and single- and multi-level stochastic algorithms. We conclude our research in Chapter 6 by exploring how NNs can be used as approximations for the moving domain time-dependent Schrödinger deterministic model established in Chapter 5, using the PCNN framework.

# CHAPTER 2

# DETECTION OF BISTABILIY IN MASS ACTION BIOCHEMICAL REACTION NETWORKS

Cellular decisions via signaling pathways are essential for complex biological systems to function. The key attribute of signaling pathways which are capable of mediating a decision-making process is switch-like behavior. Such behavior assumes that the system has (typically) two stable equilibria and there is a way to switch between them. In bistable switches, two different thresholds for switching back and forth ensure the robustness of the decision. This characteristic dose-response behaviour is called hysteretic. Distortion of these signaling pathways with switch-like behavior manifests in no switching at all, switching at incorrect input signals, irreversible switching, etc. These malfunctions result in incorrect cell decisions and may be one of the underlying causes of developmental disorders such as cancer, diabetes, and presumably a number of other pathologies [1–3]. Given that cellular decisions can consist of an immense number of biochemical interactions, smaller network motifs are often considered and can help elucidate the portions of the signaling pathway that are key to the decision-making process [54]. Although discovering essential network motifs can provide a wealth of information, obtaining these configurations is not only difficult, but costly if approached purely from an experimental point of view. For this reason, it is imperative for this process to be coupled with mathematical modeling, which can act as a guide for designing experiments. Analysis of these models is useful as they can determine the existence of bistability in the signaling pathway, an attribute directly tied to the pathway's ability to exhibit switch like behavior. Existence of bistability in chemical reaction networks has been an active area of research since the 1970s. In particular, a wealth of mathematical theory has been oriented towards network motifs that utilize mass action kinetics for the participating reactions. This is due to the fact that mass action law does not employ assumptions on time-and concentration-scale separation as do other kinetics, such as

Michaelis-Menten [55–58].

One well-established theoretical framework to preclude multistationarity (and therefore bistability) in a network motif following mass action kinetics was developed by Feinberg, Horn, and Jackson [4, 5]. This theory, aptly named Chemical Reaction Network Theory (CRNT) uses the underlying structure of the reactions in the network to identify key properties. CRNT has produced results such as the Deficiency Zero and One Theorems, which preclude bistability for certain network structures, irregardless of the kinetic constant values [4]. Although these theorems are very powerful, it is often the case that more complex networks found in cell signaling do not meet the deficiency requirement of these theorems [6]. To consider networks with higher deficiency, the Deficiency One Algorithm and Advanced Deficiency Algorithm were developed [6, 7]. These algorithms use the structure of the network to construct a system of equalities and inequalities. If these systems are solvable by either linear or nonlinear programming, they can state the existence of multiple positive steady states. However, these method do not provide any conclusions about switchability between the alternative steady states. For more details on the topic of switchability between steady states we would like to refer the reader to [9]. In addition to the Deficiency One Algorithm and Advanced Deficiency Algorithm, injectivity theory and network concordance tests have also been developed, which attempt to address those networks that are not covered by the aforementioned theory [59, 60].

Besides CRNT and linear/nonlinear programming, there are a number of alternative approaches for gaining insights into an ODE system's behavior. For example, algebraic methods utilizing the Gröbner basis approach [8, 9], cylindrical algebraic decomposition [10] and sign pattern analysis of the Jacobian [11] have been developed in concert with CRNT. The goal of which is to obtain analytical solutions to the system of ODEs at the steady state and then further analyze the solutions to detect bistability. A number of methods have also been reviewed and compared in [61]. Additionally, promising approaches have been developed that attempt to determine the parameter regions where multistability may occur using symbolic [62] or numerical [63] methods.

Recently, the optimization-based approach of [12] has gained attention as it provides an efficient procedure to search for bistability with respect to the conservation laws of a mass action system. For brevity we will refer to this method as the uniterminal approach (where the term uniterminal is defined in the Basic notation and definitions section) as the approach is limited to uniterminal networks. This particular approach attempts to find bistability by constructing an efficient optimization problem to find a saddle-node and then uses numerical continuation to confirm if the saddle-node is a saddle-node bifurcation (in general for signaling pathways under study bistability occurs via saddle-node bifurcations). This approach is of considerable interest because it allows 1) evaluation of fairly large pathways and 2) directly imposing bounds on values of the parameters of the network, such as the species' concentrations and kinetic constants. The uniterminal approach is constructed based on the assumption that 1) every reaction is endowed with mass action kinetics, 2) the network admits a strictly positive steady state (where the concentrations of all the species are positive) and 3) the network is uniterminal.

In this chapter we first develop basic definitions and notation associated with chemical reaction networks governed by mass action kinetics. Using this notation, we then describe two main theorems of CRNT, the Deficiency One and Zero Theorems. Once these theorems have been established, we consider the optimization-based approach of [12], which considers the elucidation of bistability in uniterminal networks. We then develop and present a general approach for mass action chemical reaction networks with conservation laws that is not limited to the structure of the network. Lastly, we present CRNT4SBML, our easily installable Python based package available on MacOS and Windows, which is concentrated on providing a simple workflow for the testing of core CRNT methods directed at detecting bistability in cell signaling pathways endowed with mass action kinetics.

## 2.1 Basic notation and definitions

Assuming mass action kinetics, a given chemical reaction network can be represented as a system of autonomous ODEs composed of $N$ species and $R$ reactions

$$\dot{\mathbf{c}} = f(\mathbf{c}, \mathbf{k}). \tag{2.1}$$

Here $\dot{\mathbf{c}}$ denotes the temporal derivative of the species concentrations $\mathbf{c} \in \mathbb{R}^N$, $\mathbf{c} \geq 0$, with each species having concentration $c_i$ for $i = 1, \ldots, N$. The vector $\mathbf{k} \in \mathbb{R}^R$, $\mathbf{k} > 0$, represents the kinetic rate constants of the reactions (determined by mass action kinetics), where the individual kinetic rate constants are denoted as $k_i$ for $i = 1, \ldots, R$. We also denote the individual reactions as $r_i$ for $i = 1, \ldots, R$. In addition to the system of ODEs, we also require that the network have one or more conservation laws, which we denote as follows:

$$\mathbf{C} = g(\mathbf{c}),$$

where $\mathbf{C} \in \mathbb{R}^\lambda$, $\lambda$ denotes the number of conservation laws, and each conservation law is given as $C_i$ for $i = 1, \ldots, \lambda$.

The following sections are heavily based on the CRNT terminology, such as complexes and linkage classes. Briefly, a complex (in CRNT terminology) is a list of reactants or products for a given reaction. They are denoted as $C_i$ for $i = 1, \ldots, M$. Complexes and corresponding reactions constitute the vertices and edges of the chemical reaction network or graph (also know as a C-graph). Disconnected subgraphs of this network are called linkage classes. For a given network we let $\ell$ be the number of linkage classes and denote them as $\mathcal{L}_i$ for $i = 1, \ldots, \ell$. By inspecting a linkage class further, one can then determine if a given network is uniterminal using Definitions 2.1, 2.2, and 2.3. The simplest depictions for a network being uniterminal and biterminal are depicted in Figure 2.1. Note that in the uniterminal case a linkage class containing a single complex is itself uniterminal as a single complex is strongly linked to itself [4]. Although this is assumed by CRNT, this is of little or no interest in application (as it essentially means no reaction occurs) and is simply stated for theoretical completeness.

**Definition 2.1.** Strongly linked nodes

Two nodes $C_i, C_j$ are said to be strongly linked if there is a directed path from $C_i$ to $C_j$ and also a directed path from $C_j$ to $C_i$.

**Definition 2.2.** Terminal strong linkage class

A terminal strong linkage class is a maximal set of nodes within a disconnected subgraph such that there is no edge pointing to any other set of nodes that are strongly linked.

**Definition 2.3.** Uniterminal network

We say that a network graph is uniterminal if every linkage class in the graph contains only one terminal strong linkage class.

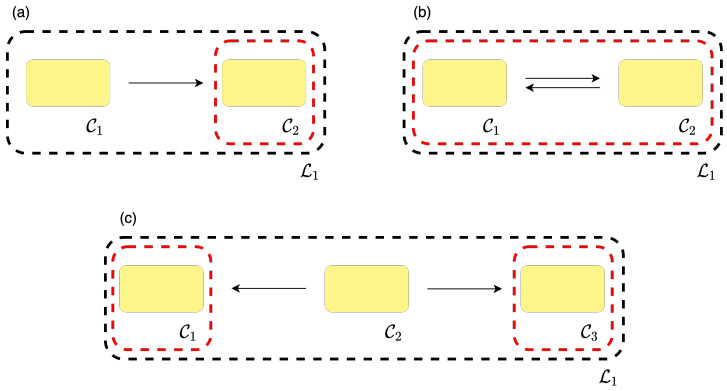
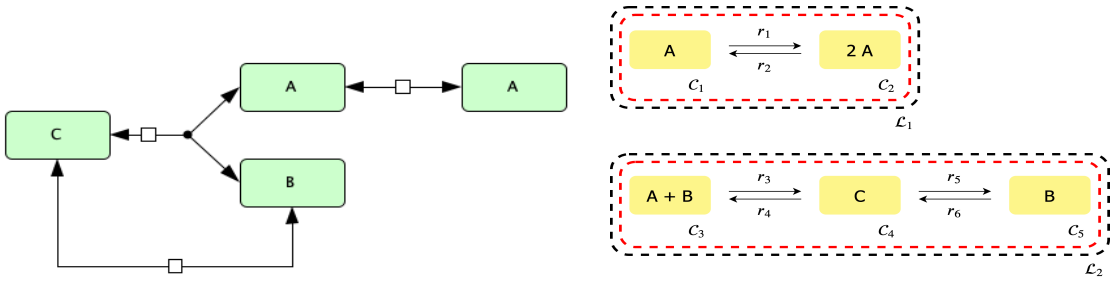



Figure 2.1: Examples of linkages classes. The black and red dashed lines in the subfigures indicate the linkage class and terminal strong linkage classes of the network, respectively. Thus, (a) and (b) represent the simplest possible linkage classes that form a uniterminal network. (c) Depicts the simplest linkage class with two terminally strong linkage classes (a necessary condition for a biterminal network).

Without loss of generality and for the sake of simplicity, we will be using the well-known Edelstein network, which is uniterminal, to further develop descriptions of the chemical reaction network. Originally, the Edelstein network was introduced in [64] and we choose to use the reduced form presented in Lecture 3 of [65]. This provides the Edelstein network as shown in Figure 2.2. Thus, we have $\mathbf{k} = (k_1, k_2, k_3, k_4, k_5, k_6)^T$ and $\mathbf{c} = (c_1, c_2, c_3)^T$ with $c_1 = [A], c_2 = [B], c_3 = [C]$. In the following subsections we develop key terminology using this network.




(a) (b)

Figure 2.2: Edelstein chemical reaction network [64]. (a) diagram of reaction network and (b) its C-graph representation.

### 2.1.1 Constructing the full ODE system

Constructing the ODE system describing species' concentration dynamics for the given reaction network is simple. Assuming mass action kinetics, we obtain the following representation for our autonomous ODEs:

$$\dot{\mathbf{c}} = YA\psi(\mathbf{c}) \Longleftrightarrow \begin{aligned} \dot{c}_1 &= k_1c_1 - k_2c_1^2 - k_3c_1c_2 + k_4c_3 \\ \dot{c}_2 &= (k_4 + k_5)c_3 - k_3c_1c_2 - k_6c_2 \\ \dot{c}_3 &= k_3c_1c_2 - (k_4 + k_5)c_3 + k_6c_2. \end{aligned} \tag{2.2}$$

The molecularity matrix $Y$ is a $N \times M$ matrix where $Y_{ij}$ corresponds to the molecularity of species $i$ in complex $j$. For our example the complexes are $C_1 = A, C_2 = 2A, C_3 = A + B, C_4 = C$, and $C_5 = B$, which provide the molecularity matrix:

$$Y = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccccc} A & 2A & A{+}B & C & B \end{array} \\ \begin{pmatrix} 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}.$$

$A$ is the $M \times M$ kinetic constant matrix, where the diagonal elements of $A$ contain the negative of the sum of the kinetic rate constants corresponding to the reactions going out from $C_i$. While the off-diagonal elements, say $A_{ij}$, contain the kinetic rate constants of the reactions going from $C_j$ to $C_i$. Here $i$ and $j$ correspond to the $i$th row and $j$th column of A, respectively

$$A = \begin{array}{c} \\ A \\ 2A \\ A{+}B \\ C \\ B \end{array} \begin{array}{c} \begin{array}{ccccc} A & 2A & A{+}B & C & B \end{array} \\ \begin{pmatrix} -k_1 & k_2 & 0 & 0 & 0 \\ k_1 & -k_2 & 0 & 0 & 0 \\ 0 & 0 & -k_3 & k_4 & 0 \\ 0 & 0 & k_3 & -(k_4+k_5) & k_6 \\ 0 & 0 & 0 & k_5 & -k_6 \end{pmatrix} \end{array}.$$

The vector $\psi(\mathbf{c}) = (\psi_1, \ldots, \psi_M)^T$ defines the mass action monomials (a product of species' concentrations) associated with each complex

$$\psi_j(\mathbf{c}) = \prod_{i=1}^{N} c_i^{Y_{ij}}, \quad j = 1, \ldots, M.$$

For our example we obtain:

$$\psi(\mathbf{c}) = (c_1, c_1^2, c_1 c_2, c_3, c_2)^T.$$

### 2.1.2 Computing the mass conservation laws

Computing the mass conservation laws that govern the network is possible by considering the $Y$ matrix. We first construct the species stoichiometric matrix $S$ with dimension $N \times R$. Its entries can be constructed from the $Y$ matrix by noticing that every reaction from $C_i$ to $C_j$ has an associated vector $Y_j - Y_i$ where $Y_j$ and $Y_i$ are the $j$th and $i$th columns of Y, respectively. In our example, this produces the stoichiometric matrix $S$:

$$S = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{cccccc} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 \end{array} \\ \begin{pmatrix} 1 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 & -1 \\ 0 & 0 & 1 & -1 & -1 & 1 \end{pmatrix} \end{array}.$$

We also denote the rank, that is the maximal number of linearly independent columns, of $S$ as $s$, for notational convenience. For our example, $s = 2$.

Note, that the number of mass conservation relations is $\lambda = N - s$. Matrix $B$ (of dimension $N \times \lambda$) that defines such relations is defined as the nullspace of $S^T$, such that $S^T B = 0$. It should be also noted that the matrix spanning the nullspace of $S^T$ is not uniquely determined, and we choose $B$ such that all of its entries are nonnegative. This choice of $B$ is always possible provided that each

conservation law represents the conservation of a chemical or moiety [12].

$$\begin{aligned} &\text{Minimize} \quad \mathbf{w}^T\mathbf{x} \\ &\text{subject to:} \\ &\qquad -\tilde{B}\,\mathbf{x} \leq \mathbf{0}, \\ &\qquad \left(\sum_{i=1}^{N}\tilde{B}_{i,1},\ldots,\sum_{i=1}^{N}\tilde{B}_{i,\lambda}\right)^T\mathbf{x} = 1, \\ &\qquad -\infty \leq \mathbf{x} \leq \infty \quad \mathbf{x}\in\mathbb{R}^{\lambda}, \\ &\qquad \mathbf{w}^T \in \mathbb{R}^{\lambda}. \end{aligned} \tag{2.3}$$

For most reaction networks, constructing a $B$ matrix with nonnegative entries can be done by using linear programming. The construction of the linear programming problem requires a firm understanding of Polyhedra. Thus, we strongly recommend that the reader consider Sections 2.2 (in particular Subsection 2.2.3) and 2.4 of [66], which describe polyhedral sets and cones, and how the nullspace of the stoichiometric matrix can be described as a polyhedral cone, respectively. We now briefly describe the linear programming problem construction. The optimization problem is formed by first considering the intersection of $Null(S^T)$ and the nonnegative vectors of $\mathbb{R}^N$, which is an infinite cone. In order to obtain a finite set of this intersection, one can consider an intersection of the infinite cone with the set of vectors that have the sum of their coordinates equal to one. The solution to this intersection is a finite convex polytope. This conclusion can be seen by considering Definition 2.29 and 2.33 of [66]. It should be noted that [66] presents Enumeration Problems in general (Section 2.3) as a way to search for the vertices of this convex polytope. We instead search for these vertices beginning from random directions via the optimization problem (2.3), which is realized by the Simplex optimization method [67].

In optimization problem (2.3), $\mathbf{w}^T$ is the vector of random search directions and $\tilde{B}$ is the initial $B$ matrix with at least one negative entry. For our purposes, $\tilde{B}$ is initially set to the basis vectors that compose the span of $Null(S^T)$. The endpoints (i.e. the minimized $\mathbf{x}$ from (2.3)) of these vertices then form a linear basis element of the original nullspace consisting of only a nonnegative vector. Thus, by taking the dot product, $\tilde{B}\cdot\mathbf{x}$, we can obtain a vector with nonegative entries, with values between zero and one. To obtain integer entries, one can then divide the vector by the smallest nonzero entry of the vector. By repeating this process multiple times and obtaining $\lambda$ unique basis

elements, one can then form a $B$ matrix with nonnegative entries, as outlined in Algorithm 1. It should be noted that $\lambda$ is representative of the number of conserved chemical moieties. Thus, by definition we are certain that $\lambda > 0$ if the system contains conservation laws. A noteworthy remark of Algorithm 1 is that it does not guarantee that one will find all $\lambda$ basis vectors, given a fixed number of iterations. Thus, if the number of the found basis vectors is less than the pre-computed $\lambda = N - s$, the number of iterations should be increased. In addition to this approach, there exists alternative ways to obtain a $B$ matrix with nonnegative entries, see [68–70].

**Algorithm 1** Pseudocode for obtaining a $B$ matrix with nonnegative entries.

Set $\tilde{B} = Null(S^T)$.

Set the number of random search directions to $niter = (N + 1)(\lambda + 1)10$.

**if** $\tilde{B}$ has at least one negative entry **then**

    **for** $i = 1$ to $niter$ **do**

        Generate the random vector $\mathbf{w} \sim \mathcal{N}(0, 1)$.

        We solve optimization problem (2.3) using the Simplex algorithm for a given $\mathbf{w}$ and obtain $\mathbf{x}$.

        Set $\mathbf{z} = \tilde{B} \cdot \mathbf{x}$.

        Add $\mathbf{z}/min\{\mathbf{z}\}$, where $min\{\mathbf{z}\} \neq 0$, to the set of possible basis vectors.

    **end for**

    Create $B^T$ from the set of possible basis vectors, where the rows correspond to $\lambda$ unique basis vectors.

**else**

    Divide the rows of $\tilde{B}^T$ by the minimum nonzero value in each row. In practice, this is sufficient enough to transform the matrix into the form with all integer values because the material conservation laws are written with integer coefficients.

**end if**

Test the linear independence of $B^T$ by confirming that there are $\lambda$ pivots in the row-reduced echelon form of $B^T$.

Using $S^T$ and the procedure outlined in Algorithm 1, we obtain the $B$ matrix below:

$$B^T = \begin{matrix} \\ C_1 \end{matrix} \begin{matrix} \begin{matrix} A & B & C \end{matrix} \\ \begin{pmatrix} 0 & 1 & 1 \end{pmatrix} \end{matrix},$$

where $C_1$ again stands for the first conservation law. To obtain an explicit statement for the conservation laws, we simply take $B^T\mathbf{c}$, giving the following conservation law:

$$C_1 = c_2 + c_3.$$

### 2.1.3 Determining the independent ODE system

For any given reaction network, the number of independent ODEs describing the system's dynamics is equal to the rank$(S) = s \leq R$ [4]. Since we will ultimately consider the Jacobian of the ODE system with respect to the concentrations $\mathbf{c}$, it is necessary to remove those ODEs that can be represented as linear combinations of other ODEs in the system. We will let the independent system of ODEs be represented as follows:

$$\dot{\boldsymbol{\varsigma}} = \hat{f}(\mathbf{c}, \mathbf{k}),$$

where $\boldsymbol{\varsigma} \in \mathbb{R}^s$ represents those species' concentrations that form an independent ODE system. To obtain the independent system, we will first see if $s = N$, if this is true, this implies that there are no conservation laws. This scenario is out of the scope of this chapter. If $s < N$ then we have conservation laws and we must continue by finding the independent system. It is at this point where knowledge of the response in the bifurcation analysis is used to determine which ODEs should remain in the independent ODE system.

For our particular example, we take $C_1$ (the sum of $c_2$ and $c_3$) as the input signal, and $c_1$ as the output or readout of the system's response. Since $c_1$ (the concentration of species A) is taken as our system's response, it is necessary for $\dot{c}_1$ to be included in $\dot{\boldsymbol{\varsigma}}$. Using this fact and conservation law $C_1$, we can see that $c_2$ has a dependence on $c_3$ of the form $c_2 = C_1 - c_3$. For this simple example, this information is sufficient enough to eliminate $\dot{c}_2$ from $\dot{\mathbf{c}}$. Indeed if we consider the

full ODE system (2.2), we see that $\dot{c}_2 = -\dot{c}_3$. Thus, for our particular example we have $\dot{\mathbf{ç}}$ given by:

$$\dot{c}_1 = k_1 c_1 - k_2 c_1^2 - k_3 c_1 c_2 + k_4 c_3$$

$$\dot{c}_3 = k_3 c_1 c_2 - (k_4 + k_5) c_3 + k_6 c_2.$$

For larger systems the above methodology may not be straightforward. In practice, a set procedure to find this independent ODE system can be conducted. The pseudocode for this procedure is provided in Algorithm 2.

**Algorithm 2** Pseudocode for finding a linearly independent ODE system using exhaustive search. Where *PDS* denotes the lists of possible dependent species for each conservation law and *DS* contains all lists that represent our dependent species.

Let $y$ be the desired response species.
Let *PDS* be a list of length $\lambda$ initialized with empty lists as its elements.
**for** $i = 1$ to $\lambda$ **do**
    Set the $i$th element of *PDS* to the list of all species (not including $y$) in $C_i$.
**end for**
Create *DS*, an exhaustive list containing unique unordered combinations, by choosing one of the species from each of the *PDS* lists. Remove combinations that have repeating species.
**for** each element in *DS* **do**
    Let $P$ be the matrix formed by choosing the rows of $YA$ corresponding to those species not in the element of *DS*.

    **if** $rank(P) = s$ **then**
        Set $\mathbf{ç}$ to those species not in the element of *DS*.
        Construct $\dot{\mathbf{ç}}$ by using those species in $\mathbf{ç}$.
        break
    **end if**
**end for**

Referring to Definition 2.6, we see that $u$ and $\epsilon_0$ correspond to $\boldsymbol{\varsigma}$ and the input signal in our example, respectively, with $u = (c_1, c_3)^T$ and $\epsilon_0 = C_1$ From this definition we see that some modifications to $\dot{\boldsymbol{\varsigma}}$ must be made, in particular, we need to include the conservation law $C_1$. To eliminate all the dependent species ($DS$) we express them in terms of $\boldsymbol{\varsigma}$ and the corresponding conservation law. In our example, this is done by replacing $c_2$ with $C_1 - c_3$. For our example, we then have $\dot{\boldsymbol{\varsigma}}$ given as follows:

$$\dot{c}_1 = k_1 c_1 - k_2 c_1^2 - k_3 c_1 (C_1 - c_3) + k_4 c_3$$
$$\dot{c}_3 = k_3 c_1 (C_1 - c_3) - (k_4 + k_5) c_3 + k_6 (C_1 - c_3).$$

The system above is then equivalent to $F\left((c_1, c_3)^T, C1\right)$.

In addition to these definitions, CRNT presents further descriptors of the chemical reaction network in consideration. For example, a reaction network can also be weakly reversible, which is defined in Definition 2.4. The last definition we will consider is that of the deficiency of the network. The deficiency of the network is provided in Definition 2.5. Based off of these definitions, one can develop theorems that can rule out bistability. We describe these in the next section. As to not distract the reader, we have provided more informal definitions throughout this section. For more formal definitions and descriptions, we would strongly encourage the reader to review Appendix A, where we utilize the notation established in [65] that provides a mathematically strict way to define a chemical reaction network, in addition to the concepts provided in this section.

**Definition 2.4.** Weakly reversible

We say that a network graph is weakly reversible if any of the following conditions are satisfied:

1. whenever a complex $C_i$ ultimately reacts to a complex $C_j$ we also have that a complex $C_j$ ultimately reacts to a complex $C_i$.

2. whenever a complex $C_i$ reacts to a complex $C_j$ we also have that a complex $C_j$ ultimately reacts to a complex $C_i$.

3. each linkage class is a strong linkage class

4. each linkage class is a terminal strong linkage class

**Definition 2.5.** Deficiency

The deficiency of a reaction network is defined by the positive integer, $\delta$, given by $\delta = M - \ell - s$.

### 2.1.4 Deficiency Theorems

For our application, we then would like to know when our system of differential equations will admit multiple positive equilibria that may lead to bistability. One approach to answering this question would be to consider a particular choice of the rate constants and then solve this system using the initial concentrations as the initial condition. This act alone may be difficult for more complex reaction networks because of the nonlinear nature of the expressions. However, if we did manage to solve this system of ODEs for these fixed rate constants, we are still not reassured that these constants will produce multiple positive equilibria points. Thus, to complete this approach in its entirety we would have to solve this system multiple times with varying rate constants until we encountered a set of constants that produces bistability. This approach may be possible for less complex reaction networks, however, it is also computationally expensive and can take a large amount of time. To make matters worse, this is only for one particular reaction network. In practice, a scientist may want to consider multiple varying chemical reaction networks for their problem.

This necessitates the need to form a theory which can use the structure of the network, rather than the system of ODEs only, to definitively say whether or not multiple positive equilibria can exist. For certain network structures, there exists complete theorems that can determine this. Two of the most notable for our applications are that of the Deficiency Zero and One Theorems as provided in Theorems 1 and 2 (for more formal theorems, see Appendix A). The theory presented was developed by Martin Feinberg and multiple of his colleagues.

**Theorem 1.** The Deficiency Zero Theorem For any reaction network of deficiency zero we have the following criteria:

1. If the network is not weakly reversible then, for an arbitrary kinetics, the differential equations for the reaction system cannot admit a positive equilibrium (i.e. an equilibrium in the set of positive species' concentrations).

2. If the network is not weakly reversible then, for arbitrary kinetics, the differential equations for the reaction system cannot admit a cyclic composition trajectory containing a positive composition (i.e. a point in the set of positive species' concentrations).

3. If the network is weakly reversible (in particular, if the network is reversible) then, for any mass action kinetics, the differential equations for the mass action system have the following properties: There exists within each positive stoichiometric compatibility class precisely one equilibrium; that equilibrium is asymptotically stable; and there cannot exist a nontrivial cyclic composition trajectory in the set of positive species' concentrations.

**Theorem 2.** The Deficiency One Theorem For any reaction network with $\ell$ linkage classes, let $\delta$ denote the deficiency of the network; let $\delta_\theta$ denote the deficiency of the $\theta^{\text{th}}$ linkage class, $\theta = 1, 2, \ldots, \ell$; and suppose that both of the following conditions are satisfied:.

1. $\delta_\theta \leq 1, \;\; \theta = 1, 2, \ldots, \ell$

2. $\delta = \sum_{\theta=1}^{\ell} \delta_\theta$.

If the network is weakly reversible (in particular, if the network is reversible) then, for any mass action kinetics, the differential equations for the mass action system admit precisely one equilibrium for all physically relevant applications.

These theorems are of particular importance to us, because it will give our general framework an extremely fast way to rule out bistability. If the first two conditions of the deficiency zero theorem are met, then this means that no concentration values for our species will provide an equilibrium to the differential equations. If condition three is met, then that means that only one equilibrium point exists. If one or zero equilibria exists, then we know that bistability cannot, since bistability requires two equilibria. Similarly, if the deficiency one theorem is satisfied, we can rule out

bistability because only one equilibrium value is possible. Although these theorems can be useful for a section of chemical reaction networks encountered in the real-world, they are not sufficient to address all biological application that we are interested in. For this reason, we need to develop further techniques.

### 2.1.5 Bistability in Reaction Networks

Before we proceed with more extensive ways to characterize if a network is bistable, we will provide a description of the phenomena we wish to find. Provided system (2.1), it's possible that the chemical reactions will resolve to different equilibrium states depending on the starting conditions. Intuitively, such non-linear phenomenon can occur when the equilibrium concentration of one species is a polynomial function with degree greater than one with respect to an individual species. In biological systems, this behavior may present itself in a situation where a particular value of a signal species produces more than one solution with respect to a response species. In particular, bistability resulting from two stable branches connected by an unstable branch mimics switch-like action. Identification of such behavior is the focus of this chapter. Determining the stability of an ODE system is achieved by considering the eigenvalues of the Jacobian at a steady state point. The Jacobian of the ODE system is a matrix representation of the first-order partial derivatives of the ODE system with respect to state variables. By constructing the Jacobian one can approximate any ODE system with a system of linear ODEs at a given point. The stability at the steady state is defined by the sign of the real part of the Jacobian's eigenvalues. If any of the eigenvalues have a positive real component, the deviation from the steady state will increase in time, which implies an unstable steady state. In the case where all of the eigenvalues have a negative real component, the deviation decreases with time, indicating a stable steady state. In some cases, such as a bistable system, a single zero eigenvalue indicates that a steady state is momentarily transitioning from stable to unstable or vice versa. This transition happens as a consequence of varying a parameter of the ODE system.

In a mathematical sense, this transition from a stable state to an unstable state can be exhibited by a saddle-node bifurcation. Sufficient conditions for a saddle-node bifurcation are given by Theorem 3, where the point must also be a saddle-node according to Definition 2.6. For our purposes, $F$ in Definition 2.6 corresponds to the linearly independent system of ODEs with a variable signal, $u$ is the independent species' concentrations, and $\epsilon_0$ is the signal of interest. Throughout the text we refer to linearly independent ODEs as independent ODEs.
It should be noted that over the years the naming of a saddle-node bifurcation point has become somewhat convoluted, in other literature it is also referred to as a turning point, fold bifurcation point, or limit point bifurcation [71]. In a bistable system, two saddle-node bifurcation points delimit the range of the signal (or bifurcation parameter) for which two stable branches of steady states exist. It is this bistability phenomenon that we are particularly interested in identifying. An example scenario for bistability is depicted in Figure 2.3, where the parameter being varied for the bifurcation analysis is the signal of the reaction network, the solid blue line denotes a stable branch, and the dashed blue line represents an unstable branch.

**Definition 2.6** ([72])**.** Saddle-node
When considering an $n-$dimensional system of ODEs, $F$, we say that $u_0 \in \mathbb{R}^n$ is a saddle-node for $F : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ at $\epsilon_0$ if $F(u_0, \epsilon_0) = 0$, the linear transformation $D_u F(u_0, \epsilon_0) : \mathbb{R}^n \to \mathbb{R}^n$ has zero as an eigenvalue with algebraic multiplicity of one, and all other eigenvalues have nonzero real parts, where $D_u F(u_0, \epsilon_0)$ denotes the directional derivative of $F$ with respect to $u$.

Theorem 3 contains some technical terminology that requires a brief introduction. The smoothness of a function depends on the number of continuous derivatives that exist over the function's domain. The range of a matrix is the span of its column vectors. The kernel (or nullspace) of a matrix is the linear subspace of the domain of the map which is mapped to the zero vector. Lastly, The Fréchet derivative is the generalized form of the derivative of a real-valued function. Further details on these mathematical concepts can be found in the original text [72] and elsewhere.

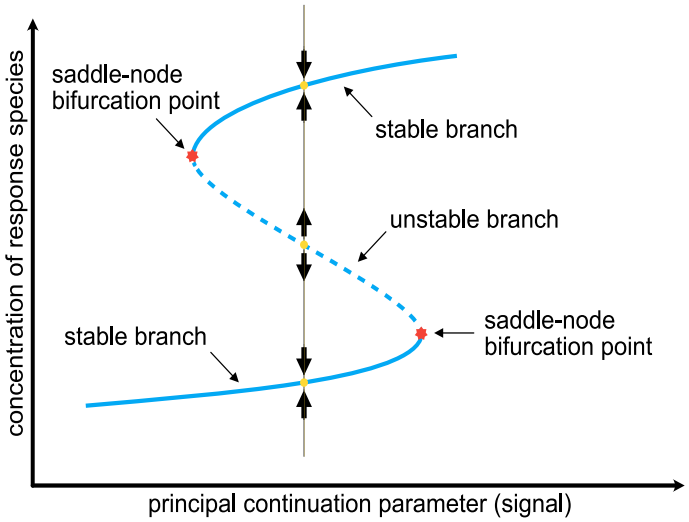



Figure 2.3: Bifurcation diagram illustrating bistability. The solid blue and dashed blue lines indicate stable and unstable branches, respectively. The figure also highlights key characteristics of bistability such as a stable and unstable point and a saddle-node bifurcation.

**Theorem 3** ([72])**.** Saddle-node Bifurcation Theorem

Suppose that $F : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is a smooth function, $u = u_0$ is a saddle-node for $F$ at $\epsilon = \epsilon_0$, and the kernel of the linear transformation $D_u F(u_0, \epsilon_0) : \mathbb{R}^n \to \mathbb{R}^n$ is spanned by the nonzero vector $k \in \mathbb{R}^n$. If $D_\epsilon F(u_0, \epsilon_0) \in \mathbb{R}^n$ and $D_{uu} F(u_0, \epsilon_0)(k, k) \in \mathbb{R}^n$ are both nonzero and both not in the range of $D_u F(u_0, \epsilon_0)$, then there is a saddle-node bifurcation at $u = u_0$. Here we define $D_{uu} F(u_0, \epsilon_0)(k, k)$ as the second Fréchet derivative of $F$ evaluated at $(u_0, \epsilon_0)$ in the directions given by $k$ and $k$.

## 2.2 Uniterminal approach for the discovery of bistability

The search for bistability in networks with mass conservation and mass action kinetics has been an area of study for some time. One of the newer approaches to finding bistability in these types of networks was presented in [12]. For brevity we will refer to this method as the uniterminal approach. This particular method attempts to find bistability by constructing an efficient optimization problem to find a saddle-node and then uses numerical continuation to confirm if the

saddle-node is a limit point. This approach is of considerable interest because it allows one to directly put bounds on values of the parameters of the network, such as the species' concentrations and reactions.

In this section we provide a short description of the mass conservation approach in [12] and then consider an alternative version of said approach that allows one to consider a general network structure. Using the structure of the network, the molecularity of the species, and CRNT, one can construct several matrices that describe the reaction network. If the network is uniterminal, then [12] provides an approach that uses these matrices to construct the locus of equilibria of the system of ODEs, denoted by $\mathcal{H}(\mathbf{c}, \boldsymbol{\alpha}, \mathbf{r}) = 0$ and is referred to as the equilibrium manifold. Where the deficiency parameters $\alpha_1, \ldots, \alpha_\delta$ are the linear coefficients of the $M - \ell$ linearly independent algebraic equations given by

$$\mathcal{H}(\mathbf{c}, \boldsymbol{\alpha}, \mathbf{r}) = A\psi - \sum_{i=1}^{\delta} \alpha_i \omega_i = 0. \tag{2.4}$$

In the description of the equilibrium manifold, $\delta$ is the deficiency of the network, $\omega_1, \ldots, \omega_\delta$ are basis elements for the deficiency subspace, $A$ is the kinetic constant matrix, and $\psi$ is a vector representing the mass action monomial.

Using the equilibrium manifold and the mass conservation matrix $B$, [12] then forms the matrix $G$ (where $D_x F$ denotes the Jacobian of $F$ with respect to $x$)

$$G(\mathbf{c}, \boldsymbol{\alpha}, \mathbf{r}) = \begin{pmatrix} D_{\mathbf{c}}\mathcal{H} & D_{\boldsymbol{\alpha}}\mathcal{H} \\ B^T & \mathbf{0} \end{pmatrix}.$$

Once $G$ is formed, the following optimization problem can be constructed

$$\begin{aligned} &\text{Minimize} && F(\mathbf{x}) \\ &\text{subject to:} && \mathcal{H}(\mathbf{c}, \boldsymbol{\alpha}, \mathbf{r}) = 0, \\ &&& \operatorname{rank}(G(\mathbf{c}, \boldsymbol{\alpha}, \mathbf{r})) = N + \delta - 1, \\ &&& \operatorname{rank}(D_{\mathbf{c}}\mathcal{H}(\mathbf{c}, \boldsymbol{\alpha}, \mathbf{r})) = \min(N, M - \ell), \\ &&& \mathbf{c} > 0 \quad \mathbf{c} \in \mathbb{R}^N, \\ &&& \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \quad (\mathbf{x}_L, \mathbf{x}_U \in \mathbb{R}^{R+\lambda}). \end{aligned}$$

By minimizing $F(\mathbf{x}) = det(G(\mathbf{c}, \boldsymbol{\alpha}, \mathbf{r}))^2$ and producing an objective function value equal to zero, this method yields a decision vector that produces either a saddle-node or a limit point for the ODE system representing the reaction network. To determine if a saddle-node or a limit point was found, numerical continuation is carried out on the ODE system with kinetic constants and species' concentrations provided by the optimization procedure. Assumptions of the uniterminal approach:

1. Every reaction is endowed with mass action kinetics.
2. The network admits a strictly positive steady state (where the concentrations of all the species are positive).
3. The network is uniterminal.

### 2.3 A general numerical approach for detecting switch-like bistability

Although the optimization approach presented in Section 2.2 in combination with hybrid optimization solvers [73] works quite efficiently, it is limited to networks that are uniterminal. This constraint is a direct result of assumptions made in the formulation of the optimization problem. More specifically, by assuming that the network is uniterminal, the approach is able to form a basis for the deficiency subspace, see section "Deficiency and equilibrium manifold" in [12]. A consequence of this result is the ability to form a square system of equations that define the equilibrium manifold of the ODE system that is compatible with the reaction polyhedron [74]. Furthermore, since this system is square, the approach can construct sufficient conditions for a saddle-node in the presence of mass conservation by minimizing the system's determinant (a scalar value available for only square systems), see section "Sufficient conditions for a saddle-node in presence of mass conservation" in [12]. To address this limiting condition and extend the reach of this optimization approach, we construct a general technique that can investigate the bistability of reaction networks regardless of their terminality.

The general approach (as depicted in Figure 2.4) uses the network structure to construct an optimization problem that searches for a saddle-node. This optimization problem differs from the

uniterminal approach as it searches for a saddle-node using direct ODE stability analysis, rather than relying on constrictive assumptions. In essence, the optimization problem consists in minimizing an objective function represented by the squared determinant of the Jacobian, so that a minimal value of zero guarantees the presence of a zero eigenvalue. Once a set of parameters conferring a saddle-node is found by this minimization, then production of a bifurcation diagram is attempted in one of two ways. One way is based on a well-established numerical continuation technique. Alternatively, the ODE system can be directly simulated from different initial conditions to compute the dose-response curve. In this case the dose-response curve serves as a surrogate of a bifurcation diagram except it does not contain an unstable branch. This option, although more computationally intensive, allows one to determine if the system is bistable in cases where numerical continuation is not possible due to an ill-conditioned Jacobian.
In continuance of 2.1 we now describe the general approach using the Edelstein chemical reaction network. Now that we have our ODE system in terms of our input signal (or bifurcation parameter), we can now continue by constructing the expressions that are necessary to check the requirements of Definition 2.6. To do this, the first item we consider is the statement that $F(u_0, \epsilon_0) = 0$, this denotes that the ODE system is at a steady state. Thus, we must ensure that $\dot{\boldsymbol{\varsigma}} = \mathbf{0}$. One straightforward way of doing this is to add the term $\sum_{i=1}^{s} [\dot{\boldsymbol{\varsigma}}_i(\mathbf{k}, \boldsymbol{\varsigma})]^2$ to the objective function (equation (2.6) in Section 2.3.2), which represents the sum of the squared derivatives. Although this formulation is robust, the trade-off includes increased computational time.

To avoid this issue, we suggest an alternative, based on symbolic constraints, formulation of the optimization problem. In this section, we propose splitting the kinetic constants into two sets, one set that is free ($\mathbf{k} \in \mathbb{R}^{R-s}$) and another set that will be fixed using symbolic expressions ($\tilde{\mathbf{k}} \in \mathbb{R}^{s}$). We analytically solve for specific kinetic constants, $\tilde{\mathbf{k}}$, to enforce a steady state. By solving for these expressions analytically, the domain of feasible points is reduced, providing a simpler optimization problem. This can be performed systematically because we are using mass action kinetics to form our ODEs. Furthermore, we are always guaranteed a unique analytical expression for a steady state (in terms of a particular choice of kinetic constants). To see this, consider the

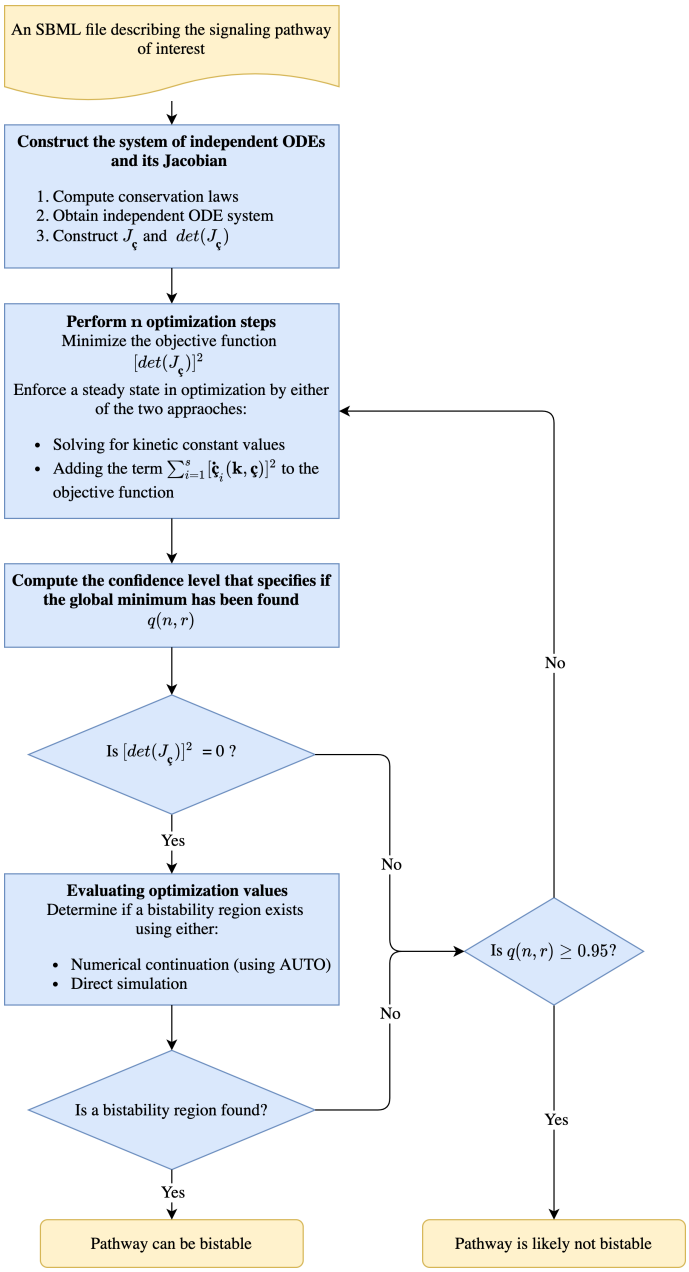



Figure 2.4: Workflow of the general approach for bistability detection in mass action chemical reaction networks. The approach is based on finding conditions that produce a Jacobian evaluated at a steady state, which has one zero eigenvalue. The symbol $\mathbf{ç}$ denotes the independent species' concentrations. A confidence level defining if a global minimum of the optimization problem has been found is denoted as $q(n,r)$.

stoichiometic matrix $S$, which has columns corresponding to the reactions that have one-to-one correspondence to the kinetic constants. As stated in Subsection 2.1.3, $\text{rank}(S) = s \leq R$, for any given reaction network. By definition, $s$ corresponds to the number of linearly independent columns of $S$ and since the columns of $S$ correspond to the kinetic constants, we are provided with a set of kinetic constants that form a linearly independent system of equations.

In practice, it is quite simple to determine the expressions for $\tilde{\mathbf{k}}$, that should be chosen to create a linear system. One particular way to choose these kinetic rate constants is to put $S$ into row reduced echelon form (RREF) and choose those columns that contain pivots. In our example:

$$S = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{cccccc} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 \end{array} \\ \begin{pmatrix} 1 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 & -1 \\ 0 & 0 & 1 & -1 & -1 & 1 \end{pmatrix} \end{array} \xrightarrow{\text{RREF}} \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{cccccc} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 \end{array} \\ \begin{pmatrix} 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array},$$

the pivots are in the columns 1 and 3, resulting in $\tilde{\mathbf{k}} = (k_1, k_3)^T$.

Given we are looking for a steady state, we must consider the following linear system:

$$U\tilde{\mathbf{k}} = \mathbf{b},$$

Here $U \in \mathbb{R}^{s \times s}$ corresponds to the coefficient matrix produced from $\dot{\mathbf{ç}}$ for a particular choice of $\tilde{\mathbf{k}}$. Vector $\mathbf{b} \in \mathbb{R}^s$ are those terms of $\dot{\mathbf{ç}}$ that do not contain kinetic constants from $\tilde{\mathbf{k}}$. For our example we have the following system for our steady state:

$$\begin{pmatrix} c_1 & -c_1(C_1 - c_3) \\ 0 & c_1(C_1 - c_3) \end{pmatrix} \begin{pmatrix} k_1 \\ k_3 \end{pmatrix} = \begin{pmatrix} k_2 c_1^2 - k_4 c_3 \\ (k_4 + k_5)c_3 - k_6(C_1 - c_3) \end{pmatrix}.$$

Solving for $k_1$ and $k_3$ provides necessary conditions for a steady state

$$\begin{aligned} k_1 &= \frac{k_2 c_1^2 + k_5 c_3 - k_6(C_1 - c_3)}{c_1} \\ k_3 &= \frac{(k_4 + k_5)c_3 - k_6(C_1 - c_3)}{c_1(C_1 - c_3)}. \end{aligned}$$

Note that for some systems, this may force a kinetic rate constant equal to zero. However, other choices of $\tilde{\mathbf{k}}$ can yield nonzero values. To account for these cases, if zero values are found, we exhaustively search through all distinct combinations of independent columns to choose $\tilde{\mathbf{k}}$ combinations and search for ones that yield no zero entries. If no combination provides nonzero values this means that no steady state exists when $\mathbf{k} > 0$ and $\mathbf{c} > 0$. In such a scenario we suggest checking the Deficiency Zero and One theorems [65]. To obtain a network that can assume a steady state with $\mathbf{k} > 0$ and $\mathbf{c} > 0$, one can consider removing the reactions that have kinetic constants equal to zero in the aforementioned computation.

### 2.3.1 Ensuring a zero eigenvalue

From the above results, we have necessary conditions for a steady state. Thus, the next item we must satisfy is that $D_u F(u_0, \epsilon_0)$ must have a zero eigenvalue with a multiplicity of one and all other eigenvalues have nonzero real parts, as stated in Definition 2.6. Effectively, $D_u F(u_0, \epsilon_0)$ is the Jacobian of the right hand side of our independent ODE system with respect to $\mathbf{ç}$, we denote this as $J_{\mathbf{ç}}$. Given that we would like to formulate this as an optimization problem, it is easy to see that satisfying the criteria that an eigenvalue of zero must have a multiplicity of one and all other eigenvalues be nonzero, will create an expensive optimization problem because we would have to calculate the eigenvalues each time the objective function is evaluated. For this reason, we will simply search for a zero eigenvalue and then check afterwards if the eigenvalue criteria are satisfied.

To find a zero eigenvalue, consider the eigenvalue problem with eigenvalue $z \in \mathbb{C}$ and corresponding eigenvector $\mathbf{v} \in \mathbb{C}^s \backslash \{0\}$:

$$J_{\mathbf{ç}}\mathbf{v} = z\mathbf{v}.$$

The characteristic polynomial can then be formed by taking the determinant and setting it equal to zero:

$$det(J_{\mathbf{ç}} - zI) = 0.$$

However, since $z = 0$, we have the following problem that reassures us that at least one eigenvalue is equal to zero:

$$det(J_{\dot{\mathbf{c}}}) = 0.$$

It is this criteria that we will use to formulate our optimization problem.

### 2.3.2 Searching for a Saddle-node using optimization

Using the definitions from the previous steps we can create the optimization problem in two alternative ways. In both formulations, the optimization problems search for species' concentrations $\mathbf{c}$ and kinetic rate constants $\mathbf{k}$ that provide a system in a steady state ($\dot{\mathbf{c}} = 0$) and at least one zero eigenvalue of the associated Jacobian $J_{\dot{\mathbf{c}}}$. One that implicitly enforces a steady state by solving for kinetic constants:

$$\begin{aligned} &\text{Minimize} && [det(J_{\dot{\mathbf{c}}})(\mathbf{x})]^2 \\ &\text{subject to:} && \\ & && \mathbf{c}_L \leq \mathbf{c} \leq \mathbf{c}_U \quad \mathbf{c} \in \mathbb{R}^N, \\ & && \mathbf{k}_L \leq \mathbf{k} \leq \mathbf{k}_U \quad \mathbf{k} \in \mathbb{R}^R. \end{aligned} \tag{2.5}$$

In this version of the optimization problem $\mathbf{c}_L$ and $\mathbf{c}_U$ are the lower and upper bounds for the species' concentrations, respectively and similarly, $\mathbf{k}_L$ and $\mathbf{k}_U$ are the lower and upper bounds for the kinetic rate constants, respectively. The decision vector $\mathbf{x} = (\{\mathbf{k}\} - \{\tilde{\mathbf{k}}\}, \{\mathbf{c}\})^T$, where $\{\mathbf{k}\} - \{\tilde{\mathbf{k}}\}$ corresponds to the set of kinetic rate constants in $\mathbf{k}$ that are not in $\tilde{\mathbf{k}}$ and $\{\mathbf{c}\}$ is the set of species' concentrations. The kinetic rate constants $\tilde{\mathbf{k}}$ are then found by using the solutions found from solving the linear system, which reassure a steady state occurs, and the conservation constants $C_1, \ldots, C_\lambda$ are found using the conservation laws.

Another formulation enforces a steady state by explicitly requiring that the derivatives of the concentrations be zero through an additional term in the objective function:

$$\begin{aligned} &\text{Minimize} && [det(J_{\dot{\mathbf{c}}})(\mathbf{k}, \dot{\mathbf{c}})]^2 + \sum_{i=1}^{s} [\dot{\mathbf{c}}_i(\mathbf{k}, \dot{\mathbf{c}})]^2 \\ &\text{subject to:} && \\ & && \mathbf{c}_L \leq \mathbf{c} \leq \mathbf{c}_U \quad \mathbf{c} \in \mathbb{R}^N, \\ & && \mathbf{k}_L \leq \mathbf{k} \leq \mathbf{k}_U \quad \mathbf{k} \in \mathbb{R}^R. \end{aligned} \tag{2.6}$$

This results in a more complex objective function, which is due to the addition of more nonlinear terms in the objective function and a higher dimensional decision vector. However, this approach is more robust as it samples from a broader space of solutions.

Since the approach is aimed towards analysis of biological pathways, we can apply bounds to the optimization problem that are based on prior knowledge. To bound the species' concentrations, one can choose the typical range of protein concentrations in a cell, $5 \times 10^{-13}$ to $5 \times 10^{-7}$ M. The complex formation kinetic constants can be bounded between $10^4$ and $10^8$ $M^{-1}s^{-1}$. A common range for complex dissociation constants is from $10^{-5}$ to $10^{-3}$ $s^{-1}$. Finally, the enzyme catalysis kinetic constants range from $10^{-3}$ to 1 $s^{-1}$. These ranges are the defaults in our Python package CRNT4SBML. However, they can be overwritten by the user to more narrow ranges if some more accurate prior knowledge exists. For further details we refer the reader to the corresponding CRNT4SBML documentation page.

Due to their nature, like the optimization problem in [12], (2.5) and (2.6) are non-convex and multi-modal. For this reason, a global optimization procedure should be used to search for the optimal solution of (2.5) and (2.6). For the results obtained throughout the Chapter, we utilize the global optimization algorithm Dual Annealing, starting from different random initial starting points that are bounded by the species' concentrations and kinetic constant values. Dual Annealing is an optimization algorithm that combines Simulated Annealing [75] and additionally applies a local search on accepted locations [76]. In particular, we utilize the Nelder-Mead simplex algorithm for all local searches. Given that we are looking specifically for a zero eigenvalue, the objective function of the optimization should obtain a minimum of zero. If a zero is found one should additionally provide a check for i) a saddle-node using Definition 2.6 and discard points that produce more than one eigenvalue that is zero (which might indicate a codim 2 bifurcation such as the Bogdanov-Takens bifurcation [77]), and ii) a saddle-node bifurcation point using Theorem 3. This allows one to remove unnecessary runs of numerical continuation. Although explicitly checking the criteria for a saddle-node bifurcation can reduce the runtime of the approach, it should be noted that it is sufficient to conduct numerical continuation.

### 2.3.3 Bayesian Stopping rule

If the optimization satisfies the condition for a saddle-node, one can state that a saddle-node exists. However, the inverse is not true: if we can't find conditions that result in a zero eigenvalue, this doesn't guarantee that a saddle-node does not exist. This is a consequence of using stochastic optimization instead of deterministic methods (based for example on interval analysis), which will provide this guarantee [78]. The issue with methods such as these is that they become computationally intractable for mass action reaction systems with more than two or three parameters, whereas stochastic optimization provides in general a good overall efficiency [79]. To address this uncertainty we compute the probability that the achieved minimum value is equal to the true global minimum. The probability is calculated using a slightly modified version of the unified Bayesian stopping rule in [80] and Theorem 4.1 of [81], where the rule was first established.

For $n$ starting optimization points, let $f_k$ be the achieved local minimum for the $k$-th decision vector, $f^*$ the true global minimum and $\tilde{f} = min\{f_1, \ldots, f_n\}$. Our objective is to estimate the probability that $\tilde{f}$ is $f^*$. Let $\alpha_k$ and $\alpha^*$ denote the probabilities that a single run of the optimization routine has converged to $f_k$ and $f^*$, respectively. Assuming that $\alpha^* \geq \alpha_k$ for all local minimum values $f_k$ we may then estimate the lower bound of the probability that $\tilde{f} = f^*$ is as follows:

$$Pr[\tilde{f} = f^*] \geq q(n, r) = 1 - \frac{(n + a + b - 1)!(2n + b - r - 1)!}{(2n + a + b - 1)!(n + b - r - 1)!},$$

Here $a$ and $b$ are the parameters of the Beta distribution $\beta(a, b)$, where we use $a = 1$ and $b = 5$ as suggested in [80]. The term $q(n, r)$ is the confidence level, where $r$ is the number of $f_k$ for $k = 1, \ldots, n$ that are in the neighborhood of $\tilde{f}$.

We say that $f_k$ is in the neighborhood of $\tilde{f}$ if the relative difference of $f_k$ and $\tilde{f}$ is less than 1%:

$$\frac{|\tilde{f} - f_k|}{\tilde{f}} \leq 10^{-2}.$$

Given the formulation of the optimization, the lowest possible minimal value is zero. Thus, if $\tilde{f}$ reaches a numerical zero value then we set $q(n, r) = 1.0$, skipping the computation of $q(n, r)$. Conventionally $q(n, r) \geq 0.95$ is considered an acceptable confidence level to make the conclusion that $\tilde{f}$ is the global minimum of the objective function.

### 2.3.4 Demonstrating the Bayesian Stopping Rule

Now that we have established an approach to estimate the probability that the minimum objective function value is the global minimum, we will demonstrate this by using an example. Consider the reaction network provided in Figure 2.5. By the Deficiency Zero Theorem of [4] it is known

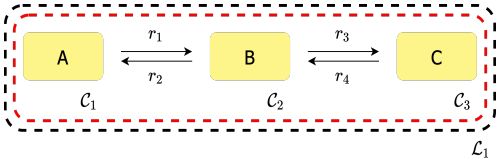



Figure 2.5: C-graph of a simple non-bistable network.

that the network cannot be bistable and there is only one equilibrium. Because of the simplicity of the reaction network we can compute the exact value of the global minimum (e.g. using Mathematica's MinValue [82]). Letting our reaction coefficients and species' concentrations be bounded between 0.01 and 100, the lowest possible value is $f^* = 9.0e - 8$. Performing 100 iterations of the optimization routine, we obtain $\tilde{f} = 9.000005389758511e - 8$ with a confidence level of $q(n, r) = 0.9999999995824618$. Since $q(n, r)$ is greater than 0.95, we would conclude that the achieved value of $\tilde{f}$ is the global minimum. For this reason, we can reject the possibility that a zero value of the objective function can be achieved.

### 2.3.5 Numerical Continuation and Direct Simulation

Once the optimization problem is solved, we have an independent ODE system with a steady state and at least one zero eigenvalue of the Jacobian. To ensure that this is a saddle-node bifurcation we need to check if this is the only zero eigenvalue and that Theorem 1 is satisfied. In practice, we check directly for the presence of the bifurcation diagram with switch-like behavior. The

existence of such a bifurcation diagram rules out undesired findings such as a Jacobian with multiple zero eigenvalues or eigenvalues with positive real parts. To generate the bifurcation diagram, we utilize either the technique of numerical continuation or directly simulate the ODEs. When performing numerical continuation, we use the values provided by the optimization routine and then leverage the established tool AUTO 2000 [83]. It is made accessible through the libroadrunner python library and its extension rrplugins [84]. To demonstrate numerical continuation, we use the following values provided by the optimization routine, which provide Figure 2.6:

$$k_1 = 0.05024595, k_2 = 0.01029913, k_3 = 0.03592955, k_4 = 0.01027423, k_5 = 0.01272131,$$

$$k_6 = 0.01006076, c_1 = 1.48685156, c_2 = 2.07275022, c_3 = 5.72214036, \text{ and } C_1 = 7.79489058.$$

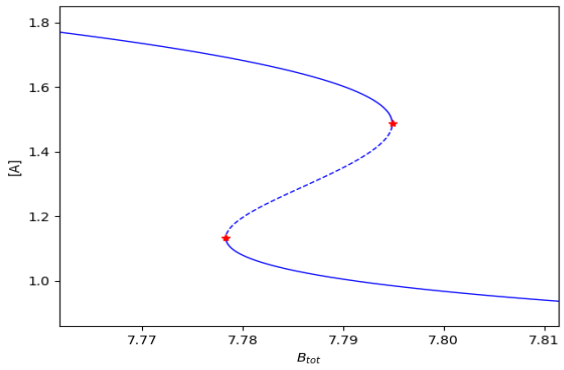



Figure 2.6: Bifurcation diagram of the Edelstein network example. The bifurcation diagram was created using the software AUTO 2000, which utilizes numerical continuation. Red markers represent the found saddle-node bifurcation points, the stable branches are solid blue lines, and the unstable branch is a dashed blue line.

It should be noted that it is possible that the optimization routine finds a certain combination of kinetic rate constant values that force the Jacobian of the system to be ill-conditioned or even singular, even if species' concentrations are varied. The situation when the Jacobian is always

singular precludes the numerical continuation approach. To overcome this type of situation we offer the option of direct simulation, which varies the user defined signal and initial conditions values for the ODE system and then integrates the ODEs until a steady state occurs. The steady state is obtained when the species' concentrations do not change between the ODE integration steps, within a predefined tolerance (here 1e-06). Given the direct simulation method is numerically integrating the system of ODEs, this method will often take longer than the numerical continuation routine. Although this is the case, direct simulation is more robust and may be able to provide a bifurcation diagram when numerical continuation cannot.

Considering high and low concentrations of species $A$ (response) for the initial value, we obtain the ODE simulations in Figure 2.7 by direct simulation, where $B_{tot} = C_1$. If we then consider the

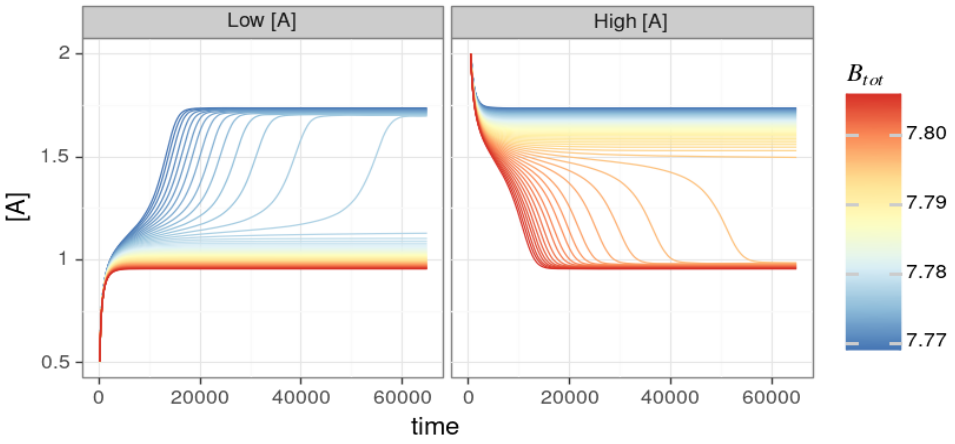



Figure 2.7: Simulation of the ODE system for the Edelstein network using different starting values of $B_{tot}$ and $[A]$. The system converges to two equilibrium states; a "lower" one at $[A] \approx 1$ and a "higher" one at $[A] \approx 1.75$. If the system starts with a low concentration of $A$ a switch between "low" and "high" equilibrium states happens at smaller concentration of $B_{tot}$. Vice versa, if the system starts with a high concentration of $A$, the switch between the equilibrium states happens at higher concentrations of $B_{tot}$.

left and right plot at time 65,000 as the equilibrium, we can plot the values of $B_{tot}$ vs the concentration of species $A$. The dose-response diagram obtained by direct simulation depicted in Figure 2.8 mirrors the bifurcation diagram obtained by numerical continuation in Figure 2.6, thus cross-validating both approaches.

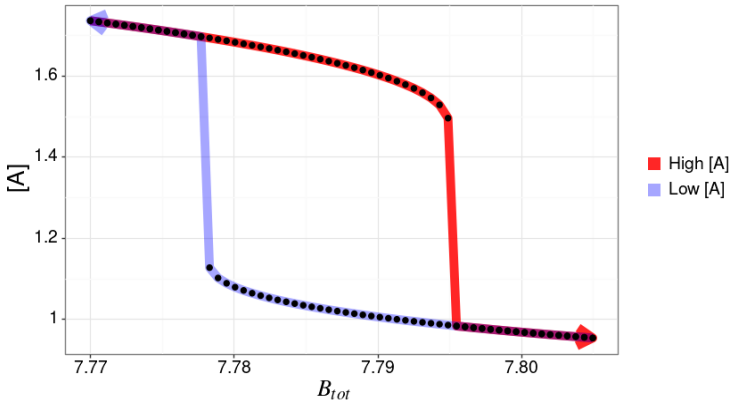
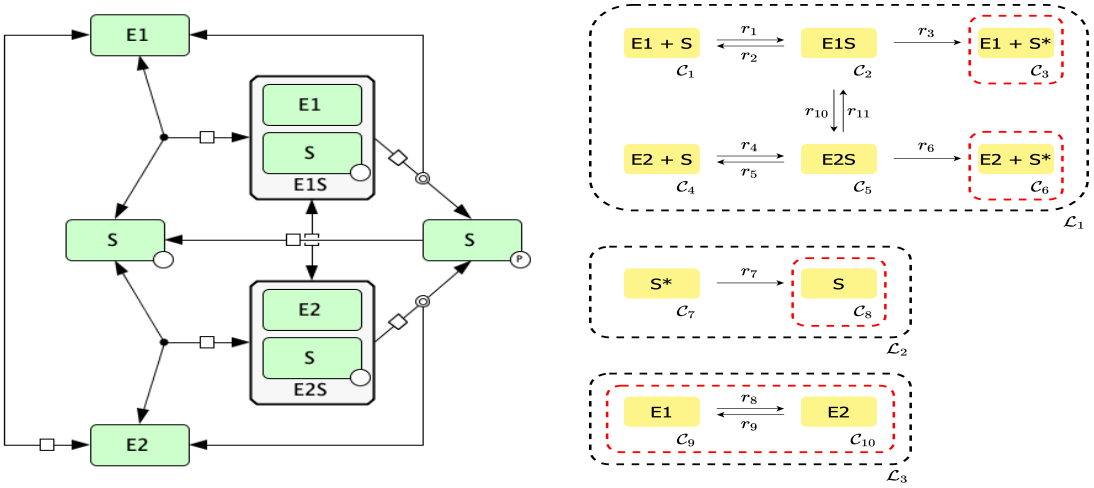



Figure 2.8: Bifurcation diagram of the example Edelstein network. It is a cross-section of the Figure 2.7 data at the time point when the system reaches the equilibrium (65,000 seconds for the given parameters). The equilibrium concentration of $A$ is plotted for both the "high" and "low" portions of the simulation, highlighted in red and light blue, respectively. The arrows on the plot indicate the direction of change in $B_{tot}$.

### 2.3.6 Bistability in a biterminal Futile Signaling Cycle

Using the general technique established in the Methods section, we will continue by considering a non-uniterminal example. For this example we will be considering a key reaction network found predominantly in eukaryotic signaling systems, namely a futile signaling cycle, that exhibits bistability when featuring a two-state kinase, as presented in [85]. This reaction network is represented in Figure 2.9. From Figure 2.9(b), one can see that the linkage class $\mathcal{L}_1$ has two terminal strong linkage classes (due to reactions $r_3$ and $r_6$). Thus, we have a biterminal linkage class and we cannot apply the theory presented in [12]. Letting

$$c_1 = [S], c_2 = [E1], c_3 = [E1S], c_4 = [S^*], c_5 = [E2], c_6 = [E2S],$$




(a) (b)

Figure 2.9: Futile signaling cycle. (a) diagram of reaction network and (b) its C-graph representation. E1 and E2 indicate alternative forms of a kinase enzyme. S is a substrate. S* is a substrate in a phosphorylated form (labeled with p on panel (a). E1S and E2S are the enzyme:substrate complexes.

and performing the steps outlined in the previous section, we obtain the following independent ODEs

$$\dot{c}_3 = k_1(E_{tot} - c_5 - c_3 - c_6)(S_{tot} - c_4 - c_3 - c_6) - k_2c_3 - k_3c_3 - k_{10}c_3 + k_{11}c_6$$

$$\dot{c}_4 = -k_7c_4 + k_3c_3 + k_6c_6$$

$$\dot{c}_5 = -k_4c_5(S_{tot} - c_4 - c_3 - c_6) + k_5c_6 + k_6c_6 + k_8(E_{tot} - c_5 - c_3 - c_6) - k_9c_5$$

$$\dot{c}_6 = k_4c_5(S_{tot} - c_4 - c_3 - c_6) - k_5c_6 - k_6c_6 + k_{10}c_3 - k_{11}c_6,$$

with conservation laws

$$E_{tot} = c_2 + c_5 + c_3 + c_6$$

$$S_{tot} = c_1 + c_4 + c_3 + c_6,$$

and fix particular kinetic rate constants to expressions (as described in the previous section) in order to ensure a steady state in the ODE system

$$
\begin{aligned}
k_1 &= \frac{k_7 c_4 + k_2 c_3 - k_6 c_6 + k_{10} c_3 - k_{11} c_6}{(E_{tot} - c_5 - c_3 - c_6)(S_{tot} - c_4 - c_3 - c_6)} \\
k_3 &= \frac{k_7 c_4 - k_6 c_6}{c_3} \\
k_4 &= \frac{k_5 c_6 + k_6 c_6 - k_{10} c_3 + k_{11} c_6}{c_5 (S_{tot} - c_4 - c_3 - c_6)} \\
k_8 &= \frac{k_9 c_5 - k_{10} c_3 + k_{11} c_6}{E_{tot} - c_5 - c_3 - c_6}.
\end{aligned}
$$

After performing the optimization routine, we obtain the following values from our decision vector

$$k_1 = 0.738028, k_2 = 3.316128, k_3 = 0.001800, k_4 = 0.050541, k_5 = 0.029997,$$

$$k_6 = 0.53685, k_8 = 2.310671, k_9 = 0.001081, k_{10} = 0.008850, k_{11} = 3.437878,$$

$$k_7 = 0.065141, c_3 = 957.287983, c_4 = 605.457182, c_5 = 136.645046, c_6 = 70.255623,$$

$$E_{tot} = 1265.114144, \text{ and } S_{tot} = 1672.513735.$$

Using the values for the species' concentrations and kinetic rate constants found, we can now begin to construct the dose-response curve. This is done by using several different concentrations of species $S^*$ for the initial value and simulating the ODE system until it has reached a steady state. For this particular example, we obtain the dose-response diagram presented in Figure 2.10.

### 2.3.7 Bistability in a biterminal Prion/Double Phosphorylation motif

Next we consider the hypothetical mechanism for prion-like conformation conversion between two states of a protein described in Figure 2.11. Kinase can be in two conformations E1 and E2. Conversion between E1/E2 proceeds through a prion-like mechanism, that is catalyzed by the enzyme E in the corresponding conformation. Only one conformation of kinase, E2, is active and

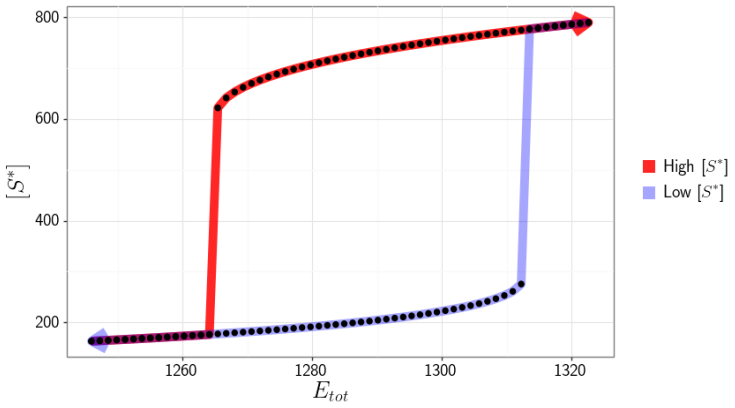
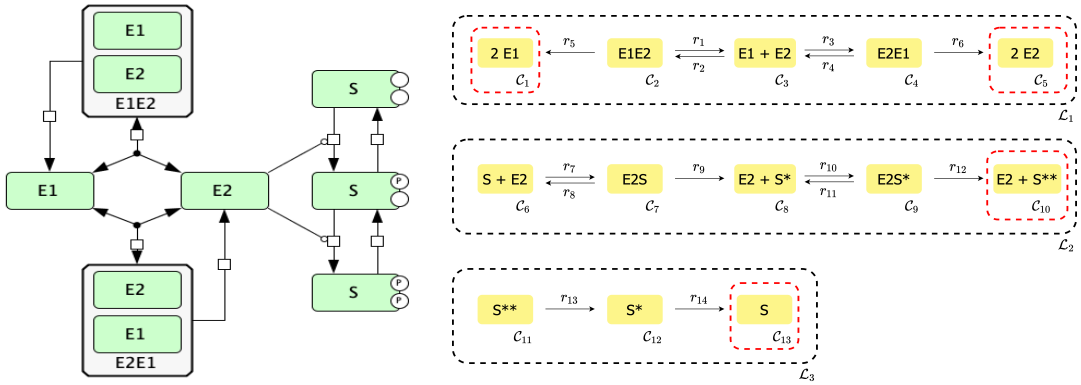



Figure 2.10: Dose-response diagram exhibiting switch-like behavior produced by a futile signaling cycle. Dots represent the equilibrium that the system converges to for individual simulations. The red and light blue paths correspond to high and low initial concentrations of $[S^*]$, respectively.

phosphorylates substrate S in two-steps. From inspection, one can see that $\mathcal{L}_1$ is a linkage class with two terminal strong linkage classes (due to reactions $r_5$ and $r_6$). Thus, we have a biterminal linkage class and we cannot apply the theory presented in [12]. Letting

$$c_1 = [E1], c_2 = [E2], c_3 = [E1E2], c_4 = [E2E1], c_5 = [S],$$

$$c_6 = [S^*], c_7 = [SE2], c_8 = [S^{**}], c_9 = [S^*E2],$$




(a) (b)

Figure 2.11: Prion/Double Phosphorylation motif with prion-like conformation conversion between the two states of the same protein. One of the conformations is assumed to be an active kinase. (a) diagram of reaction network and (b) its C-graph representation. E1 and E2 indicate alternative forms of a kinase enzyme. Only E2 is active. S, S* and S** (labeled with p on panel (a)) is a substrate in unmodified, phosphorylated, and doubly phosphorylated forms. E1E2 and E2E1 are the protein complexes that mediate conversion from one form into another.

and performing the steps outlined in the previous section, we obtain the following independent ODEs

$$\dot{c}_2 = k_1c_3 - c_2(k_2 + k_3)(E_{tot} - c_2 - c_9 - c_7 - 2c_3 - 2c_4) + (k_4 + 2k_6)c_4$$
$$+ (k_9 + k_8)c_7 - k_7c_2(S_{tot} - c_6 - c_8 - c_9 - c_7) + (k_{12} + k_{11})c_9 - k_{10}c_6c_2$$

$$\dot{c}_3 = k_2c_2(E_{tot} - c_2 - c_9 - c_7 - 2c_3 - 2c_4) - k_1c_3 - k_5c_3$$

$$\dot{c}_4 = k_3c_2(E_{tot} - c_2 - c_9 - c_7 - 2c_3 - 2c_4) - k_4c_4 - k_6c_4$$

$$\dot{c}_6 = k_9c_7 - k_{14}c_6 + k_{11}c_9 - k_{10}c_6c_2 + k_{13}c_8$$

$$\dot{c}_7 = -k_9c_7 - k_8c_7 + k_7c_2(S_{tot} - c_6 - c_8 - c_9 - c_7)$$

$$\dot{c}_8 = k_{12}c_9 - k_{13}c_8$$

$$\dot{c}_9 = -k_{12}c_9 - k_{11}c_9 + k_{10}c_6c_2,$$

with conservation laws

$$S_{tot} = c_5 + c_6 + c_7 + c_8 + c_9$$

$$E_{tot} = c_1 + c_2 + 2c_3 + 2c_4 + c_7 + c_9.$$

An alternative way to ensure the steady state in the optimization problem is to directly enforce the time derivatives of the concentrations to be zero. That is, instead of constraining some of the kinetic constants with symbolic expressions, we enforce $\dot{\boldsymbol{\varsigma}}_i = 0$ for $i = 1, \ldots, s$ explicitly, where $\boldsymbol{\varsigma}_i$ denotes a concentration of an independent species in the objective function. Specifically, the objective function will have the additional term $\sum_{i=1}^{s} [\dot{\boldsymbol{\varsigma}}_i(\mathbf{k}, \boldsymbol{\varsigma})]^2$, which ensures a steady state. For details please refer to "Perform n optimizations" box in Figure 2.4 and equation (2.6) in Section 2.3.2. This approach can be helpful in cases where the Jacobian is ill-conditioned. Here, for demonstration purposes we choose not to fix the kinetic rate constants and instead use the aforementioned robust, but more computationally intensive approach. Performing the optimization routine, we obtain the following values from our decision vector

$$k_1 = 27.963833, k_2 = 2.417993, k_3 = 2.121228, k_4 = 48.342142, k_5 = 0.910340,$$

$$k_6 = 1.802118, k_7 = 17.019827, k_8 = 92.473965, k_9 = 0.021611, k_{10} = 0.782488,$$

$$k_{11} = 3.692336, k_{12} = 0.205743, k_{13} = 0.063297, k_{14} = 0.235401, c_1 = 14.749224,$$

$$c_2 = 18.117522, c_3 = 22.377604, c_4 = 11.304051, c_5 = 27.001718,$$

$$c_6 = 8.264281, c_7 = 90.016969, c_8 = 97.695329, c_9 = 30.056006,$$

$$S_{tot} = 253.034303, \text{ and } E_{tot} = 220.303031.$$

Using the values for the species' concentrations and kinetic rate constants found, we can now directly compute the bifurcation diagram with continuation methods or alternatively obtain the dose-response curve by direct simulation. This is done by using several different concentrations of species $S^{**}$ for the initial value and simulating the ODE system until it has reached a steady state. For this particular example, we obtain the dose-response diagram presented in Figure 2.12.

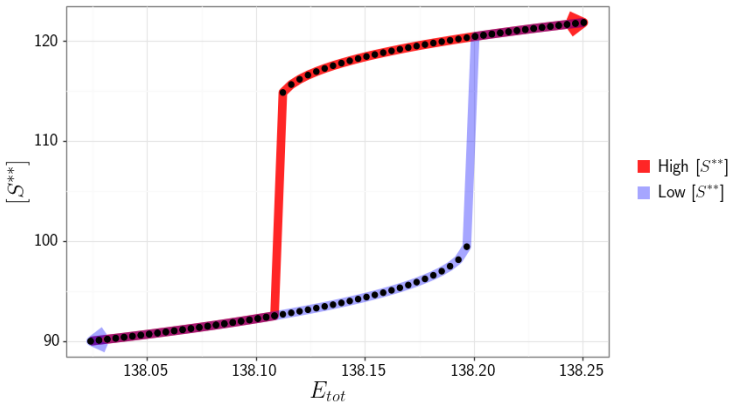



Figure 2.12: Dose-response diagram exhibiting switch-like behavior produced by the Prion/Double Phosphorylation model.

## 2.4 CRNT4SBML Python Package

Although the methods we have developed in this chapter have proven their usefulness, these methods are complicated from both pure and computational mathematics perspectives. Thus, their adoption is very limited amongst biologists. To bridge the gap between experimental biologists and the techniques covered and constructed in this Chapter, we have developed the Python package CRNT4SMBL. The source code for the package is available at github.com/PNNL-Comp-Mass-Spec/CRNT4SBML. Additionally, we have written an extensive amount of documentation (around 179 pages) on the package, which is available at crnt4sbml.readthedocs.io/en/latest/index.html. We now highlight key features of the package.

### 2.4.1 Importing Reaction Networks

Input is based on an SBML file representing the pathway of interest. Pathways can be created using a number of established tools such as CellDesigner [86]. To parse an SBML file CRNT4SBML uses the libsbml library [87], which allows for the extraction of the species and reactions. Once the file is parsed, construction of a C-graph and basic CRNT analysis is performed by the graph library NetworkX [88]. Creation of key CRNT matrices and vectors are then completed by leveraging the symbolic computation library SymPy [89].

### 2.4.2 Checking Deficiency Theory

CRNT4SBML provides methods that check the conditions for the Deficiency Zero and One theorems of CRNT. This allows one to quickly rule out those motifs that are guaranteed to not produce bistability. However, for a substantial proportion of realistic network motifs it has been found through empirical observations that these theorems cannot make an unequivocal decision, as provided in section 7 of [4].

### 2.4.3 Implementation of Mass Conservation and Semi-diffusive Approaches

To address more complicated pathways CRNT4SBML includes a full framework for the implementation of the mass conservation (uniterminal approach) and semi-diffusive approaches (a network with no conservation laws) of [12], which utilize optimization and numerical continuation. Within the mass conservation approach the decision vector is defined in terms of kinetic constants and concentrations. This feature improves usability over current implementations which require additional deficiency parameters that are not immediately clear to biochemists. In practice, these deficiency parameters are difficult to put bounds on and a particular choice can drastically affect the optimization solution.

### 2.4.4 Implementation of the General Approach

As we have highlighted in this chapter, the uniterminal approach of [12] has proven to be very effective. However, it fails to address all networks that contain conservation laws because it is limited to uniterminal networks. To address this, we constructed the general approach and we have also included this approach in CRNT4SBML. All steps of this approach have been automated within the package. For this method, we implement optimization, numerical continuation, and direct simulation routines that are specific to the general approach.

### 2.4.5 Built in Numerical Continuation and Optimization routines

Furthermore, we also implement a multi-start optimization routine and generalized it for both approaches. This routine allows the user to find multiple sets of kinetic constants and species'

concentrations that satisfy the optimization problem. To ensure that the kinetic constants and species' concentrations define chemical systems that are capable of bistability, we apply the method of numerical continuation. Numerical continuation allows for the identification of the stable and unstable branches of the ODE system, by varying the principle continuation parameter (PCP). The PCP in the continuation analysis represents the strength of the input signal and can be one of two choices. In the case of closed systems that comply with the mass conservation approach, the input signal is the total amount of the signal molecule (e.g. receptor ligand). For the semi-diffusive approach in the case of open systems, the PCP is a kinetic constant reflecting the rate of the key reaction (e.g. receptor activity upon ligand binding). The key guidance for defining the PCP is the identification of the most important parameter, be that total molecule amount or a rate constant, that serves as a signal to the motif. To conduct the continuation analysis, we leverage the established tool AUTO2000 [83]. AUTO2000 is made accessible through the libroadrunner library and its extension rrplugins [84]. Lastly, a built-in routine analyzes the output provided by AUTO2000 and examines which parameter sets produce bistability. If bistability is detected, a bifurcation diagram (PCP vs response) is created.

### 2.4.6 Concluding remarks on CRNT4SBML

CRNT4SBML has been written with the intent to provide an easily usable interface for core theories that help detect the existence of bistability in cell signaling pathways, in hopes that it will promote the unification of mathematical modeling and experimental design. The tool has been tested against several different network motifs which do and do not exhibit bistability, as provided in the examples section of the documentation for CRNT4SBML. Some of the notable examples that produce bistability include: autophosphorylation (Fig 1Ci in [12]), cycle with substrate inhibition [90], and double phosphorylation [91] motifs. The test cases that do not produce bistability include: the closed and open versions of the interferon binding process (Figure 4 in [12], the suicidal enzyme motif (Figure 1 Aii [12]), network 3.13 of [65], and example 3.D.3 of [65]. Further work will be focused on the inclusion of other core CRNT methods, which may

quickly preclude bistability in some user provided motifs.

## 2.5 Conclusion

We have developed a new general technique to detect bistability in any mass conserving reaction network. The general technique established builds on an existing approach that only allows for bistability detection in reaction networks that are uniterminal. This is achieved by first constructing an objective function using CRNT that is not dependent on the number of terminal strong linkage classes in each linkage class. Then, we formulate an optimization approach that searches for a saddle-node. Once a saddle-node is detected, the method performs either numerical continuation or direct simulation to identify if the particular set of parameters that produced the saddle-node also produce a saddle-node bifurcation. Lastly, if a saddle-node bifurcation is found, numerical continuation or direct simulation will elucidate whether or not there is another saddle-node bifurcation. The need to find two saddle-node bifurcation points is necessary as they confer back and forth switch-like behavior. Various examples provided verify the general technique and its ability to identify bistability.

Although the general technique is a useful method for the detection of bistability, there are certain difficulties it may encounter. The technique can become computationally intensive for large reaction networks with very high dimensional search spaces to be explored. To overcome this, we included an option that allows the user to perform the optimization routine using parallel computing techniques (enabled by the use of Open MPI). This is possible since the optimization routine starts from different independent initial starting points. Additionally, even if a large portion of the objective function's domain is explored, we cannot preclude bistability if a zero is not found. We address this uncertainty by computing the probability that the minimum objective function value achieved is equal to the true global minimum. In the future, we would like to couple the general technique with other methods that utilize the Gröbner basis for the ODE system. This coupling could produce further insight into the conditions of bistability for particular parameters produced by the optimization.

# CHAPTER 3

# MONTE CARLO SIMULATION

Since its introduction in the 1940s to aid in the development of nuclear weapons, Monte Carlo Simulation has proven to be an extremely useful tool in simulating stochastic processes and providing accurate metrics of the stochastic process. Due to the advancement of High Performance Computing (HPC) systems, Monte Carlo simulation has become a very practical method and is widely used in the scientific community. Although this is the case, many complex models that require a stochastic aspect to provide accurate representations of real world scenarios are far too time consuming to compute, even with the implementation of Monte Carlo simulation in parallel. In this chapter, we will present key ideas that lead to the improvement of Monte Carlo simulation approximations of the moment of a quantity of interest (QoI).

We begin by first introducing standard Monte Carlo (MC) simulation, the ideas of which were drawn from the extensive amount of books and material on the method [13, 23–25]. After considering MC, we establish key ideas behind Quasi-Monte Carlo (QMC) simulation, a method that aids in the improvement of the convergence rate of MC simulation using quasi-random points. Using MC and QMC, we demonstrate the behavior of these methods on simple random functions and a stochastic heat equation. Although MC simulation can be improved upon if QMC is used, both of these methods can still consume a large amount of HPC resources. To reduce the computational complexity of MC and QMC, we will conclude this chapter by exploring Multilevel Monte Carlo (MLMC) simulation and Multilevel Quasi-Monte Carlo (MLQMC) simulation, respectively.

## 3.1 Standard Monte Carlo

The first method that will be described is standard Monte Carlo simulation. MC is the most basic Monte Carlo simulation and is the foundation to many other evolved Monte Carlo methods. To understand MC, let us consider some random function $Q(\omega)$, which is dependent on the random

parameters $\boldsymbol{\omega}$ drawn from a uniform distribution. Typically, one chooses $\boldsymbol{\omega}$ to be an element of $\boldsymbol{\Omega} \subseteq \mathbb{R}^d$ and $(\boldsymbol{\Omega}, \mathcal{F}, \mathbb{P})$ to be the associated probability space with Euclidian sample space $\boldsymbol{\Omega}$ and Borel $\sigma$-algebra $\mathcal{F}$. The probabilities of the events in $\mathcal{F}$ are then given by the probability measure $\mathbb{P}$. Although we exclusively choose this probability space throughout the thesis, it should be known that the probability space can change according to the problem being considered. If a different probability space is chosen, it is imperative that all analysis and formulation created in this chapter be revisited.

Now that we have this stochastic function, we would like to know what the average behavior of our function is over our probability space. We find the average behavior of this function by taking the integral of $Q(\boldsymbol{\omega})$ over the given probability space. This is done by computing the integral (3.1). At this point, it is worth noting that the integral in (3.1) can be very high dimensional and the integrand can also be non-smooth. If this is the case, this integral is very difficult to calculate and an exact solution most likely does not exist. Thankfully, we can approximate this integral using MC simulation, an extremely reliable technique that can provide adequate approximations of (3.1), even if the integral is high dimensional and the integrand does not behave nicely.

$$\int_{\Omega} Q(\boldsymbol{\omega})\, d\boldsymbol{\omega} \tag{3.1}$$

MC approximates this integral by finding the stochastic moment ($\mathbb{E}[Q]$) of the non-deterministic QoI as in (3.2). This approximation then becomes more accurate as the number of samples $N_{MC}$ increases. Although we now have an approximation to the integral we would like to find, it is important to realize that we do not know how accurate this approximation is w.r.t. the exact value. This knowledge is easily obtained by calculating the variance in the evaluated QoI values, $Q(\boldsymbol{\omega})$. In MC simulation, this variance value is given by (3.3). By calculating the moment and variance of our QoI, we obtain measures of the mean and spread of the QoI, which are crucial to understanding the behavior of the QoI.

$$\mathbb{E}_{\text{MC}}\left[Q; N_{MC}\right] = \frac{1}{N_{MC}} \sum_{j=1}^{N_{MC}} Q(\boldsymbol{\omega}_j) \approx \lim_{N_{MC}\to\infty} \frac{1}{N_{MC}} \sum_{j=1}^{N_{MC}} Q(\boldsymbol{\omega}_j) = \int_{\Omega} Q(\boldsymbol{\omega})\, d\boldsymbol{\omega} \tag{3.2}$$

$$\mathbb{V}_{\mathrm{MC}}[Q; N_{MC}] = S_E^2[Q; N_{MC}] = \frac{1}{N_{MC}(N_{MC}-1)} \sum_{j=1}^{N_{MC}} \left( Q(\boldsymbol{\omega}_j) - \mathbb{E}_{\mathrm{MC}}[Q; N_{MC}] \right)^2 \tag{3.3}$$

An important fact to realize about the computation of (3.2) is that the $\boldsymbol{\omega}_j$ for $j = 1, \ldots, N_{MC}$ are independent samples of a random vector $\boldsymbol{\omega}$. This means that the approximation given by MC in (3.2) can be computed in a naturally parallel way (a technique we apply throughout this thesis), providing even more reason to consider MC simulation. Although the parallelizability of MC makes it an extremely simple process to understand, the method itself still presents problems when the realization of the QoI is expensive to compute.

Along with being computationally expensive for nonstandard QoI values, MC also has the disadvantage of a slow $O(N_{MC}^{-1/2})$ convergence rate. Hence, depending on the variance of the QoI, a large number of samples are required to obtain even a few matching digits of accuracy to the approximation of $\mathbb{E}[Q]$. One way to combat this slow convergence rate is by using deterministic points, as presented in Quasi-Monte Carlo simulation.

## 3.2 Quasi-Monte Carlo

Quasi-Monte Carlo (QMC) simulation is a method introduced in the 1950s which uses deterministic points in a high-dimensional space, generated by a low discrepancy sequence, in place of the random vector $\boldsymbol{\omega}$, to increase the convergence rate of MC simulation. This is done by making the deterministic points cover the interval of interest more evenly than random points, which leads to increased convergence in the MC approximation to the true value. To understand why this can improve the convergence rate, we consider Figure 3.1, which provides two sets "A" and "B" of data points between zero and one. The set "A" of values are generated from a uniform distribution using standard generators that are common across all coding languages. From this set "A", it is apparent that some of these random numbers can be very close to, if not overlap with, other random numbers. When these random values are close to each other, the integral in (3.1) is evaluated at points that are very similar. The evaluation of this integral for these close random values provides only a small amount of information about the behavior of the integrand that we

did not previously know, and for this reason, our approximation yields slow convergence. To avoid evaluating similar random numbers, we consider deterministic points that more appropriately cover the interval, as provided by the set "B" in Figure 3.1. By doing this, we avoid sampling the same or similar points, which provides more information about how the integrand is behaving.

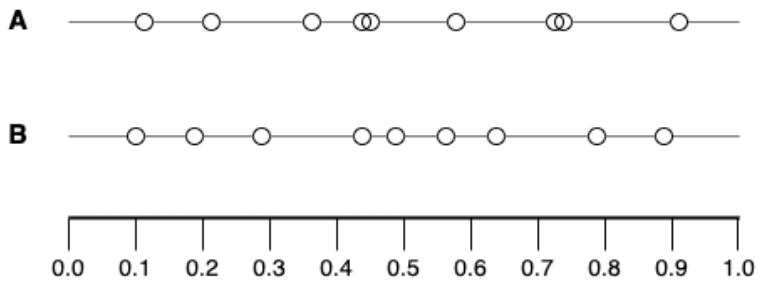



Figure 3.1: Visualizing a random draw of uniform random numbers "A" and deterministic points "B" between zero and one.

Although there are many ways to create these deterministic points, they often come from two main methods: lattice rules and digital nets. As stated in [26], if constructed correctly, lattice rules require adjustments based on the properties of the model being solved. If lattice rules were used, they would restrict the general applicability we are trying to accomplish when constructing our methods, which can lead to the efficient simulation of any stochastic model. For this reason, in this thesis we consider only digital nets, in particular, Sobol' sequences which satisfy a digital net in base two. Sobol' sequences were chosen because they are known to be a universal tool that can be applied to any model problem. However, the use of lattice rules can easily replace these Sobol points used, and if interested, the reader should refer to the following literature on the subject [27, 28].

### 3.2.1 Digital Nets and Shifting

Given we are mostly concerned with using digital nets to construct our data, we will now explain what digital nets are as well as describe the particular digital net that we will use throughout this thesis. The explanation of digital nets is a conglomerate of the information presented in the

detailed work of the following literature [26, 29–34]. A digital net, often referred to as a $(t, m, d)$-net in base $b$ is as defined in Definition 3.7. More generally, the main idea of digital nets is putting the same number of points in each allowable sub-division of the unit cube, which allows points that fit the interval more appropriately than standard number generators.

**Definition 3.7.** Let $d \geq 1, b \geq 2$, and $0 \leq t \leq m$ be integers. A point set $P$ consisting of $b^m$ points in $[0, 1)^d$ forms a $(t, m, d)$-net in base $b$ if every subinterval $\Pi_{j=1}^{d} = [\alpha_j b^{-\beta_j}, (\alpha_j + 1)b^{-\beta_j}] \in [0, 1)^d$ of volume $b^{t-m}$, with integers $\beta_j \geq 0$ and $0 \leq \alpha_j < b^{\beta_j}$ for $1 \leq j \leq d$, contains exactly $b^t$ points of $P$.

To provide a better understanding of digital nets, we will refer to Figure 4 of [29]. From this figure, we can see that we are first given $b^m = 2^4 = 16$ deterministic points, as found in the first box in the upper left corner of the figure. Given these points, we now want to partition the unit square in the figure into rectangles, while maintaining the same size and shape for each of the rectangles as stated in Definition 3.7. From the figure, we can see that the unit cube for these particular points can be split into 5 distinct allowable sub-divisions, while maintaining the property that each rectangle is the same size (as depicted by the faint dashed lines).
From these 5 allowable sub-divisions, we can see that for each sub-division there is exactly one point in each rectangle, if we assume that points on the top and right boundaries count toward the next rectangle. As one can see, if we allow the number of points to grow, while maintaining that the points be of base 2, then we obtain finer sub-divisions of the unit cube. This act of making the sub-divisions finer causes the points themselves to be spread across the unit cube in a more concise way. By making the points conform to this rule, we can increase the convergence rate of the MC simulation as stated in Section 3.2.
Even though the general idea of digital nets is fairly simple, if one tries to implement a digital net, it is apparent that some information is missing. Such as, what deterministic points can we use such that the points satisfy a digital net definition? Also, what value should $t$ be in Definition 3.7 to create sufficiently uniform data? These questions can be answered by considering Sobol' sequences. Note that a majority of the specifics about the construction of Sobol' sequences were

obtained from [92], which is also the methodology that [19] uses in the efficient construction of the Sobol' sequences we use throughout this thesis.

When Sobol was first considering quadrature methods, he wanted to compute (3.2) by replacing the random vectors $\omega_j$ with more uniform points, such that the limit would converge as fast as possible. Sobol was successful with this venture and ultimately published his work in [93], calling the sequences that created a deterministic $\omega_j$ that satisfied the fast convergence, $LP_\tau$ sequences. These $LP_\tau$ sequences ultimately became known as Sobol' sequences. We now provide a general overview of the construction of these Sobol' sequences.

Let $x_i$ for $i = 1, \ldots, N$ be a Sobol' sequence. We can then construct the $j$th component of the $i$th point in the Sobol' sequence as follows

$$x_{i,j} := i_1 v_{1,j} \oplus i_2 v_{2,j} \oplus \cdots \oplus i_{dgts} v_{dgts,j}, \tag{3.4}$$

where $i_k$ is the $k$th binary digit of $i = (i_{dgts} \ldots i_3 i_2 i_1)_2$, $i_{dgts}$ is the digit precision of the computer architecture being used, $\oplus$ is a bit-by-bit exclusive-or operator, and $(\cdot)_2$ denotes the binary representation of numbers. The difficulty in constructing these points of the Sobol' sequence comes in when one wants to construct the so called direction numbers $\{v_{1,j}, v_{2,j}, \ldots, v_{dgts,j}\}$. Thanks to Sobol, we know that these direction numbers can be computed by constructing a primitive polynomial of degree $s_j$ over the field $\mathbb{Z}_2$,

$$x^{s_j} + a_{1,j} x^{s_j - 1} + a_{2,j} x^{s_j - 2} + \cdots + a_{s_j - 1, j} x + 1, \tag{3.5}$$

where $a_{1,j}, a_{2,j}, \ldots, a_{s_j-1,j}$ are either 0 or 1. For a more detailed explanation of primitive polynomials, see [94]. Using these coefficient values, $a_{1,j}, a_{2,j}, \ldots, a_{s_j-1,j}$, we can then construct a sequence of positive integers as in (3.6) that will allow us to construct the direction numbers.

$$m_{k,j} := 2a_{1,j} m_{k-1,j} \oplus 2^2 a_{2,j} m_{k-2,j} \oplus \cdots \oplus 2^s_{j-1} a_{s_j-1,j} m_{k-s_j+1,j} \oplus 2^{s_j} m_{k-s_j,j} \oplus m_{k-s_j,j} \tag{3.6}$$

Using the details in (3.5) and (3.6), we can then obtain the final direction numbers $\{v_{1,j}, v_{2,j}, \ldots, v_{dgts,j}\}$ defined by

$$v_{k,j} := \frac{m_{k,j}}{2^k}, \text{ for } 1 \le k \le s_j. \tag{3.7}$$

It should be noted here that obtaining theoretically concrete and trusted values for the sequence in (3.6) for high dimensional Sobol' points is difficult, and for this reason, an efficient method created in [92] is used that can obtain up to $d = 21201$ dimensional Sobol' points. This unprecedented amount of dimensions is accomplished with careful construction of the $m_{k,j}$ values and primitive polynomials.

Although we have established the deterministic points we will use, we have yet to see if these points satisfy a digital net and if they provide any insight into the value of $t$ in Definition 3.7. As it turns out, a Sobol' sequence in $d$ dimensions is a $(t, d)$-sequence in base $b = 2$ as defined in Definition 3.8, which forms a $(t, m, d)$-net in base 2. Along with this property, Sobol' sequences also provide a verified way of computing $t$ in Definition 3.7. This $t$ value is as given by (3.8), and is thus dependent upon the degree of the primitive polynomial constructed.

$$t = \sum_{j=1}^{d} (s_j - 1) \tag{3.8}$$

Following the guidelines above and the details in [92], we arrive at a sequence of numbers that are optimally distributed, a fact provided by the digital net definition and appropriate value of $t$ given in [92], as well as an efficient computation of them provided by [19].

**Definition 3.8.** Let $d \geq 1, b \geq 2$, and $t \geq 0$ be integers. A sequence $\{x_i : i \geq 0\}$ of points in $[0, 1)^d$ is a $(t, d)$-sequence in base $b$ if every block of $b^m$ points, $x_{\ell b^m}, \ldots, x_{(\ell+1)b^m-1}$ for $\ell \geq 0$ and $m \geq t$, forms a $(t, m, d)$-net in base $b$.

Although these Sobol' sequences by themselves can improve the convergence rate of MC, due to the efficient computation for high dimensions provided by [92] it is apparent that these points will be the same for a fixed dimension and thus do not have a random aspect to them. For this reason, it is clear that if we used the same methodology for finding the convergence rate as we did in MC, then we would obtain a biased convergence study. To eliminate this biased estimate and maintain our comparison with MC, we will explore digital nets with shifting.

Digital nets with shifting takes the deterministic points, say $\boldsymbol{\omega}_k$ and shifts them by a random shift parameter, say $\Delta$. If we let $\boldsymbol{\omega}_k$ be the $k^{th}$ point generated by a digital net rule, a digital shifting of

the $k^{th}$ point is then given by $\omega_k \oplus \Delta$. Here $\oplus$ maps the points back into the unit cube as dipicted in Figure 1 of [29]. So for example, if we let $x = 0.625 = (0.101)_2$ and $y = 0.125 = (0.001)_2$, then $x \oplus y = (0.100)_2 = 0.5$.

### 3.2.2 Higher Order Digital Nets

The process followed by digital net rules and shifting can provide even better convergence rates if interlacing is applied to the digital nets. These higher order digital nets can obtain convergence rates of $O(N^{-\alpha})$, where $N$ is the number of samples, if the integrand being used is $\alpha$ times differentiable in each stochastic variable. Here, $\alpha$ is often referred to as the interlacing factor. Although we implement higher order digital nets, the theory behind why these higher order methods work is beyond the scope of this thesis. However, full details and explanation are available in [32, 95, 96]. Essentially, by using interlacing one can create a $(t, m, \alpha d)$-net in base 2, this increases the number of subintervals created (as can be seen from Definition 3.7) and thus makes the deterministic points cover the interval of interest in a better way. Although integrands that are $\alpha$ times differentiable in each stochastic variable obtain this high convergence rate, it is also possible for integrands that are not $\alpha$ times differentiable in each variable to achieve $O(N^{-\alpha})$, as we will demonstrate in this Chapter, Chapter 4, and Chapter 5.

To obtain a better understanding of interlacing, we will consider a simple example. If we let $x = (0.x_1x_2x_3\dots)_2, y = (0.y_1y_2y_3\dots)_2$, and $z = (0.z_1z_2z_3\dots)_2$, then the result of interlacing these three numbers with an interlacing factor of $\alpha = 3$ is the value $(0.x_1y_1z_1x_2y_2z_2x_3y_3z_3\dots)_2$. Note that this new value has three times as many bits as the original values used. This increase in the number of bits presents one concern and drawback of higher order digital nets. That is, one must use coding languages and HPC systems that allow variables to obtain high precision. For example, if one wanted to obtain $N = 2^m$ deterministic points with an interlacing factor of $\alpha$, then the precision of the machine, $p_m$, must satisfy $\alpha m \leq p_m$. This restriction places a burden on interlacing factors greater than or equal to 2.

### 3.2.3 Implementing QMC

Now that we have an understanding of the quasi-random points we will be using, we will continue by explaining the implementation of QMC. The QMC method yields an approximation to the expected QoI as

$$\mathbb{E}_{\text{QMC}}\left[Q; N_{QMC}\right] = \frac{1}{N_{QMC}} \sum_{r=1}^{N_s} \sum_{i=1}^{nn} Q(\boldsymbol{\omega}_{r,i}), \tag{3.9}$$

where the total number of realizations $N_{QMC} = N_s * nn$ and $N_s$ is the number of random shifts to be completed. To obtain the deterministic points $\boldsymbol{\omega}_{r,i}$, we will construct Sobol' sequences which satisfy a $(t, m, d)$-net in base 2, with particular interlacing factors. The construction of these points is made possible by the optimal code produced by Frances Kuo and Dirk Nuyens in [19]. The QMC method also yields an approximation to the standard error of the QoI

$$S_{E,\text{QMC}}\left[Q; N_{QMC}\right] = \sqrt{\frac{1}{N_s(N_s-1)} \sum_{r=1}^{N_s} \left( \frac{1}{nn} \left[ \sum_{i=1}^{nn} Q(\boldsymbol{\omega}_{r,i}) \right] - \mathbb{E}_{\text{QMC}}\left[Q; N_{QMC}\right] \right)^2}. \tag{3.10}$$

By using this construction, we can achieve optimal convergence rates for a particular problem. Due to the independence of the $\boldsymbol{\omega}_{r,i}$ vectors, we can parallelize the computation of (3.9). To achieve a large number of QMC based realizations we employ a message passing interface (MPI) based parallel code written in modern versions of Fortran (90+), which parallelizes the process with respect to $nn$ and $N_s$ if the number of cores used is greater than $N_s$, and parallelizes the process with respect to $N_s$ if the number of cores is less than or equal to $N_s$. As in MC, we run the parallel code using multiple cluster nodes with each node composed of dual octa-core Intel X5670 2.93GHz processors.

## 3.3 QMC Applied to Random Functions

Now that we have a concrete understanding of the construction of the QMC method and the points we will be using, let us now continue by considering QMC applied to random functions. The purpose of this section is to establish the properties of QMC with small and succinct examples. The first example we will consider is finding the mean and variance for the integral in

(3.11), using different interlacing factors valued from 1-4. In this particular example, we see that $Q(\boldsymbol{\omega}) = \boldsymbol{\omega}$, where $\boldsymbol{\omega}$ is a single variable.

$$\mathbb{E}[Q] = \int_0^1 \boldsymbol{\omega}\, d\boldsymbol{\omega} \tag{3.11}$$

Given we are trying to compute a numerical integral between zero and one, we see that the vector $\boldsymbol{\omega}$ will consist of random values between zero and one. If we were instead integrating between two and four, then we would let our vector $\boldsymbol{\omega}$ be random values between two and four. From (3.11), we can see that the integrand is infinitely differentiable. This means that according to theory, this QoI should perform very well for all interlacing factors and should also provide a convergence rate of $\mathcal{O}(N^{-\alpha})$. We also note that the exact solution to (3.11) is know to be 0.5. This allows us to construct the absolute error in our estimate, denoted abs, by taking the absolute value of the difference between our approximation to $E[Q]$ and the exact solution 0.5.

To consider the convergence rate of the approximation to (3.11), we will first run QMC with two random shifts, in addition to MC. The results of this can be found in Figure 3.2. In the left graph of Figure 3.2, we see that we have plotted the standard error, $S_E[Q]$ as defined in (3.10), for MC and QMC. Along with this, we have also plotted the number of samples, $N$, vs. $N^{-\alpha}$ (as depicted by the green lines). From this left graph, we can see that two shifts produce convergence rates that are consistent with the theory, allowing us to obtain fast convergence of $\mathcal{O}(N^{-\alpha})$.

Since the exact solution of the integral is often not known, in practice one must resort to using $S_{E,QMC}[Q; QMC]$. Thus, to ensure that this is a fair estimate of the error, we have compared $S_E[Q]$ and the absolute error of $Q$ in the right plot of Figure 3.2. From the right plot, we can see that $S_E[Q]$ provides a fair representation of the accuracy of QMC for different interlacing factors. From these plots, we have found that the theory for interlacing holds and we have increased the convergence rate of MC. However, what happens to these results when the number of shifts change?

To answer this question, we run this process again, but this time we will use eight shifts. The results for this can be found in Figure 3.3. If we refer to the left plot of Figure 3.3, we see that

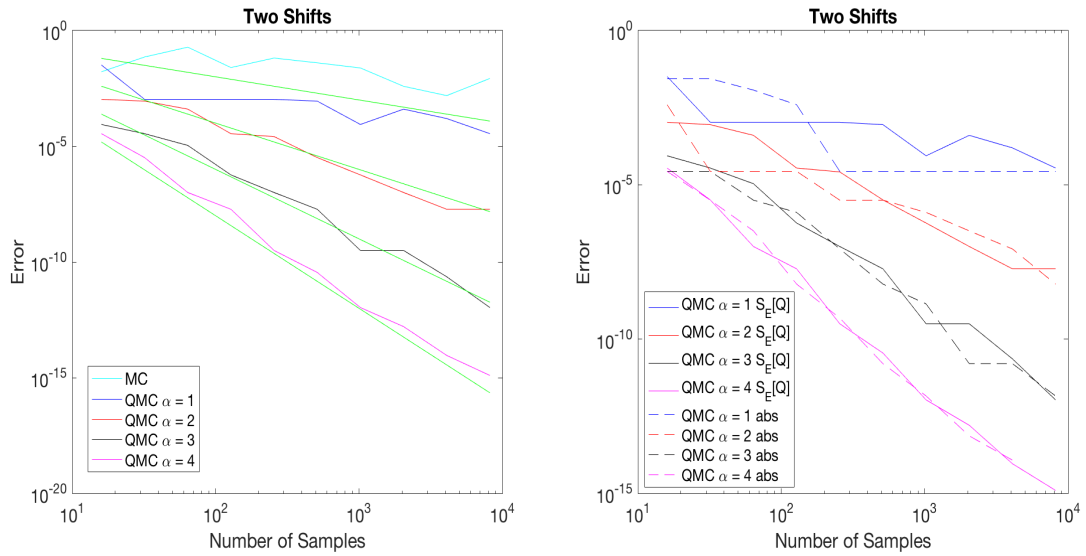



Figure 3.2: Comparing MC with QMC using $S_E[Q]$ (left) and comparing $S_E[Q]$ and absolute error for the QoI, using two shifts (right).

eight shifts makes the $S_E[Q]$ worse. However, if we consider the right plot, we see that our $S_E[Q]$ is a much better approximation of the absolute error. These results show that varying the number of shifts can dictate the behaviour of the approximation substantially and for the purposes of implementation, one should consider more shifts, rather than a small amount of shifts.

It is apparent that the number of shifts along with the interlacing factor used dictates the behaviour of QMC, but how does the smoothness of the integrand affect the results? To investigate the affect of the smoothness of the integrand, we will first consider the QoI in (3.12). Note that this integrand is not infinitely differentiable, instead it only has one derivative that is defined on the domain of integration. Thus, according to the QMC theory, only an interlacing factor of $\alpha = 1$ should produce the expected convergence rate, while the other interlacing factors are not guaranteed to produce the expected convergence rate. To see if this is the case, we run this method for the integral in (3.12), producing the output in Figure 3.4. Here, the absolute error is found with the exact solution of 2/3.

$$\int_0^1 \omega^{1/2}\, d\omega \tag{3.12}$$

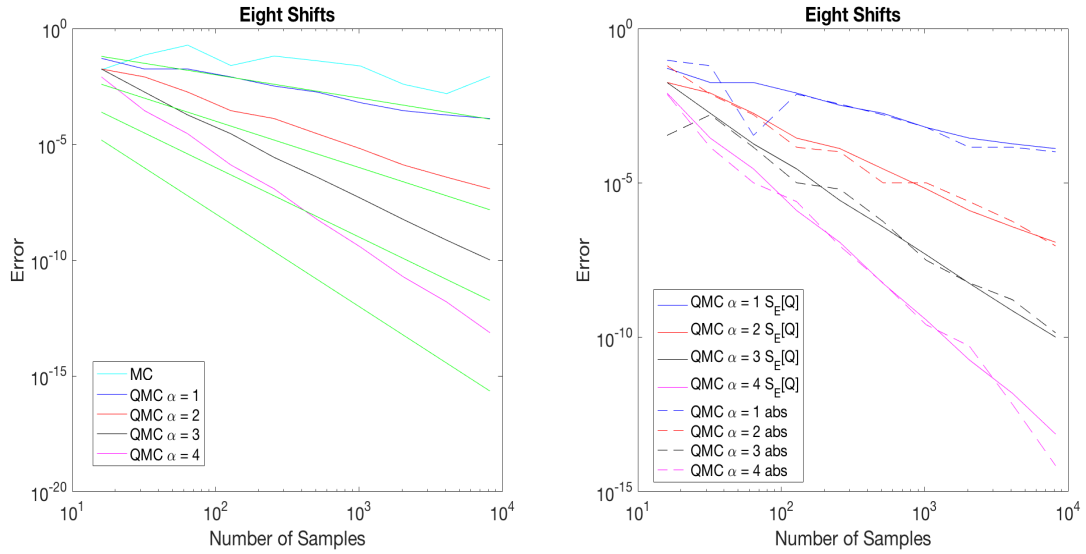



Figure 3.3: Comparing MC with QMC using $S_E[Q]$ (left) and comparing $S_E[Q]$ and absolute error for the QoI, using eight shifts (right).

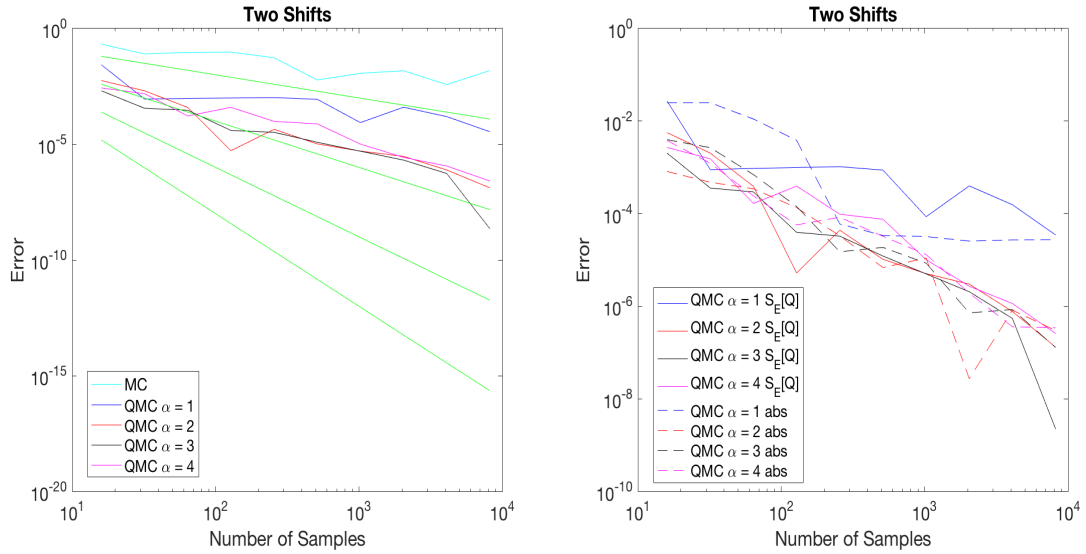



Figure 3.4: Comparing MC with QMC using $S_E[Q]$ (left) and comparing $S_E[Q]$ and absolute error for the QoI, using two shifts (right).

From the left plot of Figure 3.4, we can see that $\alpha = 1$ behaves as expected, as well as the interlacing factors 3 and 4. However, we see that $\alpha = 2$ still produces the expected convergence rate, as dictated by theory. Thus, it seems like QMC with interlacing and shifting can achieve the expected convergence rate $O(N^{-\alpha})$, even if the integrand is not sufficiently smooth. If we consider the right plot, we see that $S_E[Q]$ is still a sufficient metric to study the error of the approximation. To see if other integrands act in a similar manner to $\omega^{1/2}$, we will now consider the integrand given in (3.13).

$$Q(\omega) = \int_0^1 \omega^{3/2}\, d\omega \tag{3.13}$$

The integrand of (3.13) can be differentiated twice and has an exact value of 2/5. Since the integrand can be differentiated twice, it follows that $\alpha = 1$ and $\alpha = 2$, should produce the appropriate convergence rate, as stated by the theory, while $\alpha = 3$ and $\alpha = 4$ fail to produce the expected convergence rate. To see if this is the case, we run the method and obtain the results in Figure 3.5. These results confirm that both interlacing factors of 1 and 2 are behaving appropriately. We also see that $\alpha = 3$ and $\alpha = 4$ do not obtain convergence of order $O(N^{-3})$ and $O(N^{-4})$, respectively. This is expected behavior because the integrands are not sufficiently smooth, as dictated by the theory. Thus, it seems like the integrand can produce higher order convergence even if it is not sufficiently smooth, but this behavior is not guaranteed. Although the higher interlacing factors don't obtain the expected convergence rate, it is shown that higher interlacing factors are still beneficial in some cases, even if the integrand is not sufficiently smooth.

### 3.3.1 QMC Applied to Stochastic PDEs

From the last section, we found that the interlacing factor, the amount of shifts, and the smoothness of the integrand greatly affect the performance of QMC. Given we will ultimately apply these techniques to PDEs with stochastic aspects, it seems appropriate to show how QMC will affect a simple PDE of this form. To show the affect of QMC on PDEs, we will consider random heat flow in a thin square plate. For this simulation, we will be using the implicit second-order in space-and-time Crank-Nicolson linear Galerkin spline finite element method

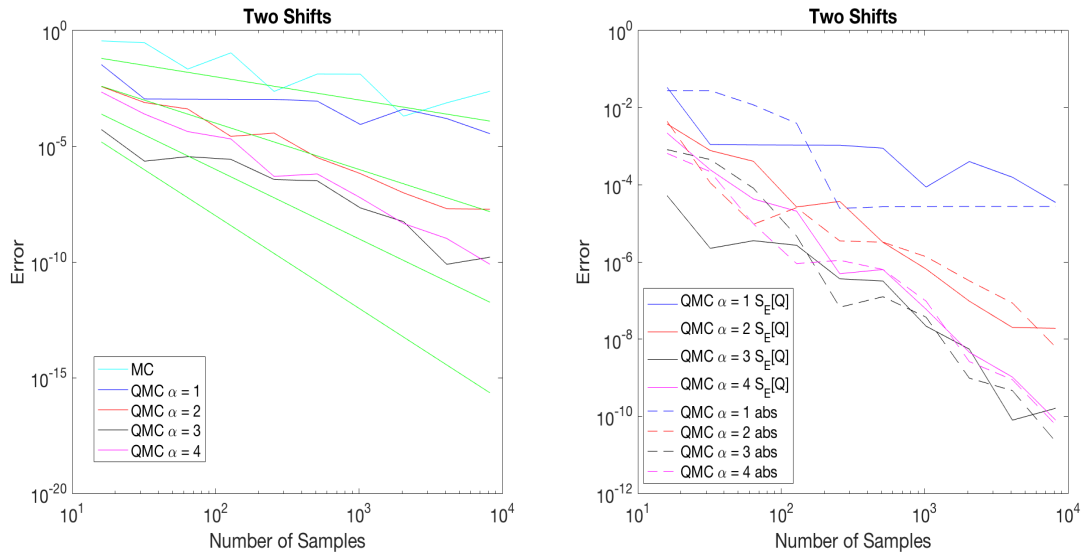



Figure 3.5: Comparing MC with QMC using $S_E[Q]$ (left) and comparing $S_E[Q]$ and absolute error for the QoI, using two shifts (right).

(FEM), for the numerical solve of the proposed PDE. The domain for the random heat flow in a thin square plate will be denoted by $D$, where $D$ is located in the $x-y$ plane occupying the region $\bar{D} = \{(x,y) \in \mathbb{R}^2 : 0 \le x \le 1,\ 0 \le y \le 1\}$.

For this problem, we again assume that $\boldsymbol{\omega}$ is an element of $\boldsymbol{\Omega} \subseteq \mathbb{R}^d$, and $(\boldsymbol{\Omega}, \mathcal{F}, \mathbb{P})$ is the associated probability space with Euclidian sample space $\boldsymbol{\Omega}$ and Borel $\sigma$-algebra $\mathcal{F}$. The probabilities of the events in $\mathcal{F}$ are given by the probability measure $\mathbb{P}$. For notational convenience, we use $\widetilde{\boldsymbol{\omega}} = (\boldsymbol{\omega}, \omega)$ to denote a random vector in $\widetilde{\boldsymbol{\Omega}} = \boldsymbol{\Omega} \times \Omega \subset \mathbb{R}^{d+1}$. The associated product probability measure in $\widetilde{\boldsymbol{\Omega}}$ is $\widetilde{\mathbb{P}} = \mathbb{P} \times \mathcal{P}$.

We will then force the temperature $u(x,y,t;\tilde{\omega})$ at each position $(x,y) \in D$ and at each time $t$ to satisfy the following,

$$\frac{\partial u}{\partial t}(x,y,t;\tilde{\omega}) - \omega \Delta u(x,y,t;\tilde{\omega}) = f(x,y,t;\tilde{\omega}), \qquad \text{in } D \quad t > 0,$$

with boundary condition

$$u(x,y,t;\widetilde{\boldsymbol{\omega}}) = 0 \ \ (x,y) \in \partial D,$$

and for $0 \leq t \leq T$ the initial temperature distribution

$$u(x, y, 0; \widetilde{\omega}) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_e} \frac{\gamma_i(\omega)\eta_j(\omega)}{\sqrt{i^2 + j^2}} \sin(i\pi x) \sin(j\pi y) \quad (x, y) \in D,$$

where the $\eta_j(\omega)$ and the $\gamma_i(\omega)$ are random variables between 0 and 1, $\omega$ is a random variable between 0 and 0.5, and $f(x, y, t; \tilde{\omega})$ is a random forcing function, which is constructed from the exact solution below

$$u(x, y, t; \widetilde{\omega}) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_e} \frac{\gamma_i(\omega)\eta_j(\omega)}{\sqrt{i^2 + j^2}} sin(i\pi x) sin(j\pi y) Exp\Big(\frac{-ijt}{N_g N_e}\Big).$$

Now that we have our PDE, we would like to quantify its randomness using MC and QMC. For this particular problem, we will let our QoI be as given in (3.14), and thus, we would like to approximate the integral in (3.15). At first, we see that there is one major difference between the PDE QoI and the random functions of the last section, that is, our PDE solution is obtained through a discretization of the PDE, rather than given by an explicit function. Thus, the question becomes, what solution of the PDE should we consider? To answer this question, we must refer to the convergence study. Using the exact solution, we find the absolute maximum nodal error over all time steps ($Err^{max}$) of the problem and obtain the convergence study given in (Table 3.1), where $h$ specifies the spatial discretization, $\Delta t = 0.25h$ specifies the temporal discretization, $T = 1.0$, and $N_g = N_e = 3$.

$$Q(\widetilde{\omega}) = \int_D u(x, y, T; \widetilde{\omega})\ dx\ dy, \ \ \widetilde{\omega} \in \widetilde{\mathbf{\Omega}}. \tag{3.14}$$

$$\mathbb{E}[Q] = \int_{\widetilde{\Omega}} Q(\widetilde{\omega})\ d\widetilde{\mathbb{P}} \tag{3.15}$$

From Table 3.1, we see that our method is converging correctly and our error is reducing with finer mesh realizations. From this table, we also see that a fine $h$ step of 7.8125E-3 seems like a solution that mimics the exact solution best, given it produces the best absolute maximum nodal error. For this reason, we use it to approximate the QoI. Using MC and QMC, we approximate (3.15) using (3.16) and (3.17), respectively, where the subscripts denote that a fine grid realization

Table 3.1: Convergence of the random heat equation.

| $h$ | $Err^{max}$ | EOC of $Err^{max}$ |
|---|---|---|
| 0.25 | 5.0566E-3 | |
| 0.125 | 1.2304E-3 | 1.9062 |
| 6.25E-2 | 3.0255E-4 | 1.9488 |
| 3.125E-2 | 7.5290E-5 | 1.9640 |
| 1.5625E-2 | 1.8800E-5 | 1.9790 |
| 7.8125E-3 | 4.7074E-6 | 1.9887 |

was used for both space and time. Also note the slight change in notation for the expectation of QMC, the notation $\mathbb{E}_{QMC,shft}[Q]$ indicates that the expectation produced by QMC was created using $shft$ amounts of shifts. We also obtain a similar notation for $S_{E,shft}[Q]$. For this particular example, we let $h^{fine} = 7.8125E-3$ and $\Delta t^{fine} = 0.25h^{fine}$, as determined by the convergence study.

$$\mathbb{E}_{\text{MC}}[Q; N_{MC}] = \frac{1}{N_{MC}} \sum_{j=1}^{N_{MC}} Q_{\Delta t^{fine}, h^{fine}}(\widetilde{\omega}_j), \tag{3.16}$$

$$\mathbb{E}_{QMC,shft}[Q; N_{QMC}] = \frac{1}{N_{QMC}} \sum_{r=1}^{M\_iter} acc_{\Delta t^{fine}, h^{fine}}(r) \tag{3.17}$$

Using the construction outlined above, we will now run MC and QMC for $\alpha = 1$ and shift amounts of 8,16, and 32. To obtain the absolute error of this problem we used the exact solution provided in (3.18). The data produced by these runs can be seen in Table 3.2. From this table, we can see that shift amounts of 8 and 32 produce $S_E[Q]$ values that can be an order of magnitude off from the absolute error. This fact becomes more apparent when we refer to Table 3.3. This table shows us that the standard error, $S_E[Q]$, for 16 shifts most closely approximates the absolute error. Thus, when considering this model, 16 shifts should be used for all QMC methods. Although this behavior is true for this PDE, it should be noted that this is not indicative of all PDEs with stochastic aspects. For more complex models, it is often better to use 32 shifts as stated in the following literature [29, 97], thus, we fix 32 shifts for the models considered in Chapters4 and 5.

$$\int_{\mathbf{\Omega}} \left( \int_D u(x, y, t; \widetilde{\omega}) dy dx \right) \, d\gamma_1 d\gamma_2 d\gamma_3 d\eta_1 d\eta_2 d\eta_3 \tag{3.18}$$

$$= \int_{\Omega} \left( \int_{D} \sum_{i=1}^{3} \sum_{j=1}^{3} \frac{\sin(\pi i x) \sin(\pi j y) \exp\left(-\frac{ijt}{9}\right) \gamma_i(\omega) \eta_j(\omega)}{\sqrt{i^2 + j^2}} dy dx \right) d\gamma_1 d\gamma_2 d\gamma_3 d\eta_1 d\eta_2 d\eta_3$$

$$= \frac{e^{-t}\left(18\sqrt{5}e^{\frac{2t}{3}} + 135e^{\frac{8t}{9}} + 5\right)}{135\sqrt{2}\pi^2}$$

Table 3.2: Error in QMC for $\alpha = 1$ and various shift amounts.

| N | 512 | 1024 | 2048 | 4096 | 8192 |
|---|---|---|---|---|---|
| $S_{E,8}[Q]$ $\alpha = 1$ | 4.2702e-4 | 1.8997e-4 | 5.08989e-5 | 2.24697e-5 | 2.93217e-5 |
| $abs_8[Q]$ $\alpha = 1$ | 8.6412e-5 | 6.5569e-5 | 7.7472e-5 | 1.5069e-5 | 2.1796e-6 |
| $S_{E,16}[Q]$ $\alpha = 1$ | 5.5541e-4 | 2.7023e-4 | 1.1698e-4 | 4.7914e-5 | 2.0099e-5 |
| $abs_{16}[Q]$ $\alpha = 1$ | 6.3017e-4 | 1.2458e-5 | 5.5532e-5 | 5.6203e-5 | 2.0919e-5 |
| $S_{E,32}[Q]$ $\alpha = 1$ | 7.2370e-4 | 3.9026e-4 | 1.8422e-4 | 8.8139e-5 | 3.6442e-5 |
| $abs_{32}[Q]$ $\alpha = 1$ | 1.1489e-4 | 8.7968e-5 | 1.1142e-4 | 5.9017e-6 | 2.1349e-5 |

Table 3.3: Difference in absolute and standard error produced by QMC for $\alpha = 1$ and various shift amounts.

| | |
|---|---|
| Max($abs_8[Q]$ - $S_{E,8}[Q]$) $\alpha = 1$ | 3.406068491984040e-04 |
| Max($abs_{16}[Q]$ - $S_{E,16}[Q]$) $\alpha = 1$ | 2.577766689167132e-04 |
| Max($abs_{32}[Q]$ - $S_{E,32}[Q]$) $\alpha = 1$ | 6.088142793326909e-04 |

### 3.4 Multilevel Monte Carlo

From the implementation of MC and QMC, we see that applications where the cost per sample is low present little difficulty for both applications. However, for complex PDEs this is often not the case because each sample requires a prolonged PDE solve. In this section, we will show how multilevel Monte Carlo (MLMC) can reduce the computational complexity and runtime of Monte Carlo simulation. The MLMC algorithm was introduced less than two decades ago and we follow the framework in the survey article [17] to further develop our Monte Carlo code.

Under certain assumptions [17, Theorem 1], it is possible to choose desired accuracy values $\epsilon$ (such as $\epsilon = [N_{MC}]^{-1/2}$ in our MC simulations). This can be achieved by performing a few more,

say, $L$-levels of MC simulations, with integrands having smaller variances, as we increase from level 0 to level $L$. The reduction in the computational cost can be achieved using certain trade-offs between finer and coarser realizations of the numerical grids.

In particular, at level $\ell = 0$ the MLMC algorithm is based on simulations using the largest number of adaptively chosen $N^{(\ell)}_{\text{MLMC}}$ samples and they obtain an initial approximation similar to the representation in (3.16), but $\Delta t^{fine}$ and $h^{fine}$ replaced by coarse level mesh parameters $\Delta t^{(\ell)}$ and $h^{(\ell)}$. If we consider the random heat equation for $\ell = 0, \ldots, L$, using (3.14), we define the $\ell$-th level QoI as

$$Q^{(\ell)}_{\Delta t^{(\ell)}, h^{(\ell)}}(\widetilde{\omega}) = \int_D u(x, y, t; \widetilde{\omega})\, dx\, dy, \;\; \widetilde{\omega} \in \widetilde{\mathbf{\Omega}}. \tag{3.19}$$

Hence corresponding to (3.15), we obtain the telescopic sum relation

$$\mathbb{E}[Q^{(L)}] = \mathbb{E}[Q^{(0)}] + \sum_{\ell=1}^{L} \mathbb{E}[Q^{(\ell)} - Q^{(\ell-1)}]. \tag{3.20}$$

The key outcome of this identity is that while the variance of $Q^{(\ell)}$, i.e. $V^{(\ell)}$, for $\ell = 0, \ldots, L$ may not be small, the variance of the difference $Q^{(\ell)} - Q^{(\ell-1)}$ in (3.20) are expected to reduce in magnitude as the level increases. The expected reduction in variances follows from the control variate analysis [13], as explained in [17, Page 3]. Because of the substantially reduced order of the variance, the MC approximations for $\mathbb{E}[Q^{(\ell)} - Q^{(\ell-1)}]$ require substantially fewer $N^{(\ell)}_{\text{MLMC}}$ samples compared to the $N^{(0)}_{\text{MLMC}}$ samples used at the initial level.

Using the tabulated values in Table 3.1 as a reference, for MLMC simulations we use the coarse to finer space-time mesh for each realization of the deterministic computation of $Q^{(\ell)}_{\Delta t^{(\ell)}, h^{(\ell)}}$ using

$$h^{(\ell)} = 2^{(-\ell-3)} \text{ and } \Delta t^{(\ell)} = 2h^{(\ell)} \qquad \ell = 0, 1, 2, 3. \tag{3.21}$$

The initial $\ell = 0$ MLMC approximations $\mathbb{E}^{(\ell)}_{\text{MLMC}}\left[Q^{(0)}\right]$ to $\mathbb{E}\left[Q\right]$ in (3.15) were computed with $\ell = 0$ and $Q^{(0)} = \Phi$ in the following representation:

$$\mathbb{E}^{(\ell)}_{\text{MLMC}}\left[\Phi\right] := \mathbb{E}_{\text{MLMC}}\left[\Phi; N^{(\ell)}_{\text{MLMC}}; \Delta t^{(\ell)}; h^{(\ell)}\right] := \frac{1}{N^{(\ell)}_{\text{MLMC}}} \sum_{j=1}^{N^{(\ell)}_{\text{MLMC}}} \Phi_{\Delta t^{(\ell)}, h^{(\ell)}}(\widetilde{\omega}^{(\ell)}_j), \tag{3.22}$$

where $\widetilde{\omega}_j^{(\ell)} \in \widetilde{\mathbf{\Omega}}$ are independent samples that change with the levels. The identity (3.20) motivates that the final (level $\ell = L$) MLMC approximation $\mathbb{E}^{(L)}_{\text{MLMC}}\left[Q^{(L)}\right]$ to $\mathbb{E}[Q]$ in (3.15) is computed using the following representation

$$\mathbb{E}^{(L)}_{\text{MLMC}}\left[Q^{(L)}\right] = \mathbb{E}^{(0)}_{\text{MLMC}}\left[Q^{(0)}\right] + \sum_{\ell=1}^{L} \mathbb{E}^{(\ell)}_{\text{MLMC}}\left[Q^{(\ell)} - Q^{(\ell-1)}\right]. \tag{3.23}$$

Unlike the MC algorithm, the MLMC approach allows the user to provide an expected accuracy value $\epsilon$ and based on the coarse to fine grid space-time mesh (such as that in (3.21)) and a certain initial number of MC samples at each level (with prescribed minimum and maximum number of levels), the MLMC framework [17] adaptively chooses the final number of samples $N^{(\ell)}_{\text{MLMC}}$ to approximately satisfy the required accuracy. From [17, Theorem 1], we also see that the MLMC algorithm requires the three values $\alpha_{MLMC}, \beta$, and $\gamma$, where $\alpha_{MLMC}$ is not associated with the interlacing factor. Although $\alpha_{MLMC}$ and $\beta$ can be approximated within the MLMC algorithm in real time, we need to provide an estimate to $\gamma$. In practice, this value can be obtained from the code mlmc_test provided by [17]. In this code, the user provides a value $N_0$, which represents the number of samples to be ran on each level of MLMC. Along with this value, the user should also supply the total number of levels they want to run mlmc_test with, say $L_{test}$. Once these two values are provided, mlmc_test then conducts the MLMC algorithm, but with a fixed number of levels and samples per level. Using these runs, the algorithm then estimates the parameter $\gamma$ using the cost per level, as provided by theory. An algorithm overview of MLMC (formulated in [17]) is presented in Algorithm 3.

**Algorithm 3** Pseudocode for MLMC

start with $L = 2$, set $\epsilon_{MLMC}$, and initial target of samples on levels $\ell = 0, 1, 2$
**while** extra samples need to be evaluated **do**
    evaluate extra samples on each level
    compute/update estimates for $V^{(\ell)}$, $\ell = 0, \ldots, L$
    define optimal $N^{(\ell)}$, $\ell = 0, \ldots, L$
    test for weak convergence
    if not converged, set $L := L + 1$, and initialize target $N^{(\ell)}$
**end while**

Now that we have established the theory behind MLMC, let us continue by showing its clear advantage over MC, using the random heat equation introduced in Section 3.3.1. As stated above, the first step in running the MLMC algorithm is to establish the parameter $\gamma$. Using the space-time mesh of (3.21), the approximation (3.23), $N_0 = 10000$, and $L_{test} = 3$, we obtain $\gamma = 3.10$ from mlmc_test. Note that if the mesh provided by (3.21) is not chosen appropriately, then mlmc_test will also yield poor results for the predicted variance, $\gamma$, kurtosis value, and the check value. These values can also yield poor results if the code is constructed incorrectly, in most cases if the check value is not sufficient, then the switch from a fine mesh to a coarse mesh is not conducted appropriately.

The next step in the process of running MLMC is to establish the $\epsilon$ value that you want to achieve and the initial number of samples per level. We first consider the $\epsilon$ value by referring to Figure 3.6, which are the best results obtained by QMC and MC, when using the random heat equation. From this graph, one can see that MC has a convergence rate of $O(N^{-0.8})$, thus if we want to mimic this convergence rate with MLMC, we need to let $\epsilon$ be approximated with $\epsilon_{MLMC} = N^{-0.8}$. Next, we must set the initial number of samples per level. Although the theory provides a rough estimation for the initial number of samples, in practice it is usually easier if one first chooses a large $\epsilon_{MLMC}$ and through trial and error finds the minimum amount of samples to conduct on each level.

Using the process we have outlined so far, $\epsilon_{MLMC} = N^{-0.8}$, $\gamma = 3.10$, and the initial number of samples on levels 0,1, and 2 to be 8,4, and 2, respectively, then one obtains the results in Figure 3.7 and Figure 3.8. Note here that the $N$ in $\epsilon_{MLMC} = N^{-0.8}$ is the number of samples used in MC. From these results, it is apparent that MLMC can produce the same standard error as MC, but does so in a way that reduces computational cost significantly. In the following chapters, we will consider how MLMC can provide even greater improvements for models that require longer runtimes per sample.

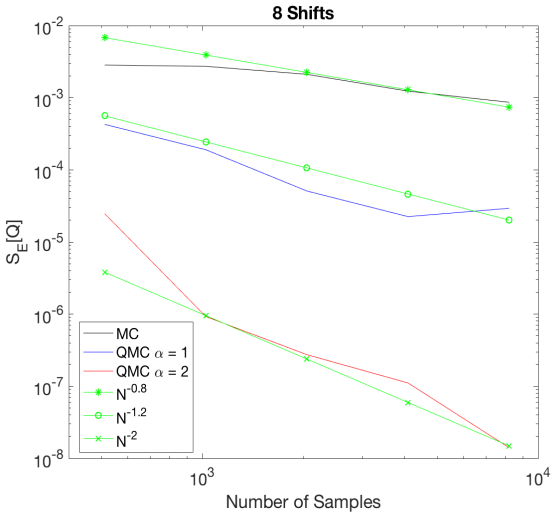



Figure 3.6: Rates of convergence for MC and QMC.

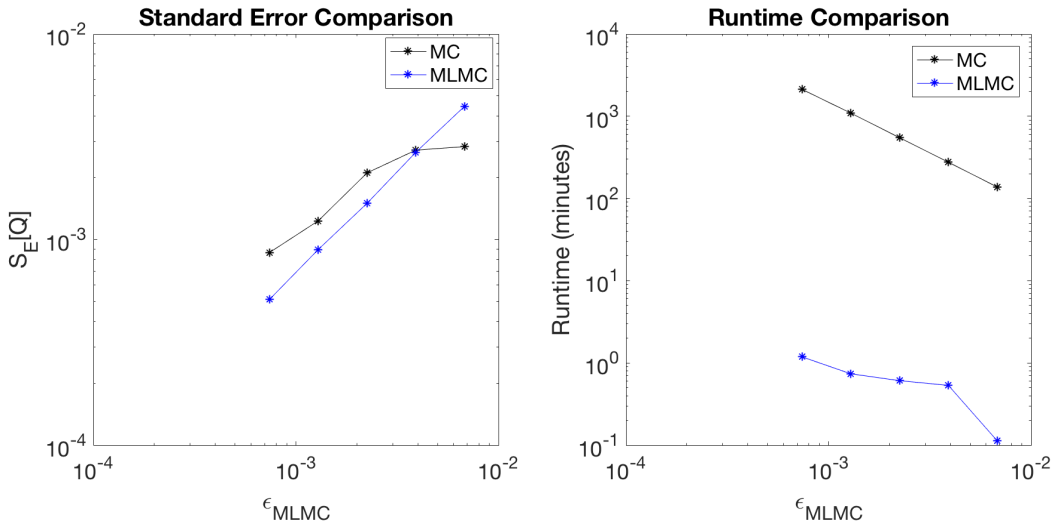



Figure 3.7: Comparing the standard error and runtime produced by MC and MLMC.

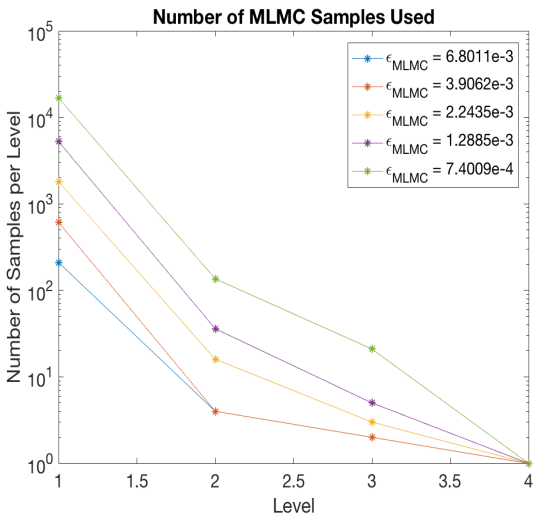



Figure 3.8: The number of samples used per level in MLMC.

## 3.5 Multilevel Quasi-Monte Carlo

Now that we have created a process that reduces the computational complexity of MC simulation, the question remains, can this process also be done for QMC simulation? In short, the answer is yes and we will now provide an overview of multilevel Quasi-Monte Carlo (MLQMC). MLQMC is implemented in a similar respect to MLMC. However, the major difference is that we now use the digital net rule created for QMC on each of the varying levels. To provide a better representation of this algorithm, please refer to Algorithm 4 below which was in part obtained from the journal articles [97] and [98]. From this algorithm overview, we see that the stopping mechanism for MLQMC is slightly changed, $N^{(\ell)} = 1$ denotes that 32 shifts of the deterministic points were used per sample, and there is no $\gamma$ value as in MLMC. In this algorithm, the bias is set to be $(Q^{(L)} - Q^{(L-1)})^2$, where $L$ is the largest level obtained within the algorithm. Another note that should be made on this algorithm is that 32 shifts is often an optimal amount of shifts for the MLQMC algorithm, even though 16 shifts was optimal for QMC. Along with this fact, it should also be noted that setting the initial target of 1 sample per level is not required, it is often beneficial in practice to use more than one initial target sample if a small $\epsilon_{MLQMC}$ is required.

**Algorithm 4** Pseudocode for MLQMC

Start with $L = 0, 1, 2$, and an initial target of 1 sample for each level,
using 32 random shifts
**while** $\sum_{\ell=0}^{L} V^{\ell} > \epsilon^2_{MLQMC}/2$ **do**
    Double $N^{(\ell)}$ on the level with the largest standard error
    **if** $L < 2$ or the bias $> \epsilon_{MLQMC}/\sqrt{2}$ **then**
        Set L = L + 1
    **end if**
**end while**

If we refer to Figure 3.6, we see that an interlacing factor of 1 has a convergence rate of $O(N^{-1.2})$ and an interlacing factor of 2 has a convergence rate of $O(N^{-2})$. Thus, when mimicking the behavior of QMC with MLQMC, we will choose $\epsilon_{MLQMC} = N^{-1.2}$ for an interlacing factor of one and for an interlacing factor of two we would choose $\epsilon_{MLQMC} = N^{-2}$. Using this setup, we then obtain the results in Figure 3.9 and Figure 3.10. From these results we see that MLQMC is producing around the same standard error as QMC for an interlacing factor of one, while reducing the computational cost significantly. Along with this, we also see that the number of samples required per level decreases as the level increases.

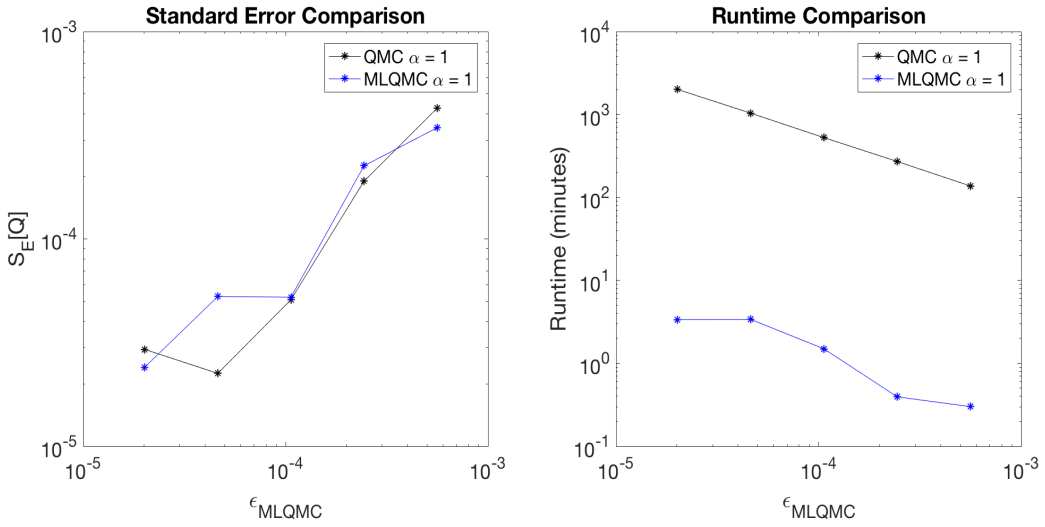



Figure 3.9: Comparing the standard error and runtime produced by QMC and MLQMC for $\alpha = 1$.

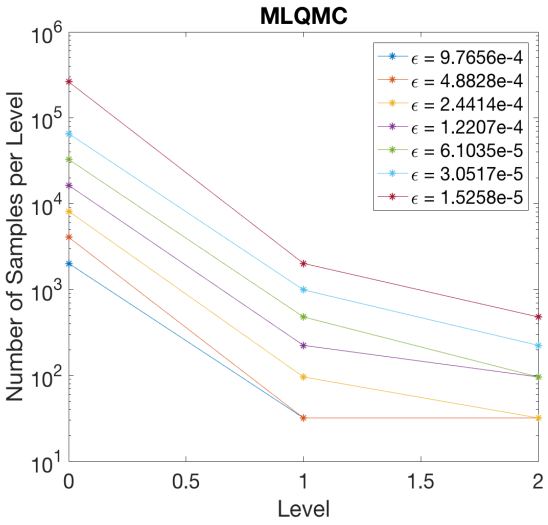



Figure 3.10: The number of samples used per level in MLQMC for $\alpha = 1$.

# CHAPTER 4

# THE ALLEN-CAHN MODEL

In this chapter, we consider a celebrated phenomenological nonlinear mathematical model describing processes such as the phase separation in the mixing and creation of alloys. It should be noted that a majority of the material in this section was obtained from our article [99] (for copyright documentation see Appendix E). This model was proposed about four decades ago by Allen and Cahn [36] and is appropriately named the Allen-Cahn (A-C) model. The A-C model is composed of a nonlinear space-time partial differential equation (PDE), initial state, and boundary conditions [37, Chapter 6]. The A-C system has been widely investigated in literature for its relation to several physical processes such as crystal growth [38], population dynamics [39], image segmentation [40], and motion by mean curvature [41–43].

In the A-C model, the phase is not uniquely determined by physical quantities such as temperature and pressure. Instead, the A-C model describes the evolution of an order parameter field, where the field is used to determine the phase. One of the key features of the A-C model is on the description of the evolution of complex initial structures of the order field to a simple desired state. Numerical simulations of such key features from known deterministic initial states have been established in several articles, see for example [38, 40, 42, 44] and the extensive references therein.

It is also known that the assumption of complete knowledge of the A-C system is not appropriate. As a consequence, stochastic counterparts of the A-C equation have been proposed, see for example [45–47] and references therein for theoretical investigations of various mathematical properties associated with such models. However, numerical investigations to quantify uncertainties (UQ) in stochastic A-C systems are yet to be investigated in the literature. Stochastic UQ computational models have been established for several linear systems such as those modeling diffusion, financial, and wave propagation processes. See for example [13–22],

which consider the simulation of statistical moments of various quantities of interest (QoI).
The main aim of this article is to develop and implement a hybrid high-order multi-level stochastic A-C computational model, in conjunction with finite element in space-time and efficient sampling of the high-dimensional stochastic space, to compute statistical moments of a QoI induced by the A-C output field. The input uncertainty in the model occurs both in the A-C PDE and in the initial state that facilitates the stochastic evolutionary process. Our hybrid deterministic-stochastic approximation framework is based on a space-time finite element method in two and three spatial dimensions, high-order approximations in high-dimensional stochastic variables, and fast evaluations using multi-level and parallel computations.
As stated in Chapter 3, the classical and widely used approach to compute statistical moments (such as the expected value) of a stochastic QoI in high-dimensional spaces, is the celebrated Monte Carlo (MC) method. A major disadvantage of the MC method, using an $N$-term approximation, is its low-order $O(N^{-1/2})$ convergence. This disadvantage becomes apparent when simulating a statistical moment of a QoI modeled by a space-time stochastic partial differential equation (SPDE). In such models this low-order convergence makes MC simulation computationally prohibitive when high accuracy is desired. A relatively recent approach used to overcome the computational cost incurred by MC simulation for SPDEs, is the multi-level MC (MLMC) algorithm which works in conjunction with space-time approximations of the deterministic SPDE. The MLMC algorithm does not improve the accuracy of MC approximations, but facilities practical MC simulations with low-order accuracy. The survey article [17] describes marked advantages of the MLMC approach and consider it to be the state-of-the-art for SPDE simulations.
In this chapter, we use an operator-splitting finite element method (FEM) framework that facilitates analytical treatment of the nonlinearity in the A-C system through a time-stepping iterative process. The advantage of said approach is that it requires only a solution of the linear algebraic system at each time-step, rather than Newton-type iterations or extrapolation approximations of the nonlinear term. Based on this framework, we then develop an FEM-MC

based approach for simulating the expected value of a QoI of the A-C SPDE system and demonstrate that the MC-based approach is not practical for the stochastic A-C system.

The main aim of the chapter is to go beyond the low-order MC approximations. In particular, we use high-order digitally-shifted quasi-MC (QMC) $N$-term stochastic variable approximations, for the A-C SPDE. The standard QMC approximations offer $\mathcal{O}(N^{-d})$ convergence, for some $1/2 < d < 1$ [16]. High-order ($d > 1$) convergence can be achieved using an *interlaced* version of QMC (iQMC) [16, 19]. For our application, the order of convergence of iQMC depends on the stochastic variable regularity of the A-C SPDE QoI. Mathematically establishing such stochastic regularity properties for nonlinear space-time SPDEs, such as the A-C model considered in this chapter, is still an open problem. Such stochastic variable regularity results have been obtained and computationally demonstrated only for relatively simple stochastic diffusion models, see [16, 19] and references therein.

Using iQMC approximations, this is the first article to computationally demonstrate the limited stochastic regularity of the QoI induced by a class of A-C systems, with complex but smooth random initial states and uncertain gradient energy in the A-C SPDE. In particular, unlike in the simple diffusion model cases [19], our A-C SPDE iQMC-FEM algorithm and implementation demonstrate that, because of the limited parameter regularity of the stochastic A-C model QoI, achieving the $\mathcal{O}(N^{-d})$ convergence with $d > 1.5$ is not possible. We demonstrate that this restriction holds even with interlaced digitally-shifted Sobol′ iQMC sampling points.

Despite this (relatively high-order) convergence limitation, we further our contribution by substantially improving the computational time and cost of our QMC-FEM and iQMC-FEM A-C numerical models. In this chapter we develop and implement a digitally-shifted multi-level variant of our FEM-QMC and FEM-iQMC algorithms for the stochastic A-C system. More precisely, we demonstrate over one thousand times efficiency (in accuracy and cpu-core computational time) can be obtained using our stochastic A-C FEM-MLQMC algorithm. This is a marked advantage when compared to the state-of-the-art FEM-MLMC and even the FEM-QMC and FEM-iQMC algorithms, for the two and three spatial dimensional stochastic A-C system.

We achieve such an efficiency through appropriate trade-offs of the convergence property of our modified FEM A-C approximations and the QMC/iQMC approximations. Numerical analysis of the Sobol′ QMC/iQMC approximations for SPDEs still remains an open problem even for simple stationary diffusion models. Such stochastic numerical analyses are restricted to the QMC samples generated using precise bounds on the stochastic regularity of the QoI arising from relatively simple linear diffusion-type models, see the survey articles [16, 19], and extensive references therein. The computational framework in this article might facilitate substantial future theoretical developments on high-order stochastic approximations for applied model problems such as the A-C SPDE system.

The rest of this chapter is organized as follows: In the next section, we introduce a stochastic version of the A-C PDE. In Section 4.2, we describe a modified Crank-Nicolson finite element method for the computational modeling of realizations of the A-C PDE. Subsequently using numerical experiments, we demonstrate the accuracy and convergence of the deterministic numerical A-C model for several realizations of the uncertain parameters in the example models. In Section 4.3, we develop MC, QMC, iQMC, and multi-level QMC approximation methods based stochastic computational models for the A-C SPDE, in conjunction with the modified FEM approximations. Using this hybrid framework, we demonstrate marked advantages of the high-order and multi-level stochastic A-C computational models by simulating the A-C SPDE in two and three spatial dimensions, and high stochastic dimensions.

## 4.1 A Stochastic Allen-Cahn Model

We are interested in computing statistical moments of a QoI arising from a stochastic evolutionary order parameter field modeled by the A-C PDE with uncertainty in the input governing the model. The field is induced by a random gradient energy and a random initial state. The non-deterministic nature of the energy and interface are represented by a random vector $\boldsymbol{\omega}$. We let $\boldsymbol{\omega} \in \Omega \subseteq \mathbb{R}^s$ denote an outcome in a probability space $(\boldsymbol{\Omega}, \boldsymbol{\mathcal{F}}, \mathbb{P})$, where $\boldsymbol{\mathcal{F}}$ is a Borel $\sigma$-algebra and the probabilities of the events in $\boldsymbol{\mathcal{F}}$ are given by the probability measure $\mathbb{P}$, and $s$ is

a stochastic dimension (that is larger than the physical spatial dimension $r = 2, 3$).

For $\boldsymbol{\omega} \in \Omega$, we use notation $\epsilon_{AC}^2(\boldsymbol{\omega})$ to denote the random gradient energy in the stochastic A-C evolutionary process introduced below. The stochastic A-C system is induced by a random spatially dependent initial state $G(\mathbf{x}; \boldsymbol{\omega})$, $\mathbf{x} \in D$, where $D \subset \mathbb{R}^r$ is a polygonal/polyhedral spatial two/three-dimensional domain. In the A-C model, we take the widely used Helmholtz free energy density (double-well potential) $F(c) = (c^2 - 1)^2/4$ with $F(\pm 1) = 0$, and its derivative $f(c) = c^3 - c$ is monotone outside $[-1, 1]$. We use the standard assumption that the mobility in the model is constant and scaled to unity.

Using this framework, the evolution of the phase representative order parameter field $c(\mathbf{x}, t; \boldsymbol{\omega})$ from its initial state $G$, at time $t = 0$ to a final time $T$, is modeled by the nonlinear stochastic A-C system with zero flux boundary conditions:

$$\frac{\partial c(\mathbf{x}, t; \boldsymbol{\omega})}{\partial t} = \frac{c(\mathbf{x}, t; \boldsymbol{\omega}) - c^3(\mathbf{x}, t; \boldsymbol{\omega})}{\epsilon_{AC}^2(\boldsymbol{\omega})} + \Delta c(\mathbf{x}, t; \boldsymbol{\omega}), \qquad \mathbf{x} \in D,\ \boldsymbol{\omega} \in \boldsymbol{\Omega},\ 0 < t \le T, \tag{4.1}$$

$$c(\mathbf{x}, 0; \boldsymbol{\omega}) = G(\mathbf{x}; \boldsymbol{\omega}), \qquad \boldsymbol{\omega} \in \boldsymbol{\Omega},\ \mathbf{x} \in D, \tag{4.2}$$

$$\frac{\partial c}{\partial \mathbf{n}} = 0, \qquad \text{on } \partial D. \tag{4.3}$$

For details on the well-posedness and related theoretical details of the deterministic counterpart A-C model, see [37, Chapter 6] and references therein.

It is easy to see that $c = -1$ and $c = +1$ are non-trivial stationary solutions of the semi-linear parabolic PDE (4.1), corresponding to the minima of the double-well potential $F$. The time-dependent solution of the A-C system transitions between the two minima and develops interfaces, and the rate of transition is controlled by $\epsilon_{AC}$. The zero level sets of the solution $c$ are known as interfaces, and the diffuse interface is a region of width $\epsilon_{AC}$ around the interface that separates the phases [37, Chapter 6].

The randomness of the gradient energy $\epsilon_{AC}(\boldsymbol{\omega})$ adds substantial complexity to mathematical investigations to quantify the transition region. For a fixed realization $\boldsymbol{\omega} \in \Omega$, the initial state $G(\cdot; \boldsymbol{\omega})$ induces the evolutionary process of the deterministic counterpart of the A-C

initial-boundary value problem (4.1)-(4.3). Corresponding to a fixed realization and time, throughout this chapter, we visualize the solution by plotting its interface, namely the zero level set of the (FEM approximate) solution of the sampled A-C system.

In Figure 4.1, we demonstrate example realizations of the stochastic initial state $G$ in two and three dimensional domains, using its zero level sets. The realized initial state interfaces in two

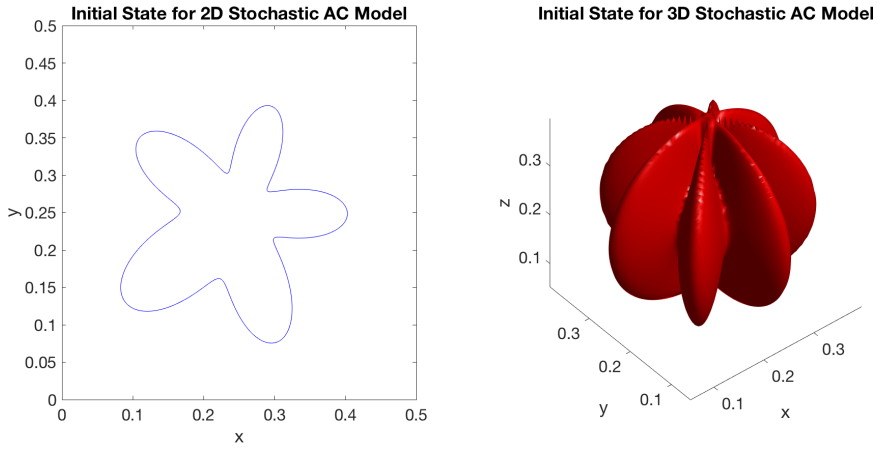



Figure 4.1: Realizations of two random initial states in two and three spatial dimensions

and three dimensional domains are from $s = 34$ and $s = 26$ dimensional probability spaces, respectively. Details of such uncertain initial states and associated evolution processes will be explained later in this chapter. From such uncertain input data, we consider the average energy QoI of the stochastic A-C system:

$$Q(\boldsymbol{\omega}) = \int_D c(\mathbf{x}, T; \boldsymbol{\omega}) \; d\mathbf{x}, \qquad \boldsymbol{\omega} \in \Omega. \tag{4.4}$$

The focus of this chapter is on computing efficient FEM approximations to $c(\mathbf{x}, T; \boldsymbol{\omega})$ for each realization $\boldsymbol{\omega} \in \Omega$, and hence develop efficient FEM-MC, QMC, and FEM-MLQMC algorithms to approximate the $s$-dimensional expected value integral

$$\mathbb{E}[Q] = \int_{\Omega} Q(\boldsymbol{\omega}) \, d\mathbb{P}(\boldsymbol{\omega}). \tag{4.5}$$

The key to multi-level approximations is to establish robust rates of convergence for the FEM approximations, as the empirically computed rates for the deterministic A-C system determine

how many levels and stochastic samples of the MC, QMC, and iQMC algorithms are needed. For a fixed realization parameter $\boldsymbol{\omega} = \boldsymbol{\omega}^*$, we use the notation $c^*(\mathbf{x}, t) = c(\mathbf{x}, t; \boldsymbol{\omega}^*)$ and $c^*(\mathbf{x}, 0) = G^*(\mathbf{x}) := G(\mathbf{x}; \boldsymbol{\omega}^*)$. In the next section, for the fixed realization deterministic A-C system, we describe and demonstrate a finite element method (FEM) based operator splitting hybrid numerical-analytical algorithm, which efficiently approximates and computes the solution $c^*(\mathbf{x}, t)$ of the nonlinear deterministic A-C system. Using a finite difference method, such an operator splitting approach and its stability properties were investigated in [44]. The hybrid nature of the algorithm facilitates solving only linear systems to compute hybrid FEM approximations to $c^*(\mathbf{x}, t)$ satisfying the nonlinear A-C model.

The hybrid method and tabulated deterministic simulation results for various space-time mesh sizes provide a framework to apply multi-level versions of stochastic realizations of the uncertain A-C SPDE model. The multi-level approach facilitates appropriate trade-offs between a large number to a fewer number of MC/QMC/iQMC samples in conjunction, respectively, with coarse and fine mesh discretizations.

### 4.2 FEM approximations for deterministic A-C system

The deterministic, fixed realization, counterpart of the stochastic A-C system (4.1)–(4.3) is:

$$\frac{\partial}{\partial t} c^*(\mathbf{x}, t) = \frac{c^*(\mathbf{x}, t) - (c^*)^3(\mathbf{x}, t)}{(\epsilon^*_{AC})^2} + \Delta c^*(\mathbf{x}, t), \ \mathbf{x} \in D, \ 0 < t \leq T, \tag{4.6}$$

$$c^*(\mathbf{x}, 0) = G^*(\mathbf{x}), \ \mathbf{x} \in D, \tag{4.7}$$

$$\frac{\partial c^*}{\partial \mathbf{n}} = 0 \text{ on } \partial D. \tag{4.8}$$

For well-posedness of the above deterministic A-C system, with weak-solution in $H^1(D)$, see [37, Chapter 6].

Following a finite difference (and a strong-solution based) approach in [44], we split (4.6) into two related systems separating out the nonlinear term from a linear space-time model, and consider these in weak sense. The resulting continuous linear system is governed by the heat equation and the cubic term in the A-C system is included in a nonlinear dynamical system

(without the spatial differential operator). More precisely, for notational convenience, using again $c^*$ to denote approximations at the continuous level for the linear-nonlinear split systems, we consider the two associated PDEs (with an implicit connection between the first and second):

$$\frac{\partial}{\partial t}c^*(\mathbf{x},t) = \Delta c^*(\mathbf{x},t), \tag{4.9}$$

$$\frac{\partial}{\partial t}c^*(\mathbf{x},t) = \frac{c^*(\mathbf{x},t)-(c^*)^3(\mathbf{x},t)}{(\epsilon^*_{AC})^2}. \tag{4.10}$$

It is useful to note that the weak solution of the heat equation system (4.7)–(4.9) is relatively smooth [37, Chapter 6]. In particular, if $G^* \in H^1(D)$, the weak solution of (4.7)–(4.9) is in $H^2(D)$. Hence, second-order FEM approximations can be obtained for (4.7)–(4.9).

Below we describe details of the hybrid approximate-analytical approach, illustrated in Figure 4.2, that is based on a modified Crank-Nicolson (M-C-N) Galerkin finite element method (FEM) for the deterministic solution of the nonlinear A-C system (4.6)-(4.8).

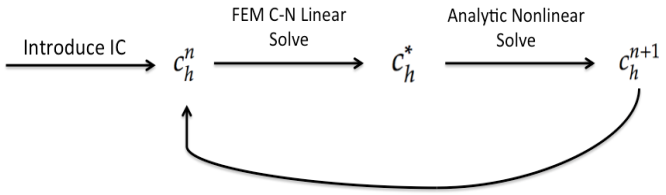



Figure 4.2: An overview of the modified FEM operator splitting algorithm.

### 4.2.1 The M-C-N Galerkin FEM approximation of the A-C system

For $\psi_1, \psi_2 \in L^2(D)$, the standard $L^2$ inner product and norm are

$$\langle \psi_1, \psi_2 \rangle = \int_D \psi_1\, \psi_2, \qquad\qquad \|\psi_1\|^2 = \langle \psi_1, \psi_1 \rangle.$$

Using this inner product and norm, it is easy to see that the standard variational form of the heat equation system (4.7)-(4.9) is:

For each fixed $t \in (0, T]$, find $c^* \in H^1(D)$ such that

$$\left\langle \frac{\partial c^*}{\partial t}, v \right\rangle = -\langle \nabla c^*, \nabla v \rangle, \qquad \text{for all } v \in H^1(D), \tag{4.11}$$

$$c^*(\mathbf{x}, 0) = G^*(\mathbf{x}), \qquad \mathbf{x} \in D. \tag{4.12}$$

More specifically, this is obtained by multiplying the PDE (4.9) by the test functions $v \in V$, providing:

$$\int_D \frac{\partial c^*}{\partial t}\, v\, d\mathbf{x} - \int_D \Delta c^*\, v\, d\mathbf{x} = 0$$

$$\iff \int_D \frac{\partial c^*}{\partial t}\, v\, d\mathbf{x} - \int_D (\nabla \cdot \nabla c^*)\, v\, d\mathbf{x} = 0m$$

integration by parts provides

$$\int_D \frac{\partial c^*}{\partial t}\, v\, d\mathbf{x} = \Big[ - \int_D \nabla c^* \nabla v\, d\mathbf{x} + \int_{\partial D} v\, \nabla c^* \cdot \mathbf{n}\, d\mathbf{x} \Big]. \tag{4.13}$$

The boundary condition $\dfrac{\partial c^*}{\partial \mathbf{n}} = 0$ on $\partial D$, allows equation (4.13) to become,

$$\int_D \frac{\partial c^*}{\partial t}\, v\, d\mathbf{x} = - \int_D \nabla c^* \nabla v\, d\mathbf{x}.$$

Using the initial condition (4.7) this statement provides the above variational formulation.

As an intermediary approximation of the solution to the A-C system, for a fixed time-step, we are first interested in approximating the weak-solution $c^*$ of (4.11)-(4.12) in a finite dimensional subspace of $H^1(D) \cap C(D)$. For spatial approximations, using a discretization parameter $0 < h < 1$ (that represents the maximum of the diameters of the tessellated subsets of $D$) we consider $V_h \subset H^1(D) \cap C(D)$, a finite dimensional subspace spanned by continuous linear splines (defined on the tessellation), satisfying the approximation theory property that for any $v \in H^s(D)$, $s \geq 2$,

$$\inf_{w_h \in V_h} \|v - w_h\| = O(h^2). \tag{4.14}$$

In (4.14), the order constant depends on the bounds of an appropriate norm of $v$, which also depends on the regularity of $v$. For this reason, the second-order rate of convergence reduces in cases where $v$ has lower regularity, say, $v \in H^s(D) \cap C(D)$, for some $1 \leq s < 2$. While the heat

equation solution has at least $H^2(D)$ regularity for smooth initial states, the weak solution of the A-C system in general has lower regularity that depends in a complex way on various input parameters. In the next subsection, we demonstrate the latter using numerical experiments. Assuming the initial data $G^* \in H^1(D) \cap C(D)$, we first project the initial data $G^*$ into $V_h := \text{span}\,\{\phi_i : i = 1, \ldots, N_h\}$.

We start the algorithm with a uniform mesh on the time interval $[0, T]$ with $N_{\Delta t} + 1$ discrete time-steps: $t_n = n\Delta t,\ n = 0, \ldots, N_{\Delta t}$ with $\Delta t = T/N_{\Delta t}$, so that the final simulation time is $t_{N_{\Delta t}} = T$. In order to obtain a central-difference second approximation of $\dfrac{\partial}{\partial t}$ using the above time-grid, we consider the PDE (4.11) at the time-grid mid-points $t_{n-1/2} = (n - 1/2)\Delta t$, for $n = 1, \ldots, N_{\Delta t}$ with step-size $\Delta t/2$ so that the unknowns in the approximations occur at the integer-indexed time-levels $t_n,\ n = 1, \ldots, N_{\Delta t}$. For a fixed $n = 1, \ldots, N_{\Delta t}$, to approximate the solution $c^*(\cdot, t_{n-1/2})$ of (4.11), we start with the C-N idea and use a second-order average approximation $c^*(\cdot, t_{n-1/2}) \approx [c^*(\cdot, t_n) + c^*(\cdot, t_{n-1})]/2$, and modify this using the nonlinear dynamical system solution at $t_{n-1}$ with a hybrid analytical approach.

The initial step of the algorithm uses the data $c_h^0 = P_h G^*$, where $P_h : H^1(D) \cap C(D) \to V_h$ denotes the projection operator. For each fixed $k = 0, \ldots, N_{\Delta t}$, we consider $c_h^* \in V_h$ with ansatz

$$c_h^*(\mathbf{x}) = \sum_{i=1}^{N_h} \alpha_i^k \phi_i(\mathbf{x}), \qquad \mathbf{x} \in D, \tag{4.15}$$

and the unknown $N_h$-dimensional coefficient vector $\boldsymbol{\alpha}^k$ with $i$-th entry $\alpha_i^k$. The coefficient vector $\boldsymbol{\alpha}^k$ will be governed by a linear system arising from the M-C-N FEM approach, as described below. For $k = 0$, we compute $\alpha_i^0,\ i = 1, \ldots N_h$ such that $c_h^* = c_h^0$ so that $\boldsymbol{\alpha}^0$ is known, when considering the M-C-N FEM procedure. Subsequently, for each fixed $k > 1$, we obtain a linear algebraic system for the unknown vector $\boldsymbol{\alpha}^k$ using a coupled analytical-numerical framework that is based on second-order fully-discrete M-C-N Galerkin approximations of the unknown $c_h^*$. The numerical part of the coupled system is obtained by the aforementioned approximation of the Galerkin variational formulation (4.11) at the time-grid mid-points with step-size $\Delta t/2$ and a second-order average approximation of $c^*(\cdot, t_{n-1/2})$.

More precisely, for a fixed $k = 1, \ldots, N_{\Delta t}$, we obtain the algebraic system for unknown vector $\boldsymbol{\alpha}^k$ using the following analytical-numerical coupled fully-discrete M-C-N Galerkin linear system, for the numerical unknown $c_h^*$ in conjunction with $c_h^0 = P_h G^*$ for $k = 1$, and for $k > 1$ in conjunction with the analytical solution to the nonlinear dynamical system (4.10) :

$$\left\langle \frac{c_h^* - c_h^{k-1}}{\Delta t}, v \right\rangle = -\left\langle \frac{\nabla c_h^* + \nabla c_h^{k-1}}{2}, \nabla v \right\rangle, \qquad \text{for all } \; v \in V_h, \tag{4.16}$$

where for $\ell = 1, \ldots, N_{\Delta t}$, using the solution $c_h^*$ of (4.16) (with $k = \ell$ and $c_h^0 = P_h G^*$),

$$c_h^\ell = \frac{c_h^*}{\sqrt{e^{-2t/(\epsilon_{AC}^*)^2} + (c_h^*)^2[1 - e^{-2t/(\epsilon_{AC}^*)^2}]}}. \tag{4.17}$$

In (4.17), $c_h^\ell$ is the analytical solution of the dynamical system (4.10), obtained using the initial condition $c_h^*$ [that is the approximate solution of the system (4.7)–(4.9)]. Thus for $k = 0, \ldots, N_{\Delta t}$, the computable M-C-N FEM solution $c_h^k \in V_h$, is our approximation to the exact solution of the deterministic A-C system (4.6)-(4.8) at $t_k$.

Clearly, the main cost for computing $c_h^k \in V_h$ is solving the linear algebraic system arising from the Galerkin formulation (4.16) for the unknown vector $\boldsymbol{\alpha}^k$. Using the ansatz (4.15) in (4.16), for each fixed $k = 1, \ldots, N_{\Delta t}$, the algebraic system is governed by the $N_h \times N_h$ sparse symmetric and positive definite matrix $\left[B + \frac{\Delta t}{2} A\right]$, where $A$ and $B$ are the standard FEM stiffness and mass matrices with respective entries $[A]_{i,j} = \langle \nabla \phi_i, \nabla \phi_j \rangle$, $[B]_{i,j} = \langle \phi_i, \phi_j \rangle$, $i, j = 1, \ldots, N_h$.

Since the governing matrix is independent of the time-step $k$, it is efficient to first obtain a Cholesky factorization of the sparse matrix $\left[B + \frac{\Delta t}{2} A\right]$ and hence use cheaper substitution solves at each discrete time iteration step $k$. It is important to note that for simulating the physical process governed by the A-C system, the final time $T$ depends on the evolution of an unknown process starting from a complex structure such as in Figure 4.1. Thus, the final time-step should be chosen with a stopping criteria, such as the interface reaching a simple/desired interface state, say, a circle or a sphere with positive radius (and hence avoid the interface collapsing to a single point). In particular for the system simulation, the final time $T$ is dictated by the application quantity of interest and not *a priori* by the chosen time-grid.

A finite-difference counterpart of the above M-C-N FEM scheme was demonstrated to be stable (for long time simulation) in [44], for some example deterministic A-C systems. This was subject to the criteria that for a chosen uniform spatial mesh parameter $h$, the time-grid parameter $\Delta t$ be of the order $10^{-3}h$, to obtain $O(h^2)$ convergence. Motivated by the numerical study in $\Delta t$, we chose a similar $\Delta t$ choice that depends on $h$ for our deterministic example model problems.

The broad focus in this chapter is on the stochastic A-C system. The multi-level stochastic simulation algorithm we construct crucially depends on the knowledge of the expected accuracy for various choices of $h$. Numerical analysis of the convergence and stability of the above M-C-N FEM scheme for the deterministic system is beyond the scope of this thesis. Numerical results in the next sections demonstrate restricted order of convergence as the input parameter choices become more complex. This demonstrates the limited regularity of the weak solution of the A-C system. In particular, as demonstrated below, for a certain choice of input parameters we obtain second-order convergence of the M-C-N FEM scheme. Following these results, we see the convergence rate is reduced as the parameter choice was changed, leading to a more complex initial state.

### 4.2.2 Numerical experiments I: Deterministic A-C system

Before we consider the A-C system of interest, we will first take a look at an example where we know the true solution and compare our M-C-N FEM scheme against it. For this example, we will consider the travelling wave solutions of the A-C system as considered in [44] in 2D. By inspection, one can see that (4.18) satisfies the A-C system (4.6) – (4.8) with $G^*(\mathbf{x}) = c(x, y, 0)$. In (4.18), $\epsilon_{tw} = 0.015$ (the gradient energy coefficient) and $s = 3/(\sqrt{2}\epsilon_{tw})$. For our simulations, we will let the final time be given by $T = 1.25/s$ and our domain be $D = (-0.5, 1.5) \times (-0.5, 1.5)$. We demonstrate convergence by computing the $N_h$-dimensional vector with entries $[\mathbf{e}_h]_n = \left| c_h^{N_{\Delta t}}(\mathbf{x}_{node,n}) - c_{exact}(\mathbf{x}_{node,n}) \right|,\ n = 1, \ldots, N_h$. Hence we can calculate the root mean-square (RMS) error $\|\mathbf{e}_h\|_2$ of the approximate solution. We tabulate the RMS errors and also compute the estimated order of convergence (EOC) from the RMS errors. Here, $c_h^{N_{\Delta t}}$ is defined by the M-C-N

algorithm based formula in (4.17), at the final time $t_{N_{\Delta t}} = T$. In Table 4.1 we provide the convergence results for the 2D travelling wave problem, which show the order two convergence of the proposed scheme. For these results, we choose $h = 2/(N_h - 1)$ and $\Delta t = T/(2N_h - 2)$ to ensure stability. Additionally in Figure 4.3, we provide the numerical solution of the travelling wave problem at various times.

$$c_{exact}(x, y, t) = \frac{1}{2}\left(1 - \tanh\left(\frac{x - st}{2\sqrt{2}\epsilon_{tw}}\right)\right) \tag{4.18}$$

Table 4.1: Convergence of 2D M-C-N FEM solutions for the travelling wave problem.

| $N_h$ | $\|\mathbf{e}_h\|_2/\|c_{exact}\|_2$ | EOC of $\|\mathbf{e}_h\|_2/\|c_{exact}\|_2$ |
|---|---|---|
| 64 | 1.2510e-1 | – |
| 128 | 3.2651e-2 | 1.9379 |
| 256 | 7.8545e-3 | 2.0555 |
| 512 | 1.8330e-3 | 2.0993 |
| 1024 | 4.5887e-4 | 1.9981 |

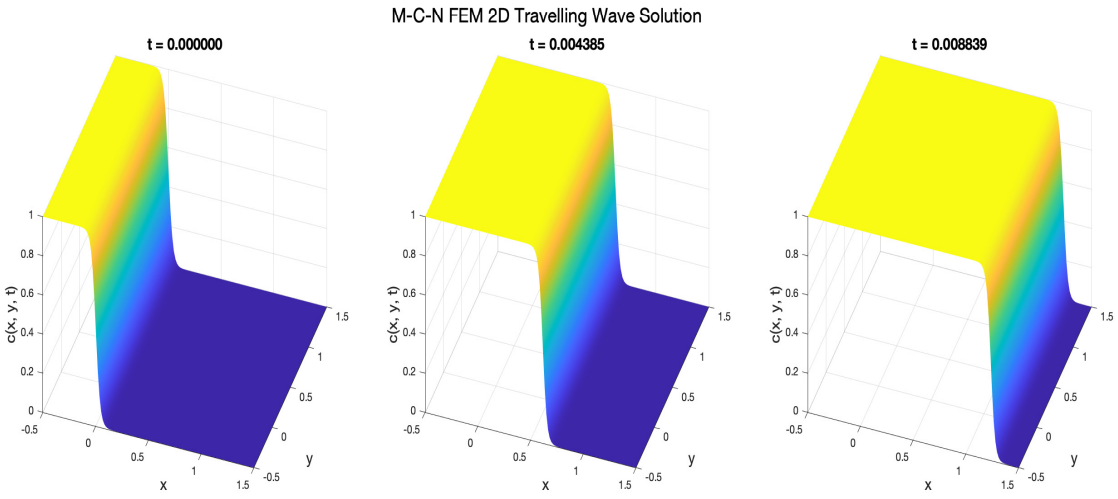



Figure 4.3: 2D M-C-N approximate A-C system solution for the travelling wave problem from the initial solution to the final time $T$.

### 4.2.3 Numerical experiments II: Deterministic A-C system

In this section we consider fixed realizations, such as those in Figure 4.1 in two- and three-spatial dimensions, of the random initial state defined by the following representations with several random parameters. The uncertainty in parameters, leading to realization of randomness, are indicated below by the symbol $\boldsymbol{\omega}$. In the two-dimensional case for $\mathbf{x} = (x, y) \in D \subset \mathbb{R}^2$, based on a standard representation used to produce non-trivial initial states for deterministic A-C numerical experiments [44], the example uncertain initial state for our numerical experiments is represented as

$$G(x, y; \boldsymbol{\omega}) = \sum_{i=1}^{L_{AC}} \sum_{j=1}^{L_{AC}} g_{ij}(x, y; \boldsymbol{\omega}), \tag{4.19}$$

where

$$g_{ij}(x, y; \boldsymbol{\omega}) = \tanh \frac{0.125 + \delta_i(\boldsymbol{\omega}) \cos(\eta(\boldsymbol{\omega})\theta) - \sqrt{(x - \gamma_j(\boldsymbol{\omega}))^2 + (y - \gamma_j(\boldsymbol{\omega}))^2}}{\sqrt{2}\epsilon_{AC}(\boldsymbol{\omega})}. \tag{4.20}$$

For the three-dimensional counterpart of the above, with $\mathbf{x} = (x, y, z) \in D \subset \mathbb{R}^3$, we take

$$G(x, y, z; \boldsymbol{\omega}) = \sum_{i=1}^{L_{AC}} \sum_{j=1}^{L_{AC}} \sum_{k=1}^{L_{AC}} g_{ijk}(x, y, z), \tag{4.21}$$

$g_{ijk}(x, y, z; \boldsymbol{\omega}) =$

$$\tanh \frac{0.125 + \delta_i(\boldsymbol{\omega}) \cos(\eta(\boldsymbol{\omega})\theta) - \sqrt{(x - \gamma_j(\boldsymbol{\omega}))^2 + (y - \gamma_j(\boldsymbol{\omega}))^2 + (z - \mu_k(\boldsymbol{\omega}))^2}}{\sqrt{2}\epsilon_{AC}(\boldsymbol{\omega})}. \tag{4.22}$$

For both the spatial dimensions, in (4.20) and (4.22), we take

$$\theta = \begin{cases} \tan^{-1}\left(\dfrac{y - 0.25}{x - 0.25}\right), & \text{if } x > 0.25 \\ \pi + \tan^{-1}\left(\dfrac{y - 0.25}{x - 0.25}\right), & \text{if } x < 0.25 \\ 1.5\pi \qquad\quad , & \text{otherwise .} \end{cases}$$

In the above representations of the initial condition, the parameters $\gamma_j$ and $\mu_k$ determine the center of the object, the gradient energy $\epsilon_{AC}$ determines the thickness of interfaces separating different

phases, $\delta_i$ determines the size of the leafs representing the non-convex nature of the initial structure, and $\eta$ determines the number of leafs of the star-shaped initial state.
In this chapter, we consider all of these parameters to be uncertain and take them as uniformly distributed random variables, $\mathcal{U}(a_v, b_v)$, where $v = \gamma, \mu, \epsilon_{AC}, \delta, \eta$ is chosen respectively for the parameters $\gamma_i, \mu_k, \epsilon_{AC}, \delta_i, \eta$ for $i, j, k = 1, \dots L_{AC}$ for some integer $L_{AC}$. The value of $L_{AC}$ will in turn determine the stochastic dimension. Taking into account the distinct nature of these parameters (such as $\eta$ being an integer, $\epsilon_{AC} < 1$ is of interest in the A-C model, various shape constraints etc.) for numerical experiments we choose

$$a_\gamma = 0.17 = a_\mu, b_\gamma = 0.27 = b_\mu; a_{\epsilon_{AC}} = 0.4, b_{\epsilon_{AC}} = 0.9; a_\delta = 0.045, b_\delta = 0.055; a_\eta = 3.0, b_\eta = 8.0.$$

For our numerical simulations we choose the spatial domain to be a square or cube, respectively, with $D = [0, 0.5]^r \subset \mathbb{R}^r$, for $r = 2, 3$ and consider a uniform spatial grid with mesh-width $h$. Using $h_c$ to represent the coarsest spatial mesh in our experiments, we refine the mesh to study the accuracy and convergence rate by choosing $h = h_c/(2^m)$ for $m = 1, 2, \dots, M$, where $M$ is chosen appropriately to demonstrate the convergence. Since the exact solution of the A-C model with the above complex initial state is not available, errors are computed by taking the finest-grid solution, for example, obtained with $h = h_{ref} = h_c/(2^{M+1})$. For a chosen $h, \Delta t$, with $\mathbf{x}_{node,n},\ n = 1, \dots, N_h$ denoting the free-nodes in the FEM mesh defining the nodal spline basis for $V_h$, we demonstrate convergence by computing the $N_h$-dimensional vector with entries
$[\mathbf{e}_h]_n = \left| c_h^{N_{\Delta t}}(\mathbf{x}_{node,n}) - c_{h_{ref}}^{N_{\Delta t}}(\mathbf{x}_{node,n}) \right|,\ n = 1, \dots, N_h$. Hence we can calculate the root mean-square (RMS) error $\|\mathbf{e}_h\|_2$ of the approximate solution. We tabulate the RMS errors and also compute the estimated order of convergence (EOC) from the RMS errors. Here, $c_h^{N_{\Delta t}}$ is defined by the M-C-N algorithm based formula in (4.17), at the final time $t_{N_{\Delta t}} = T$, and similarly the reference solution $c_{h_{ref}}^{N_{\Delta t}}$ with $h = h_{ref}$. In all our numerical experiments, we choose $\Delta t = h/100$ for the 2D model, and $\Delta t = 4h/1000$ for the 3D model to ensure stability.

In Figure 4.4 and Figure 4.5 we demonstrate distinct realizations of the above uncertain initial states. This is done by letting $L_{AC} = 16, 8$ (that correspond to $s = 34, 26$ parameters in the model), respectively for the $2D$ and $3D$ cases. Additionally, in the Appendix section Figure B.1 and Figure B.2 illustrate several other realizations of both models with different samples of the stochastic parameters. These figures show the evolution of the M-C-N approximate solutions of the stochastic A-C system for fixed $\eta = 3, 5, 8$ and randomly drawn choices of the other parameters. For example, in the 2D case, the realization values of the energy gradient parameters, respectively, are $\epsilon_{AC} \approx 0.5117, 0.8420, 0.5029$ and for the 3D case $\epsilon_{AC} \approx 0.8858, 0.7692, 0.8445$. The first column of the figures show the realizations of the initial state and the subsequent columns show the evolution of the complex initial structures to a simpler structure at time $T$. These figures demonstrate substantial changes in the structure of interfaces from initial states, and the changes depend on the choice of uncertain parameters. Hence computing statistical moments of the associated stochastic process to high accuracy require a large number samples, taking into account the low-order convergence of sampling methods such as the Monte Carlo method. After describing the accuracy of the M-C-N solution for some fixed realizations, in the next section, we develop efficient stochastic computational modeling of the A-C system with uncertainties.

### 4.2.4 Convergence of M-C-N FEM solutions for 2D and 3D numerical experiments

Numerical results in this section demonstrate the convergence of the M-C-N approximate solution for the deterministic 2D and 3D experiments. The tabulated results for the deterministic simulation will play a crucial role for developing efficient multi-level stochastic computation models. This is a result of our computational stochastic models involving appropriate trade-offs between the coarse to fine mesh choices in conjunction with large to low number of stochastic realization samples in high stochastic dimensions.

The EOC results in this section for the deterministic models highlight that it is possible to obtain the expected second-order convergence for a relatively simple version of the initial states, say, $L_{AC} = 1$ in (4.19) and (4.21). However, the bounds of the A-C solution for a complex version

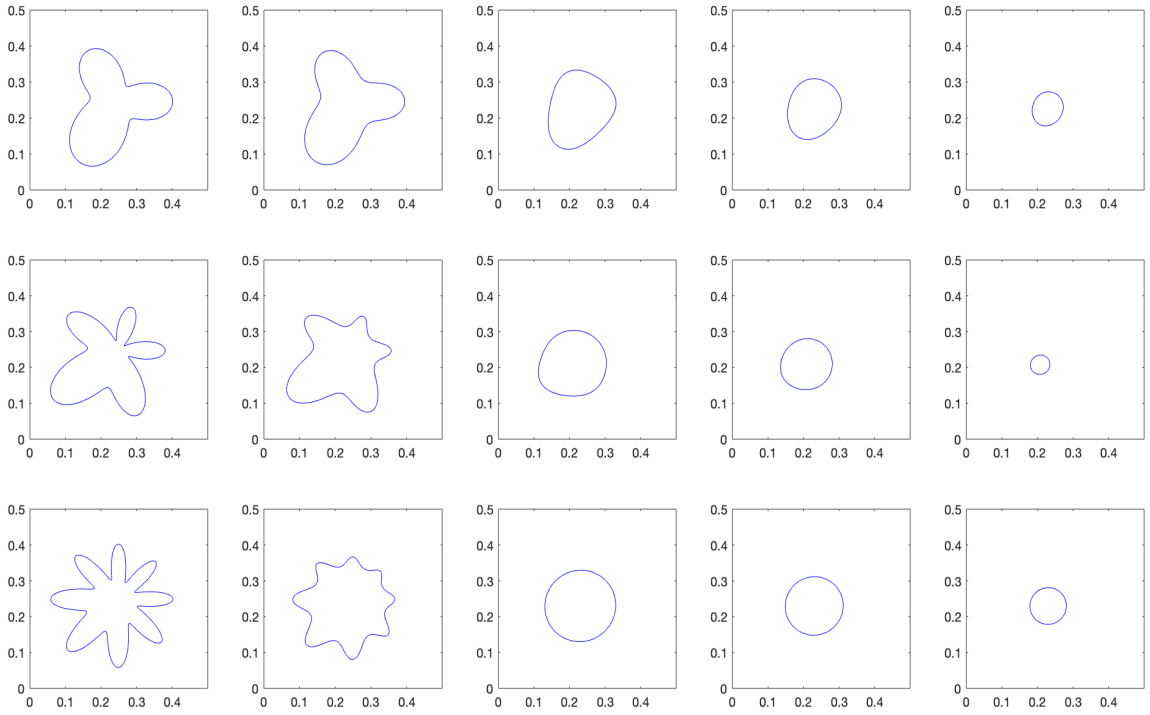


Figure 4.4: 2D M-C-N approximate A-C system solutions with $L_{AC} = 16$, $\eta = 3$ and $\epsilon_{AC} \approx 0.5117$ (top row), $\eta = 5$ and $\epsilon_{AC} \approx 0.8420$ (middle row), and $\eta = 8$ and $\epsilon_{AC} \approx 0.5029$ (bottom row).

($L_{AC} > 1$) of the initial state in (4.19) and (4.21) with several parameters does require finer grids to achieve good accuracy and also reduces the EOC. This is a result of the limited regularity of the exact solution. To demonstrate this, we first consider the 2D deterministic A-C model with $L_{AC} = 1$ in (4.19) and choose $\gamma_1 = 0.25, \delta_1 = 0.05, \eta = 3.0$, and a random value $\epsilon_{AC} \approx 0.5411$. Results in Table 4.2 clearly demonstrate the second-order convergence of the M-C-N FEM approximations.

Next, we consider the complex initial structure, with $L_{AC} = 16$ in (4.19) and for each of three choices $\eta = 3, 5, 8$, we choose one realization of other random variables in (4.19) to get three deterministic initial states for the 2D A-C system. The initial states obtained are those shown in the first column of Figure 4.4 and the evolution of these initial states governed by the A-C system is shown in the subsequent columns of Figure 4.4. The tabulated results in this section, with

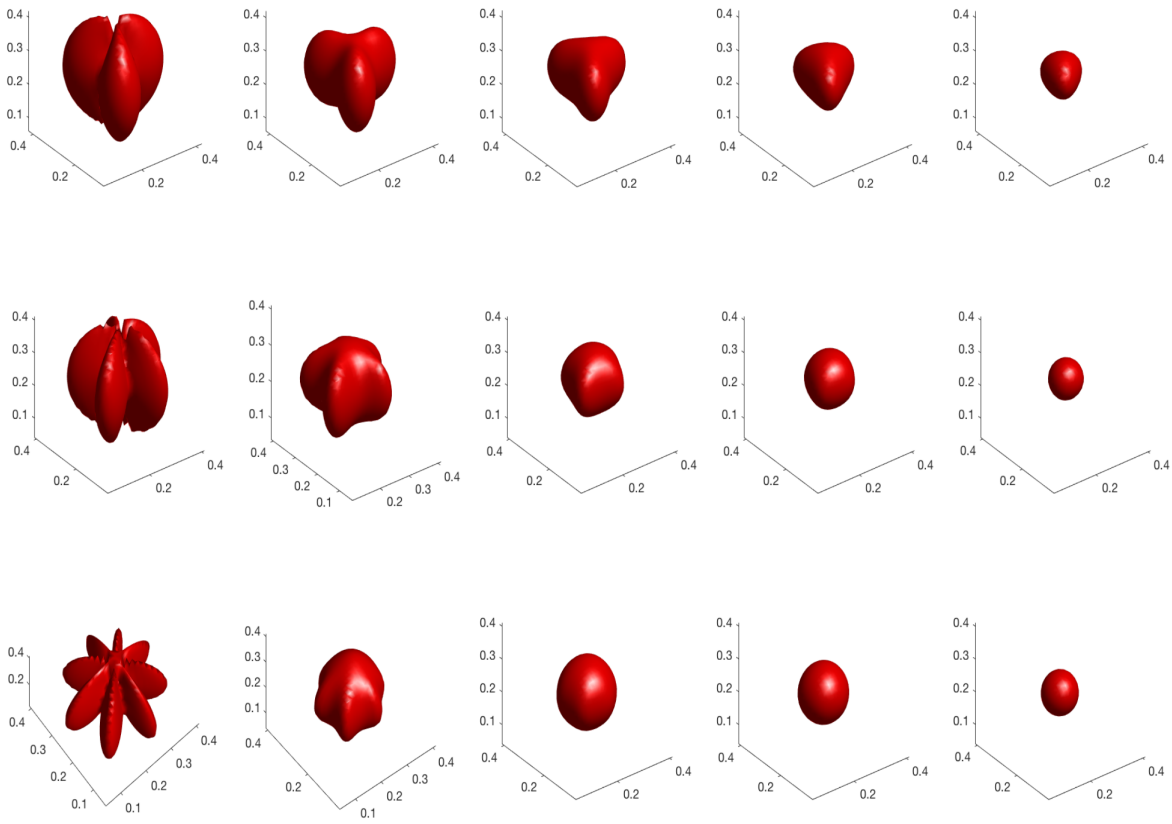


Figure 4.5: 3D M-C-N A-C system solutions with $L_{AC} = 8$, $\eta = 3$ and $\epsilon_{AC} \approx 0.8858$ (top row), $\eta = 5$ and $\epsilon_{AC} \approx 0.7692$ (middle row), and $\eta = 8$ and $\epsilon_{AC} \approx 0.8445$ (bottom row).

Table 4.2: Convergence of 2D M-C-N FEM solutions with $L_c = 1, \eta = 3, \epsilon_{AC} \approx 0.5411$.

| $h$ | $\|\mathbf{e}_h\|_2$ | EOC of $\|\mathbf{e}_h\|_2$ | $\|\mathbf{e}_h\|_2/\|c^{N_{\Delta t}}_{h_{ref}}\|_2$ | EOC of $\|\mathbf{e}_h\|_2/\|c^{N_{\Delta t}}_{h_{ref}}\|_2$ |
|---|---|---|---|---|
| $h_c = 1/40$ | 2.4981E-4 | – | 4.6864E-3 | – |
| $h_c/2$ | 6.2648E-5 | 1.9955 | 1.2268E-3 | 1.9336 |
| $h_c/4$ | 1.5704E-5 | 1.9961 | 3.1444E-4 | 1.9640 |
| $h_c/8$ | 3.9307E-6 | 1.9983 | 7.9603E-5 | 1.9819 |
| $h_c/16$ | 9.7863E-7 | 2.0059 | 1.9933E-5 | 1.9977 |

absolute and relative errors, demonstrate the convergence of the M-C-N FEM solutions. The reference solutions to compute the errors are obtained using the fine mesh-grid with $h_{ref} = h_c/32$ and $h_{ref} = h_c/16$ for the 2D and 3D simulations, respectively, $h_c$ for the 2D simulation is as in Table 4.3, Table 4.4, Table 4.5 and for the 3D simulation as in Table 4.6-Table 4.8. Compared to the results in Table 4.2, which correspond to the simple case of $L_{AC} = 1$, the complexity of the

initial state with $L_{AC} = 16$ demonstrates that substantially finer grids are required to achieve good accuracy. In addition to this, we see that the EOC is restricted. This behavior can be attributed to the complex dependence on parameters of the bounds of the A-C continuous system solution and also perhaps due to limited regularity of the exact solution.

Table 4.3: Convergence of 2D M-C-N FEM solutions with $L_c = 16, \eta = 3, \epsilon_{AC} \approx 0.5117$.

| $h$ | $\|\|\mathbf{e}_h\|\|_2$ | EOC of $\|\|\mathbf{e}_h\|\|_2$ | $\|\|\mathbf{e}_h\|\|_2/\|\|c^{N_{\Delta t}}_{h_{ref}}\|\|_2$ | EOC of $\|\|\mathbf{e}_h\|\|_2/\|\|c^{N_{\Delta t}}_{h_{ref}}\|\|_2$ |
|---|---|---|---|---|
| $h_c = 1/80$ | 9.8483E-3 | – | 4.9848E-3 | – |
| $h_c/2$ | 4.6558E-3 | 1.0808 | 2.3985E-3 | 1.0554 |
| $h_c/4$ | 2.1784E-3 | 1.0957 | 1.1324E-3 | 1.0828 |
| $h_c/8$ | 9.3966E-4 | 1.2130 | 4.9067E-4 | 1.2065 |
| $h_c/16$ | 3.1508E-4 | 1.5764 | 1.6490E-4 | 1.5731 |

Table 4.4: Convergence of 2D M-C-N FEM solutions with $L_c = 16, \eta = 5, \epsilon_{AC} \approx 0.8420$.

| $h$ | $\|\|\mathbf{e}_h\|\|_2$ | EOC of $\|\|\mathbf{e}_h\|\|_2$ | $\|\|\mathbf{e}_h\|\|_2/\|\|c^{N_{\Delta t}}_{h_{ref}}\|\|_2$ | EOC of $\|\|\mathbf{e}_h\|\|_2/\|\|c^{N_{\Delta t}}_{h_{ref}}\|\|_2$ |
|---|---|---|---|---|
| $h_c = 1/80$ | 1.8821E-2 | – | 6.2165E-3 | – |
| $h_c/2$ | 8.2866E-3 | 1.1835 | 2.7867E-3 | 1.1576 |
| $h_c/4$ | 3.6609E-3 | 1.1786 | 1.2424E-3 | 1.1654 |
| $h_c/8$ | 1.5216E-3 | 1.2667 | 5.1876E-4 | 1.2600 |
| $h_c/16$ | 4.9920E-4 | 1.6079 | 1.7059E-4 | 1.6045 |

Table 4.5: Convergence of 2D M-C-N FEM solutions with $L_c = 16, \eta = 8, \epsilon_{AC} \approx 0.5029$.

| $h$ | $\|\|\mathbf{e}_h\|\|_2$ | EOC of $\|\|\mathbf{e}_h\|\|_2$ | $\|\|\mathbf{e}_h\|\|_2/\|\|c^{N_{\Delta t}}_{h_{ref}}\|\|_2$ | EOC of $\|\|\mathbf{e}_h\|\|_2/\|\|c^{N_{\Delta t}}_{h_{ref}}\|\|_2$ |
|---|---|---|---|---|
| $h_c = 1/80$ | 1.3646E-2 | – | 6.8867E-3 | – |
| $h_c/2$ | 6.2029E-3 | 1.1375 | 3.1862E-3 | 1.1120 |
| $h_c/4$ | 2.71195E-3 | 1.1936 | 1.4056E-3 | 1.1807 |
| $h_c/8$ | 1.1199E-3 | 1.2760 | 5.83060E-4 | 1.2695 |
| $h_c/16$ | 3.6754E-4 | 1.6074 | 1.9179E-4 | 1.6041 |

We also remark that, in general, the EOC values are appropriate only when computed using converged solutions with very good accuracy, say, less than 0.1% RMS error. In this sense, the EOC values corresponding to $\|\mathbf{e}_h\|_2 > 10^{-3}$ may not reflect the theoretical rate of convergence. However, for practical purposes approximate solutions with $\|\mathbf{e}_h\|_2 < 10^{-2}$ accuracy are sufficient.

Thus, the results in this section can help identify mesh-size values required to achieve such practical accuracy for sample realizations, which will allow us to efficiently carry out thousands of realizations when computing the statistical moments of the QoI. Such trade-offs are required, especially for 3D spatial models with complex structures that need to simulate thousands of realizations to understand the stochastic nature of the computed solutions.

Next, we consider the 3D model simulated results in Figure 4.5 and demonstrate the convergence of the M-C-N FEM approximate solutions in Table 4.6, Table 4.7, Table 4.8. As discussed previously, our aim in these 3D simulations is to identify the mesh-grid values that achieve at least 1% RMS error accuracy. As remarked above, the EOC values in Table 4.6, Table 4.7, Table 4.8 do not reflect the theoretical rate of convergence, since $\|\mathbf{e}_h\|_2 > 10^{-3}$.

Table 4.6: Convergence of 3D M-C-N FEM solutions with $L_c = 8, \eta = 3, \epsilon_{AC} \approx 0.8858$.

| $h$ | $\|\mathbf{e}_h\|_2$ | EOC of $\|\mathbf{e}_h\|_2$ | $\|\mathbf{e}_h\|_2/\|c^{N_{\Delta t}}_{h_{ref}}\|_2$ | EOC of $\|\mathbf{e}_h\|_2/\|c^{N_{\Delta t}}_{h_{ref}}\|_2$ |
|---|---|---|---|---|
| $h_c = 1/10$ | 0.2170 | – | 3.6643E-2 | – |
| $h_c/2$ | 7.3158E-2 | 1.5683 | 1.4333E-2 | 1.3542 |
| $h_c/4$ | 2.1739E-2 | 1.7507 | 4.6199E-3 | 1.6334 |
| $h_c/8$ | 5.9763E-3 | 1.8630 | 1.3255E-3 | 1.8013 |

Table 4.7: Convergence of 3D M-C-N FEM solutions with $L_c = 8, \eta = 5, \epsilon_{AC} \approx 0.7692$.

| $h$ | $\|\mathbf{e}_h\|_2$ | EOC of $\|\mathbf{e}_h\|_2$ | $\|\mathbf{e}_h\|_2/\|c^{N_{\Delta t}}_{h_{ref}}\|_2$ | EOC of $\|\mathbf{e}_h\|_2/\|c^{N_{\Delta t}}_{h_{ref}}\|_2$ |
|---|---|---|---|---|
| $h_c = 1/10$ | 0.2272 | – | 4.3230E-2 | – |
| $h_c/2$ | 6.7337E-2 | 1.7547 | 1.4826E-2 | 1.5439 |
| $h_c/4$ | 2.1770E-2 | 1.6291 | 5.1923E-3 | 1.5137 |
| $h_c/8$ | 6.2865E-3 | 1.7920 | 1.5638E-3 | 1.7314 |

Table 4.8: Convergence of 3D M-C-N FEM solutions with $L_c = 8, \eta = 8, \epsilon_{AC} \approx 0.8445$.

| $h$ | $\|\mathbf{e}_h\|_2$ | EOC of $\|\mathbf{e}_h\|_2$ | $\|\mathbf{e}_h\|_2/\|c^{N_{\Delta t}}_{h_{ref}}\|_2$ | EOC of $\|\mathbf{e}_h\|_2/\|c^{N_{\Delta t}}_{h_{ref}}\|_2$ |
|---|---|---|---|---|
| $h_c = 1/10$ | 0.5041 | – | 8.7992E-2 | – |
| $h_c/2$ | 0.3722 | 0.4376 | 7.5239E-2 | 0.2259 |
| $h_c/4$ | 9.3474E-2 | 1.9934 | 2.0476E-2 | 1.8775 |
| $h_c/8$ | 1.8868E-2 | 2.3086 | 4.3116E-3 | 2.2477 |

### 4.3 High-order and multilevel stochastic A-C computational model

In the stochastic A-C model and the QoI $Q(\omega)$ described in (4.1)–(4.4), we recall that randomness in the model is reflected in the notation by the dependence of the QoI on $\omega$. The main focus of this section is on developing high-order approximations to the expected value $\mathbb{E}[Q]$. For numerical experiments in this section, we use the stochastic initial states in (4.2) given by the representations in (4.19)–(4.22) . Thus, the number of uncertain parameters in the model, such as in (4.19) and (4.21), yields a random vector $z(\omega)$, which we assume in this section to be uniformly distributed. The number of entries in the vector defines the stochastic dimension $s$, which we take to be large ($> 10$) in comparison to the spatial dimension of the stochastic A-C system.
For our numerical experiments, with initial states in (4.19)–(4.22), the sample domain $\Omega$ is the product of $s$-rectangles, of the type $[a_v, b_v]$, and the probability measure in $\mathbb{E}[Q]$ is the product uniform measure. Each of these intervals can be bijectively mapped onto a reference interval of unit length, say $[0, 1]$, and hence, with $\widetilde{\Omega}$ being the $s$-dimensional unit hypercube $[0, 1]^s$, we may write $\mathbb{E}[Q]$ as the $s$-dimensional integral

$$\mathbb{E}\left[Q\right] = \int_{\widetilde{\Omega}} Q(z)\, \mathrm{d}\mathbb{P}_s(z). \tag{4.23}$$

The variance of the QoI has the representation

$$\mathbb{V}\left(Q\right) = \mathbb{E}\left[\left(Q - \mathbb{E}\left[Q\right]\right)^2\right]. \tag{4.24}$$

The $s$-dimensional integrals in (4.23)–(4.24) give, respectively, the measures of the mean and spread of the QoI $Q$. We recall that $Q(z(\omega)) = Q(c(\omega))$. That is, the QoI depends on the stochastic solution $c(\omega) : H^1(D) \to \mathbb{R}$ of the A-C system (4.1)-(4.3), at $t = T$. In practice, as described in the last section, we approximate $c(\omega)$ with $c_h(\omega) : V_h \to \mathbb{R}$, the M-C-N FEM approximation defined in (4.17) at $\ell = N_{\Delta t}$. We recall, from discussions in the previous section, that $\Delta t$ in turn depends on $h$ of the form $\Delta t = \mu h$ for some fixed $\mu < 1$. Hence, in this section, we suppress the explicit $\Delta t$ dependence notation in the solution and the QoI. In practice, we compute the M-C-N FEM approximation of the QoI, $Q_h(z(\omega)) := Q(c_h(\omega))$.

The main focus of this section is as follows: For a chosen accuracy $\epsilon$, (say, $\epsilon = 0.1\%$), develop an efficient computational stochastic A-C model, in conjunction with the M-C-N FEM approximation of the A-C system, to approximate the first-moment $\mathbb{E}[Q]$ that achieves the chosen accuracy with marked reduction in computational time (say, over 100-times) compared to the standard method for such approximations.

### 4.3.1 Monte Carlo approximations

Due to the large stochastic dimension of the A-C QoI, application of the standard (low- or high-order) tensor-product quadrature formula to directly discretize the $s$-dimensional integral in $\mathbb{E}[Q]$ becomes computationally prohibitive. To address this computational complexity, we utilize the methods described in Chapter 3. For completeness, we restate these methods in the following subsections and describe them in terms of the presented A-C system. The standard choice to discretize such a large dimensional integral is to use the Monte Carlo (MC) approximation [14]. That is, for a chosen sample size $N_{\text{MC}}$, the MC approximation to the mean of the QoI $Q_h$ is

$$\mathbb{E}_{\text{MC}}[Q_h; N_{\text{MC}}] = \frac{1}{N_{\text{MC}}} \sum_{j=1}^{N_{\text{MC}}} Q_h(z_j), \tag{4.25}$$

where $z_j$ for $j = 1, \ldots, N_{\text{MC}}$ are independent samples of the random vector $z(\omega)$. For each sample $z_j$, the QoI $Q_h(z_j)$ depends on the solution of the deterministic A-C model with sample parameter values of $z_j$ applied to the system.

As $N_{\text{MC}}$ increases, the MC approximation (4.25) converges to $\mathbb{E}[Q_h]$ in (4.23) w.r.t. the standard (variance) error. The standard error of the MC approximation is

$$S_{E,\text{MC}}[Q_h; N_{\text{MC}}] = \sqrt{\frac{1}{N_{\text{MC}}(N_{\text{MC}} - 1)} \sum_{j=1}^{N_{\text{MC}}} \left[ Q_h(z_j) - \mathbb{E}_{\text{MC}}[Q_h; N_{\text{MC}}] \right]^2}. \tag{4.26}$$

The order of convergence of the MC approximation in (4.25) to $\mathbb{E}(Q_h)$, measured in the standard error, is $O(N_{\text{MC}}^{-1/2})$ [13, 14], and the order constant of the approximation depends on the variance of the integrand $Q_h$. Thus, to achieve the chosen accuracy $\epsilon$ we require $N_{\text{MC}} \sim \epsilon^{-2}$. For example with $\epsilon = 10^{-3}$, we require $N_{\text{MC}} \sim 10^6$ and for each of these (about one million) samples we need to solve the nonlinear A-C system. In Figure 4.6 we demonstrate the need to take large $N_{\text{MC}}$ values

for the MC approximation of the 2-D stochastic A-C system (with $L_{AC} = 16$ in (4.19) and $h = h^{fine} = h_c/16$ used in Table 4.3–Table 4.5). All computed results in this section were obtained using a cluster of compute nodes with each node comprising of two twelve-core Intel Xeon E5-2680 2.50GHz processors.

The simulated MC approximation errors in Figure 4.6 demonstrate that a large number of samples are required to achieve even about 1% accuracy. In particular, the RHS of Figure 4.6 shows that despite using over half a million CPU-core minutes (that is about one year of CPU-core hours) for the stochastic 2D A-C system, the MC approximations cannot achieve the desired accuracy $\epsilon = 10^{-3}$. Thankfully, we are able to demonstrate the total of about one year of CPU-core hour results using multiple compute nodes, as the MC approximations are naturally parallel w.r.t. the samples. Thus, it is apparent that the standard MC approximation is not efficient or practical, especially for the 3D stochastic A-C system.

### 4.3.2 Quasi Monte Carlo approximations

For a chosen accuracy $\epsilon$, we are now interested in improving the relation of number of samples (on the $s$-dimensional unit hypercube $[0, 1]^s$) to the accuracy. That is, instead of the quadratic relation $N_{MC}^2 \sim \epsilon^{-1}$ (based on the low-order $N_{MC}^{-1/2}$ convergence of the MC quadrature rule), we are interested in an efficient relation of the form $N^\beta \sim \epsilon$, for some $\beta \leq 1$. That is, with $\alpha = 1/\beta$, we require approximations of the integral (4.23) with convergence order $\mathcal{O}(N^{-\alpha})$, for appropriate $\alpha \geq 1$. This can be achieved by using the quasi Monte Carlo (QMC) samples, with $\alpha$ depending on the regularity of the QoI. While the MC random samples depend only on the probability distribution of the integral (4.23), pure QMC samples are chosen deterministically (and then randomly shifted), and accuracy depends crucially on the regularity with respect to the uncertain parameters in the QoI. Randomly shifting the deterministic QMC samples facilitates unbiased errors estimates, and for appropriate comparisons with errors obtained using random MC samples. For details of obtaining deterministic QMC points and the associated randomization framework, see [16, 19] and references therein.

Because of the relatively higher rate of convergence, the practical advantage of the QMC approximations can be easily seen: For example, with $\alpha = 1$, to achieve $\epsilon = 10^{-3}$ accuracy for the QMC approximation we need to choose the number of QMC samples $N_{\text{QMC}} \approx 10^3$, a significant reduction compared to the required about one million samples in the MC case. Since the pure QMC samples are deterministic, for fair comparison with the MC accuracy and unbiased error estimates, it is standard to randomize the pure QMC points [16, 19].

For generation of QMC points and shifts, we follow the software framework in [19]. Since our high performance M-C-N FEM approach was written in modern versions of Fortran (90+), which facilitates easier use of the Intel MKL library PARDISO for setting up (on multiple compute nodes) and solving the FEM-discretized sparse algebraic systems, we developed our own Fortran90+ version code of the procedure described in [19] to generate the digital-shifted QMC points. Similar to our parallel realizations of the M-C-N FEM A-C MC stochastic algorithm, to achieve a large number of QMC based realizations we used the message passing interface (MPI) to parallelize the tasks, with respect to the QMC realizations.

In this chapter, we consider a digital net based construction of the pure (deterministic) QMC points. The well known Sobol′ points are digital nets of base 2, and in general the polynomial lattice rule provides an approach to construct digital nets [16, 19]. Briefly, with $N_q = 2^m$, for $x \in [0, 1)$ the construction starts with a finite base 2 representation with coefficients $x_1, x_2, \ldots$ that is limited to the number of bits in the representation to $m$. Hence, we can construct the pure QMC points $\boldsymbol{p}_i$ for $i = 1, \ldots, 2^m$ in the $s$-dimensional unit hypercube using the procedure described in [16, 19].

We choose the polynomial lattice QMC samples, especially the Sobol′ points as pure QMC points for numerical experiments, because these points can be subsequently interlaced to obtain higher-order approximations, with theoretical convergence rate $\alpha > 1$. (The other well known class of QMC points are the randomly-shifted lattice points [16, 19], that can achieve at most order one convergence and cannot be interlaced to obtain high order approximations for high-dimensional integrals with smooth integrands.)

Henceforth, we refer to $\alpha$ as an interlacing factor [16, 19], and first we consider the case $\alpha = 1$ (that does not involve interlacing of the polynomial lattice QMC points), leading to standard polynomial lattice QMC approximations with digital-shifted randomized QMC points $\boldsymbol{z}_{r,i}, r = 1, \ldots, N_s,\ i = 1, \ldots, N_q$. The digital-shifted QMC approximations $\mathbb{E}_{\text{QMC}}[Q_h; N_{QMC}]$ to $\mathbb{E}(Q_h)$ are

$$\mathbb{E}_{\text{QMC}}[Q_h; N_{QMC}] = \frac{1}{N_s N_q} \sum_{r=1}^{N_s} \sum_{i=1}^{N_q} Q_h(\boldsymbol{z}_{r,i}), \qquad N_{\text{QMC}} = N_s N_q. \tag{4.27}$$

Similar to (4.26), the standard error for the QMC approximation is

$$S_{E,\text{QMC}}[Q_h; N_{QMC}] = \sqrt{\frac{1}{N_s(N_s-1)} \sum_{r=1}^{N_s} \left( \frac{1}{N_q} \sum_{i=1}^{N_q} Q_h(\boldsymbol{z}_{r,i}) - \mathbb{E}_{\text{QMC}}[Q_h; N_{QMC}] \right)^2}. \tag{4.28}$$

Using the pure QMC Sobol′ points with digital shift $N_s = 32$, we simulated the 2-D stochastic A-C system with $L_{AC} = 16$ in (4.19) for several values of $N_{\text{QMC}}$ and with mesh-grid size $h = h^{fine} = h_c/16$ used in Table 4.3–Table 4.5. The standard error results in Figure 4.6 for the QMC simulation clearly demonstrate the $O(N_{\text{QMC}}^{-1+\delta})$ convergence of the QMC stochastic A-C model, for some very small $\delta$. Along with this, in order to achieve $\epsilon$ accuracy, we see that only $N_{\text{QMC}} \sim \epsilon^{-1}$ realizations of the A-C computational model are required, a marked advantage compared to the MC simulation. The CPU-core time of the simulations (RHS plot in Figure 4.6) demonstrates that compared to the MC approximations, the QMC approximations require less than 1/100-th of the CPU-core time compared to the MC case, for a similar accuracy achieved by both the approximations.

In general, it is known [16, 19] that the standard QMC approximations, with interlacing factor $\alpha = 1$, can achieve $1 - \delta$ rate of convergence for some $0 < \delta < 1/2$. The QMC-based 2D stochastic A-C system simulated results in Figure 4.6 show the best rate of convergence, with $\delta$ close to zero. However, for the computational stochastic A-C model in 3D (with $L_{AC} = 8$, $h = h^{fine} = h_c/16$ used in Table 4.6–Table 4.8), simulation results in Figure 4.7 using the standard QMC ($\alpha = 1$), and the digital shift $N_s = 32$, we see that the convergence rate is $1 - \delta$, with $\delta \approx 0.1$. This rate of convergence can be further improved, provided that the QoI $Q_h$ is sufficiently

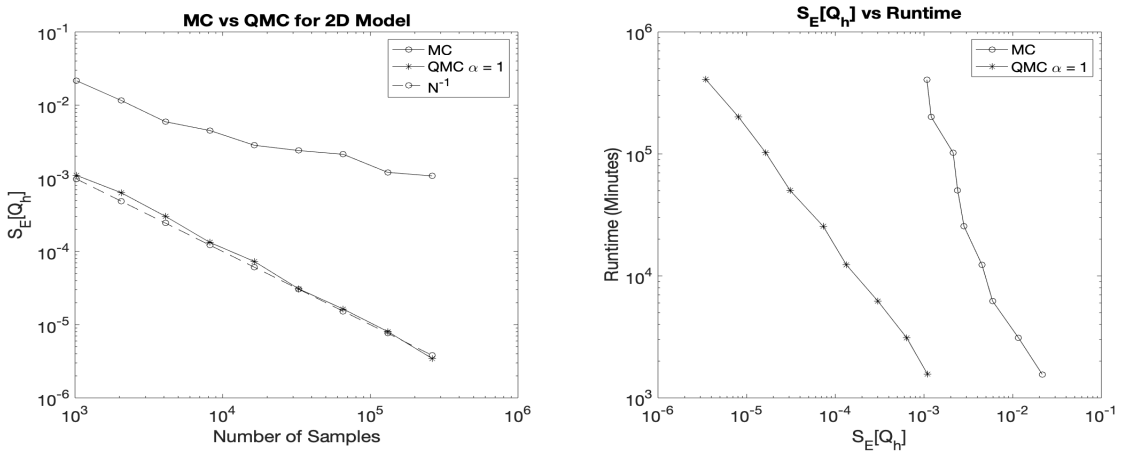



Figure 4.6: MC- and QMC-approximations based simulation results obtained using ($L_{AC} = 16$) stochastic 2D A-C model results to compute the mean of the M-C-N FEM QoI $Q_h(\omega)$, with $h = h^{fine}$.

smooth with respect to the uncertain parameters, using an interlaced version of the QMC (iQMC) to obtain high order convergence $\alpha - \delta$ for interlacing factors $\alpha = 2, 3, 4$. The $\alpha = 4$ restriction is because of argumentation of bits in the interlaced construction of the points and restriction of bit representations in double-precision.

The case $\alpha > 2$ requires substantial smoothness of the QoI [16, 19] which may not be possible for complex processes such as those modeled by the A-C system. The $\alpha > 2$ case convergence rate was demonstrated in the literature, see [19] and references therein, only for simple linear diffusion model problems with rapid decay in randomness (indirectly reducing the dominant stochastic dimension of the problem to less than five). Such an artificial reduction of stochastic dimensions are not possible for the stochastic model in this chapter. For our stochastic A-C nonlinear system experiments, the stochastic dimension is determined by a large number of parameters induced by the choice of $L_{AC}$ in the double/triple sum representations in (4.19) and (4.21).

For an interlacing factor $\alpha$, we use the briefly discussed polynomial lattice approach by taking $\alpha$ distinct numbers in $[0, 1)$ and interlace their bits to obtain a polynomial lattice rule in $\alpha\, s$ dimensions. For example, with $\alpha = 2$, we start with two coordinates $x, y \in [0, 1)$ with $x = (0.x_1x_2 \ldots x_m)_2$, $y = (0.y_1y_2 \ldots y_m)_2$ and we interlace their bits to obtain the new point

$w = (0.x_1y_1x_2y_2 \ldots x_my_m)_2$. Using representations of the type $w$, the interlaced pure QMC points $\boldsymbol{p}_i^{\alpha}$ are constructed. Subsequently, digital shifting of these points using $N_s$ shifts lead to the iQMC points $\boldsymbol{z}_{ri}^{\alpha},\ r = 1, \ldots, N_s,\ i = 1, \ldots, N_q$. The iQMC approximations and associated errors have representation as in (4.27)–(4.28) with $\boldsymbol{z}_{r,i}$ replaced with $\boldsymbol{z}_{r,i}^{\alpha}$. For our our numerical experiments, we use the interlaced form of the Sobol′ points.

Numerical results for the 3D stochastic A-C model in Figure 4.7 for the case $\alpha = 2$ demonstrate that the iQMC approximations provide improved accuracy and reduced CPU-core time compared to the standard QMC approximations. Furthermore, the observed rate convergence, in Figure 4.7, being less than 1.5 for iQMC approximations with $\alpha = 2$, indicates the restricted regularity of the stochastic A-C system QoI with respect to the uncertain parameters. Because of such a restricted regularity, the stochastic A-C model does not benefit by using the $\alpha = 3, 4$ interlacing factor in iQMC approximations.

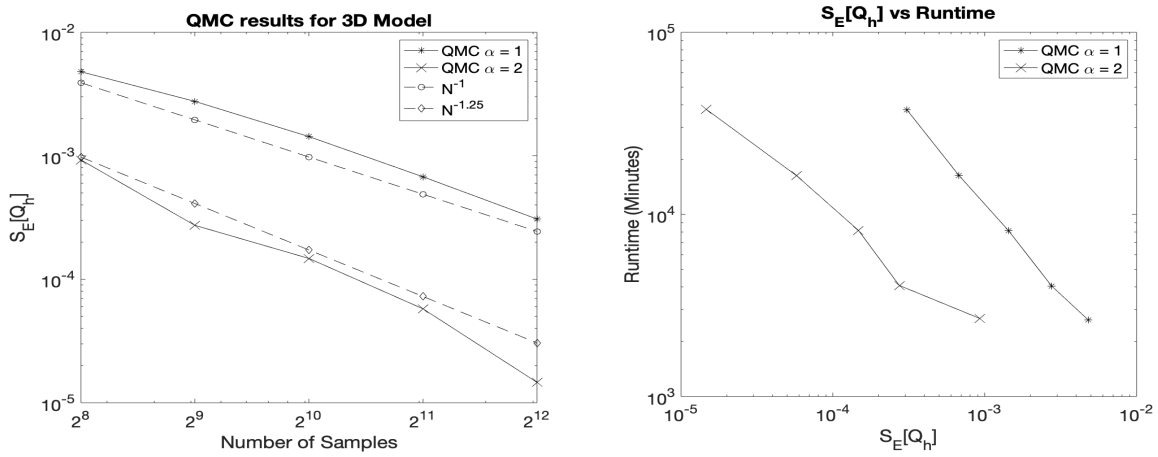



Figure 4.7: QMC simulation results with $\alpha = 1, 2$ for the ($L_{AC} = 8$) stochastic 3D A-C model.

### 4.3.3 Multilevel Quasi Monte Carlo Simulation

Although the QMC and iQMC approximation results observed above, with $\alpha = 1, 2$, substantially improve both the accuracy and CPU-core hours compared to the MC simulations (by 100-times), simulated results in Figure 4.6 and Figure 4.7, with fixed $h = h^{fine}$, demonstrate that to achieve $\epsilon = O(10^{-\mu})$ accuracy the CPU-core time is approximately $10^{\mu}$ minutes for the M-C-N A-C QMC

and iQMC 2D and 3D models. In particular, aiming for $\epsilon \approx 10^{-4}$ accuracy, if we focus on the high-order approximation results in Figure 4.6 for the 2D case (with $L_{AC} = 16, \alpha = 1$) and in Figure 4.7 for the 3D case (with $L_{AC} = 8, \alpha = 2$), a required simulation time of over 160 CPU-core hours (about 7 days) is needed to achieve 0.01% accuracy, which may not be practical especialy for larger scale A-C models in higher stochastic dimensions.

In this section, we are interested in further reducing these required QMC/iQMC CPU-core times (by 100-times), with the constraint that the rate of convergence of the QMC/iQMC approximations cannot be improved because of the limitation of the regularity of the stochastic solution of the A-C model with uncertainties. We achieve this by using an adaptive multi-level version of the QMC/iQMC approximations. The multiple application of the QMC/iQMC approximations will be adapted to the user specified accuracy, with the aim to reduce CPU-core time and still achieve the desired accuracy. The multi-level (ML) approach was first proposed in the context of the MC approximations, see [17] and references therein. In recent years, the ML approach using QMC/iQMC was applied to simple stationary linear diffusion models, see the survey article [19] and references therein. This chapter is the first numerical investigation of using QMC, iQMC and their ML counterparts (henceforth called the MLQMC) for nonlinear evolutionary stochastic A-C systems.

The QMC approximations (4.27) are based on using a fixed spatial mesh-grid $h$, typically a fine mesh-grid with $h = h^{fine}$ to ensure a good approximation for each realization of the $N_{\text{QMC}}$, leading to achieving the accuracy $\epsilon \sim N_{\text{QMC}}^{\alpha-\delta}$, where $\alpha$ is the interlacing factor and $0 < \delta < 1$ depends on the smoothness of the QoI w.r.t. the uncertain parameters [16, 19]. Similar to MC approximations, the order constant in the $O(N^{\alpha-\delta})$ convergence of QMC/iQMC approximations depends on the variance of the integrand. Thus the precise numerical accuracy of $\epsilon$ depends crucially on the integrand variance occurring in the QMC/iQMC approximations of the integral. The latter knowledge is a key factor in devising the MLQMC scheme, in conjunction with discretization parameters in the FEM and QMC approximations and appropriate trade-offs in the spatial and stochastic degrees of freedom.

For the MLQMC approximations, we first choose a desired accuracy $\epsilon$ and the adaptive choice of the number of QMC points will depend on a list of mesh-sizes $h_\ell, \ell = 0, 2, 3, \ldots$, and the accuracy. The list of mesh sizes, such as $h_\ell = h_0/2^\ell$, are chosen based on the accuracy data obtained through prior investigations for some sample choice of uncertain parameters in the model. For example, we carried out such an investigation in Section 4.2.3 for the A-C 2D and 3D models with $h_0 = h_c$ and for $\ell = 0, \ldots 4$, and for the fixed choice of parameters in the models. In particular, the tabulated values in Table 4.3–Table 4.5, and Table 4.6–Table 4.8 provide data for the expected accuracy of the M-C-N FEM solution $c_{h_\ell}$ of the A-C system at final time $t_{N_{\Delta t}}$, respectively, in 2D and 3D. Hence, the accuracy of the QoI $Q_{h_\ell}$, obtained by integrating the FEM solution $c_{h_\ell}$ in the 2D/3D spatial variables, is expected to be better than the associated error in the tables. Further, for $\ell = 1, \ldots, 4$, since

$$\left|Q_{h_\ell} - Q_{h_{\ell-1}}\right| \lessapprox \|c_{h_\ell} - c_{h_{\ell-1}}\| \to 0, \qquad \text{as} \quad h_\ell \to 0,$$

the behavior of $Q_{h_\ell}$ and $Q_{h_{\ell-1}}$ are similar, and hence we expect the variance of $[Q_{h_\ell} - Q_{h_{\ell-1}}]$ to get smaller as $\ell$ increases.

Motivated by the above observations, instead of the fixed mesh-grid size $h$ for the MC/iQMC approximation, the MLQMC approximation uses multiple coarser mesh-grids, with $h = h_\ell$ *$(\ell = 0, \ldots, L)$, which is influenced by the total $L$-level telescoping integral identity:

$$\mathbb{E}\left[Q_{h_L}\right] = \mathbb{E}\left[Q_{h_0}\right] + \sum_{\ell=1}^{L} \mathbb{E}\left[Q_{h_\ell} - Q_{h_{\ell-1}}\right]. \tag{4.29}$$

The $L$th level is adaptively chosen in the MLQMC algorithm. This is done by ensuring that the approximation $\mathbb{E}\left[Q_{h_L}\right]$ achieves the desired accuracy $\epsilon$ and also that the cost of using the QMC/iQMC procedure for using QMC/iQMC approximations in $L$ levels does not exceed the cost of using the single-level QMC/iQMC for the problem, to achieve the desired accuracy with a fine mesh $h = h^{fine}$. There are several technical details associated with this adaptive procedure that involves tuning of various cost parameters. Such details can be found in [17], and we use the same adaptive approach for the MLQMC algorithm. In theory, the existence of the final level $L$

(to achieve the desired accuracy $\epsilon$) is ensured by [17, Theorem 1]. In practice, if one utilizes the adaptive procedure established in [17] for the MLQMC algorithm, as demonstrated for various applications with MLMC, it is often sufficient to initially set $L = 2$ with one sample and appropriate shifts, say 32, at each level. Similar to other applications, for the model in this chapter we follow the established procedure in [17], and we also observed that the standard choice works efficiently for our MLQMC simulations. We recall that the accuracy is based on the standard variance error, which is given in the discrete form in (4.28). Since the variance of the integrands in the second term of (4.29) is expected to be sufficiently small, in practice only a few levels are required in the MLQMC algorithm.

Unlike the QMC/iQMC procedure that starts with the fixed choices of $N_{\text{QMC}}$ and $h$, the MLQMC procedure starts with a desired accuracy $\epsilon$ and a list of spatial mesh-grid parameters $h_\ell, \ell = 0, 1, \ldots, M$. At the initial level, the number $N_{\text{QMC}}$ is chosen (typically based on the above study) but uses a relatively coarser mesh $h = h_0$. Subsequently, for the level $\ell \geq 1$, the procedure adaptively chooses (and reduces) the smaller number of $N^\ell_{\text{MLQMC}}$ solves of the model, using a finer mesh with $h = h_\ell$, and the procedure stops with some level $L \leq M$ after achieving the desired accuracy.

Similar to (4.27), for the level $\ell = 0, 1, \ldots L$, we define the QMC approximation for a function $\Phi$ (that depends on the FEM mesh-grid parameter and stochastic variables) as

$$\mathbb{E}_{\text{MLQMC}}\left[\Phi; N^\ell_{\text{MLQMC}}\right] = \frac{1}{N^\ell_s N^\ell_q} \sum_{r=1}^{N^\ell_s} \sum_{i=1}^{N^\ell_q} \Phi^{(\ell)}(\boldsymbol{z}^{(\ell)}_{r,i}), \qquad N^\ell_{\text{MLQMC}} = N^\ell_s N^\ell_q, \tag{4.30}$$

where $\boldsymbol{z}^{(\ell)}_{r,i}$ are QMC/iQMC digital-shifted points chosen for the integrand $\Phi^{(\ell)}$ that depends on the mesh-size $h_\ell$ and, for $\ell \geq 1$, also depends on $h_{\ell-1}$ . With the stochastic A-C model QoI $Q_h(\boldsymbol{z}(\boldsymbol{\omega})) = Q(c_h(\boldsymbol{\omega}))$, using the ansatz (4.29) and (4.30), the MLQMC approximation to $\mathbb{E}\left[Q_{h_L}\right]$ is defined as

$$\mathbb{E}^{(L)}_{\text{MLQMC}}\left[Q_{h_L}\right] := \mathbb{E}_{\text{MLQMC}}\left[Q_{h_0}; N^0_{\text{MLQMC}}\right] + \sum_{\ell=1}^{L} \mathbb{E}_{\text{MLQMC}}\left[Q_{h_\ell} - Q_{h_{\ell-1}}; N^\ell_{\text{MLQMC}}\right]. \tag{4.31}$$

The associated MLQMC variance and standard errors are defined as

$$\mathbb{V}\left[Q_{h_L}\right] = \frac{1}{N^0_{\text{MLQMC}}}\mathbb{V}[Q_{h_0}] + \sum_{\ell=1}^{L}\frac{1}{N^\ell_{\text{MLQMC}}}\mathbb{V}\left[Q_{h_\ell} - Q_{h_{\ell-1}}\right], \qquad S_{E,ML}[Q_{h_L}] = \sqrt{\mathbb{V}\left[Q_{h_L}\right]}. \quad (4.32)$$

In our implementation, for the stochastic A-C model M-C-N FEM MLQMC algorithm, we fix the shift values $N^\ell_s = 32$ for all levels, and choose $h_0 = h_c/2$, where $h_c$ for the 2D A-C model is as defined in Table 4.3–Table 4.5, and for the 3D case as defined in Table 4.6–Table 4.8. Thus, the coarse to fine mesh-grid choices $h_\ell = h_c/2^{\ell+1},\ \ell = 0, 1, 2, \ldots$ ensure increasing accuracy for the M-C-N FEM solutions for each realization, as the level increases. As we observe from simulated results in Figure 4.8–Figure 4.11, for various choices of the accuracy values $\epsilon$, the automatically chosen $L = 2$ was sufficient to achieve the desired accuracy. Results in Figure 4.9 and Figure 4.11 demonstrate that the MLQMC algorithm adaptively chose the number of QMC/iQMC samples that substantially decreases as the level of the MLQMC approximation increases.

Figure 4.8 and Figure 4.10 demonstrate substantial computational efficiency of the MLQMC algorithm compared to even the efficient QMC approximations (that were already demonstrated in Figure 4.6 to achieve over 100-times efficiency compared to the standard MC approximations), respectively, for the 2D and 3D simulated stochastic models with high-order accuracy. For example, as demonstrated in Figure 4.8, the 2D stochastic A-C model MLQMC approximations (with $\alpha = 1$) achieve about $10^{-5}$ accuracy using a total CPU-core time that is 100-times less than the CPU-core time required using the QMC approximations. Similarly, we observe over 100-times similar speed-up of the stochastic M-C-N FEM MLQMC algorithm for the 3D stochastic A-C model with $\alpha = 2$ compared to the same achieved using the iQMC algorithm to achieve less than $10^{-3}$ accuracy. In summary, our M-C-N FEM stochastic MLQMC A-C model is very efficient, achieving both high accuracy and marked reduction in computational time when compared to standard MC-type approximations.

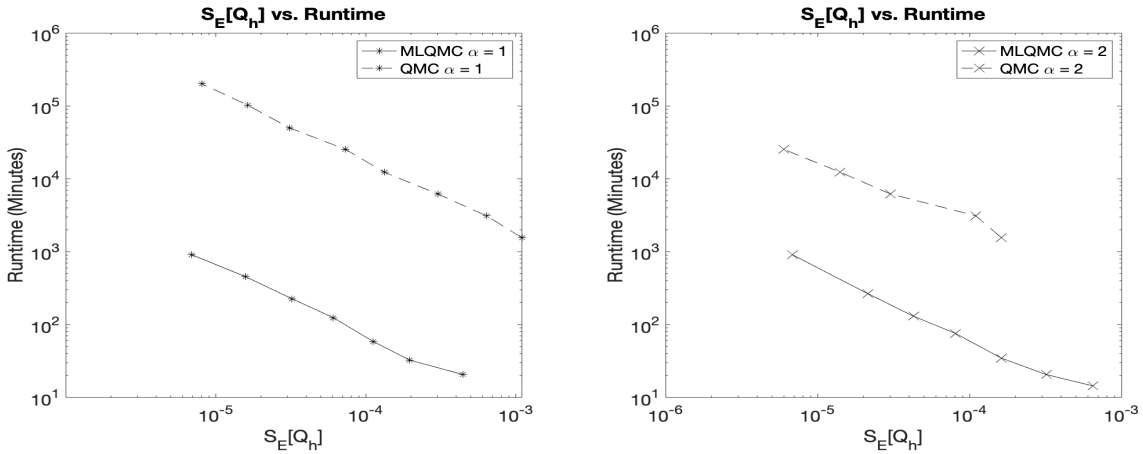



Figure 4.8: Stochastic 2D AC model results using QMC/iQMC and MLQMC Algorithms.

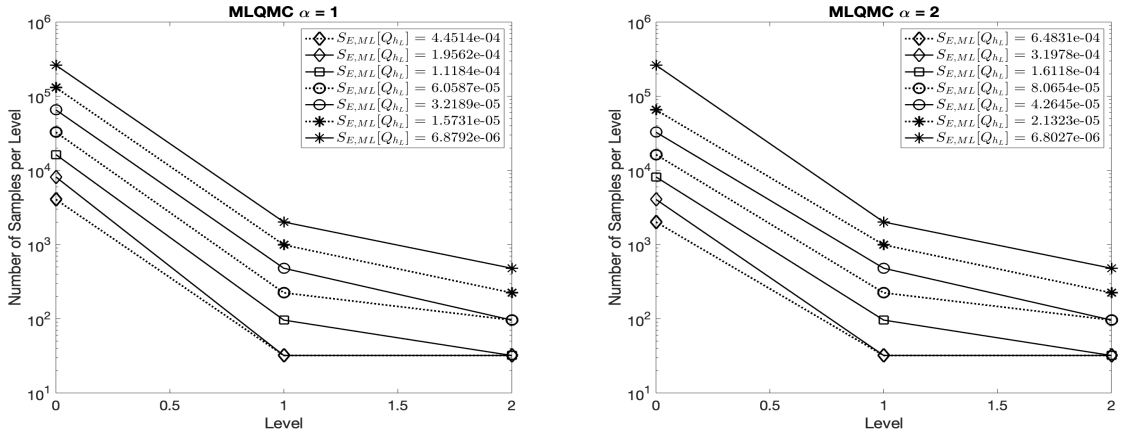



Figure 4.9: Stochastic 2D AC model with number of samples per level for MLQMC simulation. These plots are obtained by MLQMC simulations with $\epsilon = 1/\sqrt{2^k}$, for $k = 10, \ldots, 16$, and the corresponding descending $S_{E,ML}[Q_{h_L}]$ values are shown in the legend.

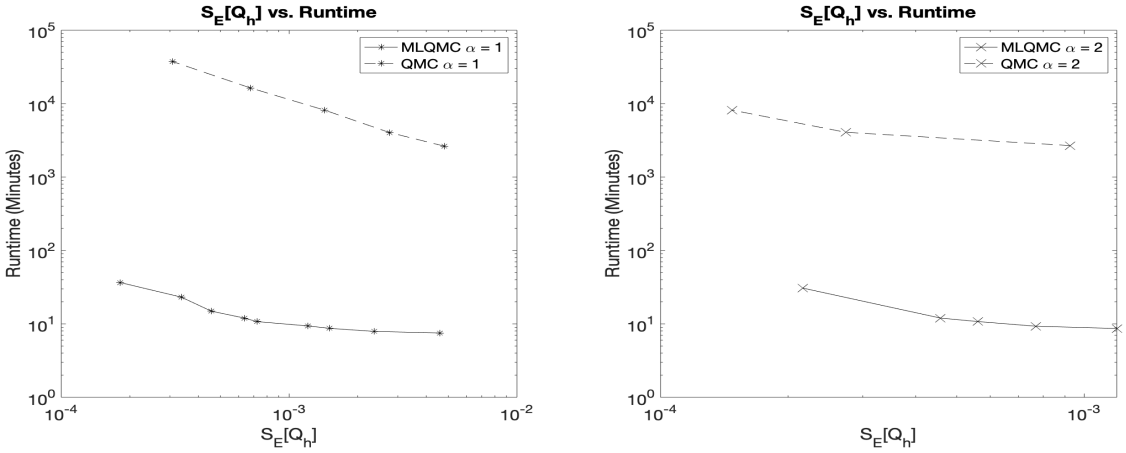



Figure 4.10: Stochastic 3D AC model results using QMC/iQMC and MLQMC Simulation.

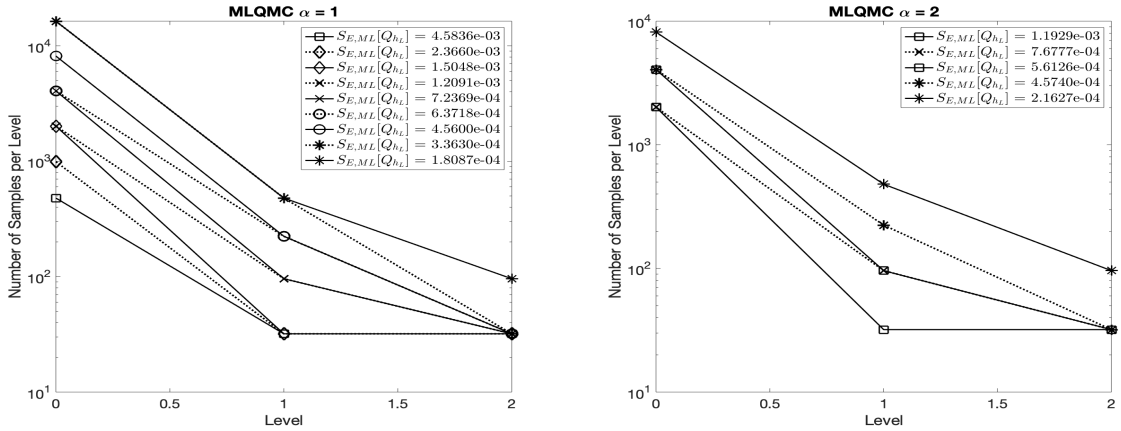



Figure 4.11: Stochastic 3D AC model with number of samples per level for MLQMC simulation. These plots are obtained by MLQMC simulations with $\epsilon = 1/\sqrt{2^k}$. For the left plot we chose $k = 13, 15, 17, 18, 19, 20, 21, 22, 23$, and the corresponding descending $S_{E,ML}[Q_{h_L}]$ values are shown in the legend of the plot. For the right plot, we chose $k = 18, 19, 20, 21, 22$.

# CHAPTER 5

# A MOVING DOMAIN STOCHASTIC MODEL

In this chapter, we describe and demonstrate a computational model, using a moving-mesh finite element method (FEM) and the multilevel Monte Carlo (MLMC) algorithm developed in Chapter 3, to simulate statistical moments of a moving domain induced stochastic quantum transients. It should be noted that a majority of the material in this section was obtained from our article [21] (for copyright documentation see Appendix F). For a review on the theory and applications of transients, we refer to [48] and 288 references therein. The earliest of these references are due to Moshinsky. A closely related deterministic stationary-domain version of the model considered in this chapter was proposed in the seminal 1952 paper by Moshinsky [49], entitled *Diffraction in Time* (DiT). Even several decades after introduction of the DiT model, there continues to be substantial interest in understanding more practical variants and solvability of the model. For a recent effort on an exactly solvable DiT model, see [50] and references therein. We refer to the above references for large literature on the DiT models, including reasons for the term DiT attributed to a quantum matter transient waves phenomenon related to that in optics.
Transients are processes induced by sudden interaction of particles impinging on a shutter or changes due to initial states or boundary conditions. As described in the review article [48, Page 2], the transient models investigated mainly in the (Physics) literature are those for which analytical solutions can be constructed using classical functions (such as Fresnel integrals). Such analytic constructions impose restrictions on the model such as requiring stationary shutters, deterministic initial conditions, or no boundary conditions (to apply Fourier/Laplace transforms). These include the seminal DiT model by Moshinsky [49] and the recently investigated exactly solvable variant DiT model [50] and references therein. The model in [50] admits a time-dependent aperture/shutter function $\chi(t)$ and represents the resulting unknown particle wave function $\Psi(x,t)$ as an improper-spatial-integral on $(-\infty,\infty)$, with its integrand depending on a

deterministic initial state and a fundamental solution Green's kernel, satisfying the time-dependent one-space dimension Schrödinger equation. In this simplified case, the improper integral can be evaluated only for certain special forms of $\chi(t)$ and the approach in [50] also assumes that the Schrödinger Green's kernel can be evaluated exactly. In general, the latter is not possible when the one-dimensional Schrödinger equation is considered in conjunction with time-dependent spatial-boundedness ($[\beta, \gamma(t)]$, see Figure 5.1) of the moving shutter (as the beam of the particles impinge on the shutter), associated boundary conditions, and a non-deterministic continuous initial state.

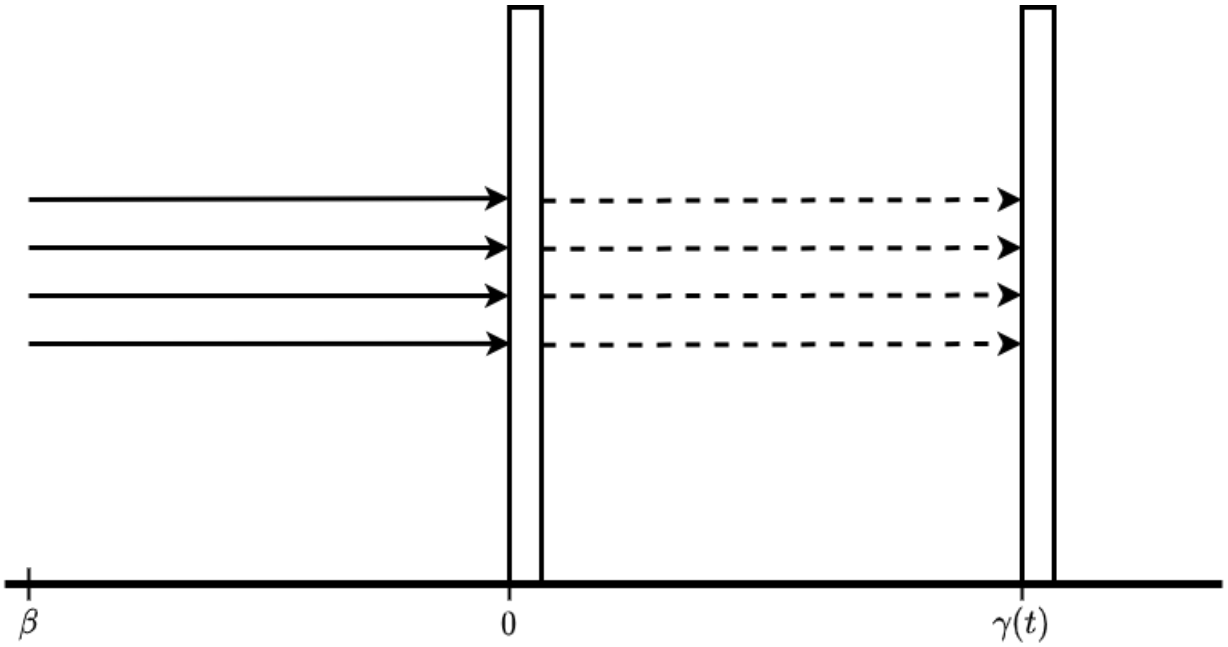



Figure 5.1: An illustration of the DiT model governed by a moving shutter with speed $\gamma'(t)$ and a continuous initial state. The shutter on the left represents the closed shutter's position at $t = 0$ and the right shutter is a closed shutter at the position determined by $\gamma(t)$.

The deterministic case, with a constant $\gamma(t)$ (and a discontinuous initial state) of the stochastic model considered in this chapter reduces to the stationary shutter DiT model introduced by Moshinsky [49], which models the deterministic quantum dynamics of a suddenly released beam of independent particles of mass $m$. In this special stationary case, the time evolution of a quantum particle with wave function $\psi(x, t)$ satisfying the Schrödinger equation can be written as

the Moshinsky function, $M(x,k,t)$ [48, 49], and the concept of DiT can be tied to the quantum density profile $|\psi(x,t)|^2$ [48, 49]. Here, $k$ is the wavenumber and $\lambda = 2\pi/k$ is the wavelength, so that $\hbar/\lambda$ and $\hbar/(m\lambda)$ (with $\hbar$ being the Planck's constant) are respectively the nominal momentum and velocity, of a quasi-monochromatic atomic beam impinging on a stationary shutter. These constants occur together as a coefficient of the diffusion part of the Schrödinger equation [49]. In this chapter we use the notation $\overline{\alpha}(k)$ to denote the coefficient.

### 5.1 A moving domain stochastic DiT model

The main focus of this chapter is developing and demonstrating an algorithm to compute statistical moments of a quantity of interest (QoI), determined by the stochastic quantum density profile $|\psi(x,t;\boldsymbol{\omega})|^2$, which are crucial for quantifying uncertainties in the QoI. The stochastic quantum density profile integrand is determined by the evolution of the unknown wave function $\psi$ satisfying the following stochastic Schrödinger model on a moving domain $D(t) = (\beta, \gamma(t))$, where $\gamma(t)$ represents the position of the moving shutter at time $t$:

$$\mathrm{i}\frac{\partial}{\partial t}\psi(x,t;\widetilde{\boldsymbol{\omega}}) + \bar{\alpha}(k)\frac{\partial^2}{\partial x^2}\psi(x,t;\widetilde{\boldsymbol{\omega}}) = 0, \quad x \in D(t;\omega), \qquad \widetilde{\boldsymbol{\omega}} \in \widetilde{\boldsymbol{\Omega}}, \quad 0 < t \leq T, \tag{5.1}$$

$$\psi(x,0;\widetilde{\boldsymbol{\omega}}) = \mu(x)g_k(x) + R(x;\boldsymbol{\omega}), \qquad x \in \overline{D(0;\omega)}, \quad \widetilde{\boldsymbol{\omega}} = (\boldsymbol{\omega},\boldsymbol{\omega}), \quad \boldsymbol{\omega} \in \boldsymbol{\Omega}. \tag{5.2}$$

In the above model, for convenience, we assume homogeneous Dirichlet boundary conditions at $x = 0$ and at $x = \gamma(t;\omega)$, for all $t \geq 0$.

In this chapter we take the non-constant speed of the shutter to be non-deterministic, of the form $\gamma'(t) = \omega f(t)$, where $\omega$ is from a one-dimensional probability space $(\Omega, \mathcal{F}, \mathcal{P})$ and $f : [0,T] \to \mathbb{R}$ is a real-valued function. We denote the resulting moving non-deterministic domain as $D(t;\omega) := (\beta, \gamma(t;\omega))$, for $t \in [0,T]$, with $T$ being the final simulation time of interest. We note that for the domain to be well defined, $\gamma(t;\omega) \geq \beta$ for all $t \in [0,T]$ and $\omega \in \Omega$. In practice at time $t = 0$, depending on the choice of $f(t)$, the initial domain $D(0;\omega)$ need not depend on $\omega$.

In addition, we augment the complexity of the variant stochastic version of the Moshinsky's problem by taking the initial state to be a random field, depending on a random vector $\boldsymbol{\omega}$. Here, $\boldsymbol{\omega}$ is an element of $\mathbf{\Omega} \subseteq \mathbb{R}^d$, and $(\mathbf{\Omega}, \mathcal{F}, \mathbb{P})$ is the associated probability space with Euclidian sample space $\mathbf{\Omega}$ and Borel $\sigma$-algebra $\mathcal{F}$. The probabilities of the events in $\mathcal{F}$ are given by the probability measure $\mathbb{P}$. For notational convenience, in (5.1)–(5.2) and throughout the chapter, we use $\widetilde{\boldsymbol{\omega}} = (\boldsymbol{\omega}, \omega)$ to denote a random vector in $\widetilde{\mathbf{\Omega}} = \mathbf{\Omega} \times \Omega \subset \mathbb{R}^{d+1}$. The associated product probability measure in $\widetilde{\mathbf{\Omega}}$ is $\widetilde{\mathbb{P}} = \mathbb{P} \times \mathcal{P}$.

Because of the complexity of the stochastic model on the moving domain, in this chapter, we take the initial random state of the model in (5.2) to be continuous in the spatial variable. Developing an efficient moving-mesh and stochastic realization algorithm of this continuous initial state model is a challenging problem. In a future article, we plan to investigate the extension of the model to allow for random discontinuous initial states on moving domains. Such discontinuous data based stochastic models are challenging even for fixed spatial domain cases.

In this setting, with a large $d$-stochastic dimension, the released initial state of the beam of particles is a square-integrable random process and the covariance operator of the random process induces the associated Karhunen-Loéve (KL) series expansion of the field [14]. In practice, assuming decay in the Fourier coefficients of the field, the initial state may be approximated by a truncated KL expansion with $d$-terms. This leads to our $d + 1$-dimensional stochastic model induced by the perturbation of the wavenumber dependent deterministic initial state $\mu(x)g_k(x),\ x \in [\beta, \gamma(0)]$, by a random field $R(x; \boldsymbol{\omega})$. In the stationary shutter deterministic Moshinsky model on $(-\infty, \infty)$ from [49], $g_k(x) = \exp(\mathrm{i}kx)$ (a plane wave) and $\mu(x) = 0$ for $x \in [0, \infty)$. In our simulations, we also choose $g_k$ to be the plane wave and choose $\mu$ such that the initial state satisfies the boundary conditions at $x = \beta$ and at $x = \gamma(0)$.

To illustrate the decay assumption in the random field, say with mean zero, consider for example with $x \in [0, 1]$, the eigenfunctions of the second differential operator $-d^2/dx^2$ (with domain restricted to functions satisfying homogeneous Dirichlet boundary conditions at $x = 0$ and at $x = 1$) given by $\sin(j\pi x)$, and associated eigenvalues $j^2\pi^2$, for $j = 1, 2, \ldots$ . The integral

covariance operator inducing the random field, $C = [-d^2/dx^2]^{-2}$, has the same eigenfunctions, but the eigenvalues of $C$ decay with $O(j^{-4})$ [100]. Hence, depending on the appropriate accuracy required in the representation of the random field, we can choose a parameter $d$ to truncate the KL expansion and represent the random field using $d$ random variables $\omega_j$, for $j = 1, \ldots, d$ (say uniformly distributed on $[-0.5, 0.5]$) as

$$R(x;\boldsymbol{\omega}) = \sum_{j=1}^{d} \frac{\omega_j}{j^2} \sin(j\pi x), \qquad x \in [0,1], \quad \boldsymbol{\omega} = (\omega_1, \cdots, \omega_d) \in \boldsymbol{\Omega}. \tag{5.3}$$

For example with $d = 64$ (which we use in our numerical experiments), the truncated terms in the series KL expansion are less than $O(10^{-4})$. In our model, since the domain at time $t = 0$ is $[\beta, \gamma(0)]$, with $\beta < 0$, the arguments of the $\sin(j\cdot)$ functions need to be appropriately scaled using change of variables, to satisfy the homogeneous Dirichlet conditions at $x = \beta$ and at $x = \gamma(0)$. That is, we use the representation

$$R(x;\boldsymbol{\omega}) = \sum_{j=1}^{d} \frac{\omega_j}{j^2} \sin(j\pi(x-\beta)/(\gamma(0)-\beta)), \quad x \in [\beta, \gamma(0)], \quad \boldsymbol{\omega} = (\omega_1, \cdots, \omega_d) \in \boldsymbol{\Omega}. \tag{5.4}$$

In this chapter, we are interested in efficiently computing statistical moments of the QoI, $Q(\widetilde{\omega})$ at the final simulation time $T$, where

$$Q(\widetilde{\omega}) = \int_{D(T;\omega)} |\psi(x, T; \widetilde{\omega})|^2 dx, \qquad \widetilde{\omega} \in \widetilde{\boldsymbol{\Omega}}. \tag{5.5}$$

The QoI provided in (5.5) can be physically interpreted as the probability that a particle is found in $D(T;\omega)$. For computation of an approximation to the first statistical moment (expected value) of the QoI in (5.5), we need to approximate the $d + 1$ dimensional integral

$$\mathbb{E}[Q] = \int_{\widetilde{\boldsymbol{\Omega}}} Q(\widetilde{\omega})\, d\widetilde{\mathbb{P}}(\widetilde{\omega}). \tag{5.6}$$

An approximation of the high-dimensional integral in (5.6) requires a large number, say $N_{MC}$, of sampling/realization points in the $(d + 1)$-dimensions and use of non-standard quadratures, such as the low-order accurate Monte Carlo (MC) [13] approximation. The advantage of the MC approximation is in its simplicity of choosing the realization points in the probability space

$(\widetilde{\mathbf{\Omega}}, \widetilde{\mathcal{F}}, \widetilde{\mathbb{P}})$ and that the rate of convergence of the MC approximation of the integral in (5.6) is independent of the high stochastic dimension. However, because the accuracy of the MC approximation is $O(N_{MC}^{-1/2})$, in general a large number of samples, $N_{MC}$, are required. For *each* such fixed sample, say $\widetilde{\omega}^*$, we need to simulate the associated deterministic moving domain DiT model to compute $Q(\widetilde{\omega}^*)$ in (5.5) with integrand $|\psi^*(x,T)|^2$, where $\psi^*(x,t) = \psi(x,t;\widetilde{\omega}^*)$, $x \in D^*(t) = D(t;\omega^*)$, and $t \in [0,T]$.

In the next section, we describe a moving-mesh FEM for efficiently solving the deterministic moving domain DiT model to compute $\psi^*(x,t)$ and demonstrate the second-order accuracy of the algorithm. The Crank-Nicolson algorithm is unconditionally stable, requiring no CFL condition on the time discretion step, and in our simulation we take the spatial and time discretization steps to be of the same size. Our simulation of the deterministic FEM model suggests that a fine time-dependent spatial-mesh, say $h_{fine}(t)$, of the moving domain is required, in conjunction with the matching fine time-mesh, to obtain better than $O(10^{-3})$ accuracy. In particular, we aim for this accuracy to be better than that which is used for the approximate representation of the random field $R$, using the truncated KL expansion. Hence, computing approximate expected values of the QoI using the standard MC algorithm requires a large number of realizations of the deterministic model, with a fine space-time mesh for each realization.

In the next section, we tabulate deterministic moving-mesh FEM simulation results for various mesh sizes, with (approximate second-order) increase in accuracy as the mesh becomes finer. These results also provide a framework to apply a multilevel version of MC that facilitates an appropriate trade-off between a large number of MC samples to a fewer number of MC samples in conjunction, respectively, with coarse and fine mesh, depending on the level. The initial level of MLMC [17] may be considered as standard MC, but with coarser space-time mesh and the later levels may be regarded as MC with fewer samples and a finer mesh as stated in Chapter 3. In Section 5.3, we describe and apply the standard MC and MLMC algorithms to simulate the expected value of the QoI and demonstrate that the computational cost of the MLMC algorithm is substantially better than the MC algorithm, for the stochastic moving shutter DiT model.

Throughout this chapter, even for a fixed time $t$, we use the operator notation $\frac{\partial}{\partial t}$ acting on a time-variable dependent function to denote the first partial derivative of the function w.r.t. the second variable. Thus, for example, $\frac{\partial}{\partial t}\psi^*(x,t)$ means that we first take the partial derivative of the function $\psi^*$ w.r.t. the second variable and then evaluate the resulting function at the fixed time $t$.

### 5.2 A deterministic moving-mesh computational DiT model

High-order finite difference and element methods for the Schrödinger equation have been applied in many areas over several decades, see for example [101–103]. However, these developed methods were mainly restricted to the Schrödinger equation on fixed domains. For recent work on deterministic moving-mesh models, see [104] and references therein, including the 2009 survey article [105] (with a large number of references) on the development of moving-mesh methods over the last four decades. To our knowledge, this chapter is the first work on applying moving-mesh techniques for simulation of the deterministic and stochastic DiT models.
In this section, for a fixed realization $\widetilde{\omega}^* = (\omega^*, \omega^*)$, we describe an algorithm to efficiently approximate $\psi^*(x,t)$, satisfying the deterministic moving domain Schrödinger model, induced by a monochromatic beam of particles moving parallel to the $x$-axis impinging on a shutter that moves perpendicular to the beam in its frame, with a deterministic speed $\gamma'(t) = \omega^* f(t)$:

$$\begin{aligned} \mathrm{i}\frac{\partial}{\partial t}\psi^*(x,t) + \bar{\alpha}(k)\frac{\partial^2}{\partial x^2}\psi^*(x,t) &= 0, \qquad x \in D^*(t) = (\beta, \gamma(t)), 0 < t \le T, && (5.7)\\ \psi^*(x,0) &= \mu(x)g_k(x) + R(x;\boldsymbol{\omega}^*), \quad x \in \overline{D^*(0)}, && (5.8)\\ \psi^*(\beta,t) &= 0 = \psi^*(\gamma(t),t), \qquad 0 \le t \le T. && (5.9) \end{aligned}$$

For each fixed $t \in [0,T]$ and $v_1, v_2 \in L^2(D^*(t))$, we use the notation

$$\langle v_1, v_2\rangle_t = \int_{D^*(t)} v_1(x)\overline{v_2(x)}\, dx,$$

to denote the standard inner product of square integrable functions defined on $D^*(t)$. Using the homogeneous Dirichlet boundary conditions, we now derive the weak form of the deterministic

model (5.7)–(5.9). Letting $v \in H_0^1(D^*(t))$ and multiplying (5.7) by v, we obtain the following:

$$\int_{D^*(t)} \left( i\frac{\partial \psi^*}{\partial t}(x,t) + \bar{\alpha}(k)\frac{\partial^2 \psi^*}{\partial x^2}(x,t) \right) v(x,t)\, dx = 0$$

$$\iff i\int_{D^*(t)} \frac{\partial \psi^*}{\partial t}(x,t)\, v(x,t)\, dx + \bar{\alpha}(k)\int_{D^*(t)} \frac{\partial^2 \psi^*}{\partial x^2}(x,t)\, v(x,t)\, dx = 0$$

$$\iff i\int_{D^*(t)} \frac{\partial \psi^*}{\partial t}(x,t)\, v(x,t)\, dx + \bar{\alpha}(k)\int_{D^*(t)} \frac{\partial}{\partial x}\left( \frac{\partial \psi^*}{\partial x}(x,t) \right) v(x,t)\, dx = 0.$$

Using integration by parts, we obtain:

$$i\int_{D^*(t)} \frac{\partial \psi^*}{\partial t}(x,t)\, v(x,t)\, dx + \bar{\alpha}(k)\left[ \left[ \frac{\partial \psi^*}{\partial x}(x,t)\, v(x,t) \right]\Big|_{\beta}^{\gamma(t)} - \int_{\beta}^{\gamma(t)} \frac{\partial \psi^*}{\partial x}(x,t)\, \frac{\partial v}{\partial x}(x,t)\, dx \right] = 0$$

$$\iff i\int_{D^*(t)} \frac{\partial \psi^*}{\partial t}(x,t)\, v(x,t)\, dx + \bar{\alpha}(k)\left[ \frac{\partial \psi^*}{\partial x}(\gamma(t),t)\, v(\gamma(t),t) - \frac{\partial \psi^*}{\partial x}(\beta,t)\, v(\beta,t) \right.$$

$$\left. - \int_{\beta}^{\gamma(t)} \frac{\partial \psi^*}{\partial x}(x,t)\, \frac{\partial v}{\partial x}(x,t)\, dx \right] = 0$$

$v = 0$ on $\partial D^*(t)$ then gives:

$$i\int_{D^*(t)} \frac{\partial \psi^*}{\partial t}(x,t)\, v(x,t)\, dx - \bar{\alpha}(k)\int_{D^*(t)} \frac{\partial \psi^*}{\partial x}(x,t)\, \frac{\partial v}{\partial x}(x,t)\, dx = 0$$

Thus, we have the week form: For each fixed $t \in (0,T]$, find $\psi^*(\cdot,t) \in H_0^1(D^*(t))$ such that

$$\mathrm{i}\left\langle \frac{\partial}{\partial t}\psi^*(\cdot,t), v \right\rangle_t - \bar{\alpha}(k)\left\langle \frac{\partial}{\partial x}\psi^*(\cdot,t), \frac{\partial}{\partial x}v \right\rangle_t = 0, \quad \text{for all } v \in H_0^1(D^*(t)), \tag{5.10}$$

$$\psi^*(x,0) = \mu(x)g_k(x) + R(x;\omega^*), \quad x \in \overline{D^*(0)}. \tag{5.11}$$

For each fixed $t \in [0,T]$, let $V_{h(t)}$ be a finite dimensional subspace of $H_0^1(D^*(t))$, spanned by continuous splines of degree $p \geq 1$ defined on a spatial mesh of $D^*(t)$, say with uniform size $h(t)$, satisfying the approximation theory property that for any $v \in H^s(D^*(t))$, $s \geq p+1$,

$$\inf_{w_{h(t)} \in V_{h(t)}} \|v - w_h\|_{L^2(D^*(t))} = O([h(t)]^{p+1}).$$

With this abstract setting, a theoretical way of defining a semi-discrete Galerkin spline FEM is to find $\psi^*_{h(t)}(\cdot,t) \in V_{h(t)}$ satisfying (5.10) and (5.11), with $\psi^*$, $H^1_0(D^*(t))$, and $\mu g_k + R(\cdot,\boldsymbol{\omega}^*)$ replaced, respectively, with $\psi^*_{h(t)}$, $V_{h(t)}$ and $P_{h(t)}(\mu g_k + R(\cdot,\boldsymbol{\omega}^*))$. Here, $P_{h(t)}(\mu g_k + R(\cdot,\boldsymbol{\omega}^*))$ is a projection of the initial data $\mu g_k + R(\cdot,\boldsymbol{\omega}^*)$ onto $V_{h(t)}$.

Given that our model problem requires a moving domain, there is a need to efficiently setup the time-dependent non-standard stiffness, mass, and related hybrid matrices. In particular, unlike the fixed-domain case, the linear spline functions depend nonlinearly on the time-mesh and hence efficient setting up of the moving-mesh FEM algebraic system, including an appropriate Jacobian of the spline basis functions, is required.

To this end, below we give concrete details of a fully-discrete moving-mesh FEM, which includes setting up the system matrices and the resulting algebraic system that needs to be solved at each discrete time step. We restrict our attention below to the continuous linear splines ($p = 1$) and the approach can be generalized for high-order splines and non-uniform mesh parameters.

For discretization of the operator $\dfrac{\partial}{\partial t}$, we consider a uniform mesh of the time interval $[0,T]$ with mesh-width $\Delta t = T/M_{\Delta t}$ and denote the resulting discrete time points as $t_i,\ i = 0,\ldots,M_{\Delta t}$ with $t_0 = 0$ and $t_{M_{\Delta t}} = T$. Thus, we first restrict the above model to the discrete time levels: For each $m = 1,\ldots,M_{\Delta t}$, find $\psi^*(\cdot,t_m) \in H^1_0(D^*(t_m))$ such that

$$\begin{aligned} \mathrm{i}\left\langle \frac{\partial}{\partial t}\psi^*(\cdot,t_m), v\right\rangle_{t_m} - \bar{\alpha}(k)\left\langle \frac{\partial}{\partial x}\psi^*(\cdot,t_m), \frac{\partial}{\partial x}v\right\rangle_{t_m} &= 0, \quad v \in H^1_0(D^*(t_m)), \qquad (5.12)\\ \psi^*(x,0) &= \mu(x)g_k(x) + R(x;\boldsymbol{\omega}^*), \\ & \qquad x \in \overline{D^*(0)}. \qquad (5.13) \end{aligned}$$

For each $m = 0,\ldots,M_{\Delta t}$, we choose a uniform spatial mesh parameter $h(t_m) := \dfrac{\gamma(t_m)-\beta}{N_h}$ and partition the time-dependent spatial domain $\overline{D^*(t_m)}$ into equally spaced points $\mathrm{x}_j(t_m),\ j = 1,\ldots,N_h+1$ so that $\mathrm{x}_1(t_m) = \beta$ and $\mathrm{x}_{N_h+1}(t_m) = \gamma(t_m)$. Thus, the moving-mesh is determined by the time-dependent spatial nodal vector $\mathbf{x}^m = [\mathrm{x}_1(t_m),\ldots,\mathrm{x}_{N_h+1}(t_m)]^T$ and the elements $I_j(t_m) = [\mathrm{x}_j(t_m),\mathrm{x}_{j+1}(t_m)]$, for $j = 1,\ldots,N_h$, and $m = 0,\ldots,M_{\Delta t}$. In Figure 5.2, we

visualize the time-dependent spatial mesh points obtained using a coarse mesh with $M_{\Delta t} = N_h = 20$, for two realizations of the shutter speed function $\gamma(t)$.

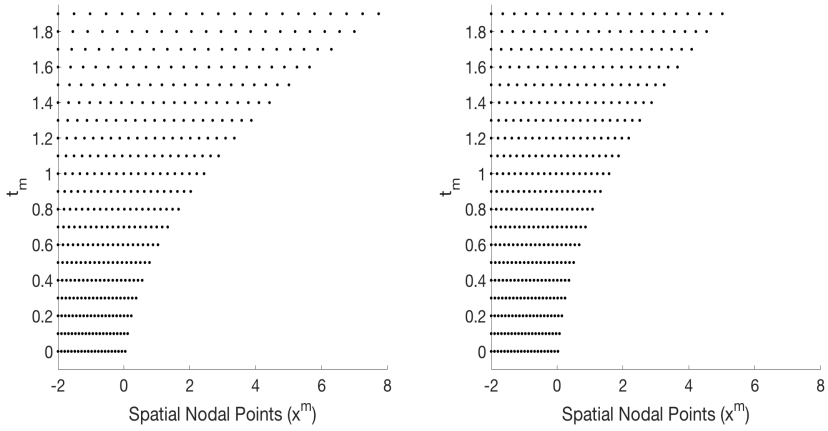



Figure 5.2: Two realizations of moving-mesh configurations.

For each discrete time $t_m,\ m = 0, \ldots, M_{\Delta t}$, we seek an FEM approximate solution $\psi^*_{h(t_m)}(\cdot, t_m) \in V_{h(t_m)}$ of (5.12)-(5.13), where

$$V_h(t_m) := \text{span}\{\phi_j(x; \mathbf{x}^m),\ j = 1, \ldots, N_h + 1\}, \tag{5.14}$$

and for $j = 2, \ldots, N_h$, the basis functions are defined as

$$\phi_j(x; \mathbf{x}^m) = \begin{cases} \dfrac{x - \mathrm{x}_{j-1}(t_m)}{\mathrm{x}_j(t_m) - \mathrm{x}_{j-1}(t_m)}, & x \in I_{j-1}(t_m) \\ \dfrac{\mathrm{x}_{j+1}(t_m) - x}{\mathrm{x}_{j+1}(t_m) - \mathrm{x}_j(t_m)}, & x \in I_j(t_m) \\ 0 \quad , & \text{otherwise} \end{cases} . \tag{5.15}$$

In addition, we define $\phi_1$ and $\phi_{N_h+1}$ as below

$$\phi_1(x; \mathbf{x}^m) = \begin{cases} \dfrac{\mathrm{x}_2(t_m) - x}{\mathrm{x}_2(t_m) - \mathrm{x}_1(t_m)}, & x \in I_1(t_m) \\ 0 \quad , & \text{otherwise} \end{cases}$$

$$\phi_{N_h+1}(x; \mathbf{x}^m) = \begin{cases} \dfrac{x - \mathrm{x}_{N_h}(t_m)}{\mathrm{x}_{N_h+1}(t_m) - \mathrm{x}_{N_h}(t_m)}, & x \in I_{N_h}(t_m) \\ 0 \quad , & \text{otherwise} \end{cases} .$$

For $m = 0, \ldots, M_{\Delta t}$, with $h_m = h(t_m)$, to compute $\psi^*_{h_m}(\cdot, t_m; \mathbf{x}^m) := \psi^*_{h_m}(\cdot, t_m) \in V_{h(t_m)}$, we use the ansatz:

$$\psi^*_{h_m}(x, t_m; \mathbf{x}^m) = \sum_{i=1}^{N_h+1} \Psi_i(t_m)\, \phi_i(x; \mathbf{x}^m), \qquad x \in D^*(t_m). \tag{5.16}$$

In (5.16), using the initial condition (5.13), we obtain its projection onto $V_{h(t_0)}$ by taking

$$\Psi_i(t_0) = \mu(\mathrm{x}_i(t_0)) g_k(\mathrm{x}_i(t_0)) + R(\mathrm{x}_i(t_0); \boldsymbol{\omega}), \qquad i = 1, \ldots, N_h + 1. \tag{5.17}$$

For $m = 1, \ldots, M_{\Delta t}$, we obtain the $N_h - 1$ unknown coefficients $\Psi_i(t_m)$ by forcing the representation in (5.16) to satisfy a fully-discrete version of (5.12) by a second-order discretization of $\dfrac{\partial}{\partial t}$ using the mid-time-nodal points $t_{m-1/2} = (t_m + t_{m-1})/2$ with mesh-width $\Delta t/2$. Because of the nonlinear dependence of the basis functions $\phi_i$ in the time variable, it is convenient to first consider a simplified representation of $\dfrac{\partial}{\partial t}\psi^*_{h_m}(\cdot, t_m)$. Below we use the notation $\mathbf{x}^m_j$ to denote the $j$-th component of the vector $\mathbf{x}^m$, for $j = 1, \ldots, N_h + 1$. Using (5.16) we then obtain

$$\begin{aligned}
&\frac{\partial}{\partial t}\psi^*(x, t_m; \mathbf{x}^m) \\
&= \sum_{i=1}^{N_h+1} \frac{d}{dt}\Psi_i(t_m)\phi_i(x; \mathbf{x}^m) + \sum_{i=1}^{N_h+1} \Psi_i(t_m) \sum_{j=1}^{N_h+1} \frac{\partial}{\partial \mathbf{x}^m_j}\phi_i(x; \mathbf{x}^m)\frac{d}{dt}\mathrm{x}_j(t_m) \\
&= \sum_{i=1}^{N_h+1} \frac{d}{dt}\Psi_i(t_m)\phi_i(x; \mathbf{x}^m) + \sum_{j=1}^{N_h+1} \left[\sum_{i=1}^{N_h+1} \Psi_i(t_m)\frac{\partial}{\partial \mathbf{x}^m_j}\phi_i(x; \mathbf{x}^m)\right]\frac{d}{dt}\mathrm{x}_j(t_m) \\
&= \sum_{i=1}^{N_h+1} \frac{d}{dt}\Psi_i(t_m)\phi_i(x; \mathbf{x}^m) + \sum_{j=1}^{N_h+1} \left[\frac{\partial}{\partial \mathbf{x}^m_j}\psi^*(x, t_m; \mathbf{x}^m)\right]\frac{d}{dt}\mathrm{x}_j(t_m) \\
&= \sum_{i=1}^{N_h+1} \frac{d}{dt}\Psi_i(t_m)\phi_i(x; \mathbf{x}^m) + \sum_{j=1}^{N_h+1} \left[-\frac{\partial}{\partial x}\psi^*(x, t_m; \mathbf{x}^m)\phi_j(x; \mathbf{x}^m)\right]\frac{d}{dt}\mathrm{x}_j(t_m),
\end{aligned} \tag{5.18}$$

where for the bracketed terms, in the penultimate step we used (5.16) and the final step can be obtained by calculating derivatives of the basis functions in (5.15), w.r.t. the components in $\mathbf{x}^m$ and relating to the derivative w.r.t. the spatial variable $x$. Thus, using (5.18),

$$\frac{\partial}{\partial t}\psi^*(x, t_m; \mathbf{x}^m) = \sum_{i=1}^{N_h+1} \left[\frac{d}{dt}\Psi_i(t_m) - \frac{d}{dt}\mathrm{x}_i(t_m)\frac{\partial}{\partial x}\psi^*(x, t_m; \mathbf{x}^m)\right]\phi_i(x; \mathbf{x}^m). \tag{5.19}$$

The above simplified representation yields, in addition to the standard fixed-mesh mass matrix, terms involving discretization of moving mesh terms and an associated matrix with entries composed of both the basis functions and their derivatives.

### 5.2.1 A fully-discrete moving mesh FEM and complex algebraic systems

In this section, we use the following simplified notation to approximate the time derivatives in (5.19) at $t_{m-1/2} = (m-1/2)\Delta t$, for $m = 1, \ldots, M_{\Delta t}$ with second order accuracy obtained by letting

$$\frac{d}{dt}\Psi_i(t_{m-1/2}) \approx \frac{\Psi_i^m - \Psi_i^{m-1}}{\Delta t} \text{ and } \frac{d}{dt}\mathrm{x}_i(t_{m-1/2}) \approx \frac{\mathrm{x}_i^m - \mathrm{x}_i^{m-1}}{\Delta t}. \tag{5.20}$$

Following a Crank-Nicolson type approach, for $i = 1, \ldots, N_h + 1$, using (5.15), let $\phi_i^{m-1/2}(x) = \phi_i(x; \mathbf{x}^{m-1/2}), \quad x \in \overline{D^*(t_{m-1/2})}$. Using the anstaz (5.16), we consider the approximation, for $x \in \overline{D^*(t_{m-1/2})}$,

$$\psi^*_{h_{m-1/2}}(x, t_{m-1/2}; \mathbf{x}^{m-1/2}) \approx \psi^{*,m-1/2}(x) := \sum_{i=1}^{N_h+1} \frac{\Psi_i^m + \Psi_i^{m-1}}{2}\, \phi_i^{m-1/2}(x). \tag{5.21}$$

Using (5.19), (5.20) and (5.21), our fully-discrete moving-mesh FEM approximation to (5.10) can be described as follows. Using the data $\Psi_i^0 \in \mathbb{C}$ in (5.17), for $m = 1, \ldots, M_{\Delta t},\ i = 1, \ldots, N_h + 1$, find the coefficients $\Psi_i^m \in \mathbb{C}$ of the ansatz (5.16) of the approximation $\psi^*_{h_m}(x, t_m; \mathbf{x}^m) \in V_{h(t_m)}$ such that

$$\begin{aligned} &\mathrm{i}\left\langle \sum_{i=1}^{N_h+1} \left\{ \frac{\Psi_i^m - \Psi_i^{m-1}}{\Delta t} - \frac{\mathrm{x}_i^m - \mathrm{x}_i^{m-1}}{\Delta t} \frac{\partial}{\partial x}\psi^{*,m-1/2} \right\} \phi_i^{m-1/2}, \phi_j^{m-1/2} \right\rangle_{t_{m-1/2}} \\ &-\bar{\alpha}(k) \left\langle \frac{\partial}{\partial x}\psi^{*,m-1/2}, \frac{\partial}{\partial x}\phi_j^{m-1/2} \right\rangle_{t_{m-1/2}} = 0. \end{aligned}$$

$\Longleftrightarrow$

$$\begin{aligned} &\mathrm{i}\left\langle \sum_{i=1}^{N_h+1} \left\{ \frac{\Psi_i^m - \Psi_i^{m-1}}{\Delta t}\phi_i^{m-1/2} - \frac{\mathrm{x}_i^m - \mathrm{x}_i^{m-1}}{\Delta t} \sum_{r=1}^{N_h+1} \left( \frac{\Psi_r^m + \Psi_r^{m-1}}{2} \frac{\partial \phi_r^{m-1/2}}{\partial x} \right) \phi_i^{m-1/2} \right\}, \phi_j^{m-1/2} \right\rangle_{t_{m-1/2}} \\ &-\bar{\alpha}(k) \left\langle \sum_{i=1}^{N_h+1} \frac{\Psi_i^m + \Psi_i^{m-1}}{2} \frac{\partial}{\partial x}\phi_i^{m-1/2}, \frac{\partial}{\partial x}\phi_j^{m-1/2} \right\rangle_{t_{m-1/2}} = 0 \end{aligned}$$

$$\Longleftrightarrow$$

$$\mathrm{i}\left\langle -\sum_{i=1}^{N_h+1}\frac{\mathrm{x}_i^m-\mathrm{x}_i^{m-1}}{\Delta t}\sum_{r=1}^{N_h+1}\left(\frac{\Psi_r^m+\Psi_r^{m-1}}{2}\frac{\partial\phi_r^{m-1/2}}{\partial x}\right)\phi_i^{m-1/2},\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}}$$
$$+\mathrm{i}\left\langle\sum_{i=1}^{N_h+1}\frac{\Psi_i^m-\Psi_i^{m-1}}{\Delta t}\phi_i^{m-1/2},\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}}-\bar{\alpha}(k)\left\langle\sum_{i=1}^{N_h+1}\frac{\Psi_i^m+\Psi_i^{m-1}}{2}\frac{\partial}{\partial x}\phi_i^{m-1/2},\frac{\partial}{\partial x}\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}}=0$$

$$\Longleftrightarrow$$

$$\mathrm{i}\sum_{i=1}^{N_h+1}(\Psi_i^m-\Psi_i^{m-1})\left\langle\phi_i^{m-1/2},\phi_j^{m-1/2}\right\rangle+\frac{\mathrm{i}}{2}\sum_{i=1}^{N_h+1}(\Psi_i^m+\Psi_i^{m-1})\left\langle-\frac{\partial\phi_i^{m-1/2}}{\partial x}\sum_{r=1}^{N_h+1}(\mathrm{x}_r^m-\mathrm{x}_r^{m-1})\phi_r^{m-1/2},\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}}$$
$$-\frac{\bar{\alpha}(k)\Delta t}{2}\sum_{i=1}^{N_h+1}(\Psi_i^m+\Psi_i^{m-1})\left\langle\frac{\partial}{\partial x}\phi_i^{m-1/2},\frac{\partial}{\partial x}\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}}=0.$$

Expanding further, we obtain

$$\sum_{i=1}^{N_h+1}\Psi_i^m\left[\mathrm{i}\left\langle\phi_i^{m-1/2},\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}}-\frac{\bar{\alpha}(k)\Delta t}{2}\left\langle\frac{\partial}{\partial x}\phi_i^{m-1/2},\frac{\partial}{\partial x}\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}}\right.$$
$$\left.-\frac{\mathrm{i}}{2}\left\langle\frac{\partial\phi_i^{m-1/2}}{\partial x}\sum_{r=1}^{N_h+1}(\mathrm{x}_r^m-\mathrm{x}_r^{m-1})\phi_r^{m-1/2},\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}}\right]$$
$$=\sum_{i=1}^{N_h+1}\Psi_i^{m-1}\left[\mathrm{i}\left\langle\phi_i^{m-1/2},\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}}+\frac{\bar{\alpha}(k)\Delta t}{2}\left\langle\frac{\partial}{\partial x}\phi_i^{m-1/2},\frac{\partial}{\partial x}\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}}\right.$$
$$\left.+\frac{\mathrm{i}}{2}\left\langle\frac{\partial\phi_i^{m-1/2}}{\partial x}\sum_{r=1}^{N_h+1}(\mathrm{x}_r^m-\mathrm{x}_r^{m-1})\phi_r^{m-1/2},\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}}\right]. \tag{5.22}$$

For $m=0,\ldots,M_{\Delta t}$, consider the $N_h-1$ dimensional vector $\underset{\sim}{\Psi}^m=[\Psi_2^m,\Psi_3^m,\ldots,\Psi_{N_h}^m]^T$, and consider the following $(N_h-1)\times(N_h-1)$ mass, stiffness, and hybrid matrices, with $(i,j)$-th entries being

$$[\mathcal{M}^m]_{ij}=\left\langle\phi_i^{m-1/2},\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}},\quad[\mathcal{A}^m]_{ij}=\left\langle\frac{\partial}{\partial x}\phi_i^{m-1/2},\frac{\partial}{\partial x}\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}}$$
$$[Q^m]_{ij}=\left\langle\frac{\partial\phi_i^{m-1/2}}{\partial x}\sum_{r=1}^{N_h+1}(\mathrm{x}_r^m-\mathrm{x}_r^{m-1})\phi_r^{m-1/2},\phi_j^{m-1/2}\right\rangle_{t_{m-1/2}},$$
$$i,j=1,\ldots,N_h+1. \tag{5.23}$$

Using these forms, we can then solve for the exact entries of these matrices for $i, j = 2, \ldots, N_h$ as provided below

$$[\dot{\mathcal{A}}^m]_{ij} = \begin{cases} \int_{x_{j-1}(t_{m-1/2})}^{x_j(t_{m-1/2})} \left[\frac{1}{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}\right]^2 dx + \int_{x_j(t_{m-1/2})}^{x_{j+1}(t_{m-1/2})} \left[\frac{-1}{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}\right]^2 dx, & \text{if} \quad i = j \\ \int_{x_j(t_{m-1/2})}^{x_{j+1}(t_{m-1/2})} \left[\frac{1}{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}\right]\left[\frac{-1}{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}\right] dx, & \text{if} \quad i = j+1 \\ \int_{x_{j-1}(t_{m-1/2})}^{x_j(t_{m-1/2})} \left[\frac{-1}{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}\right]\left[\frac{1}{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}\right] dx, & \text{if} \quad i = j-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

$$[\mathcal{A}^m]_{ij} = \begin{cases} \frac{1}{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})} + \frac{1}{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}, & \text{if} \quad i = j \\ \frac{-1}{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}, & \text{if} \quad i = j+1 \\ \frac{-1}{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}, & \text{if} \quad i = j-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

$$[\mathcal{M}^m]_{ij} = \begin{cases} \int_{x_{j-1}(t_{m-1/2})}^{x_j(t_{m-1/2})} \left[\frac{x - x_{j-1}(t_{m-1/2})}{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}\right]^2 dx + \int_{x_j(t_{m-1/2})}^{x_{j+1}(t_{m-1/2})} \left[\frac{x_{j+1}(t_{m-1/2}) - x}{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}\right]^2 dx, & \text{if} \quad i = j \\ \int_{x_j(t_{m-1/2})}^{x_{j+1}(t_{m-1/2})} \left[\frac{x - x_j(t_{m-1/2})}{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}\right]\left[\frac{x_{j+1}(t_{m-1/2}) - x}{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}\right] dx, & \text{if} \quad i = j+1 \\ \int_{x_{j-1}(t_{m-1/2})}^{x_j(t_{m-1/2})} \left[\frac{x_j(t_{m-1/2}) - x}{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}\right]\left[\frac{x - x_{j-1}(t_{m-1/2})}{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}\right] dx, & \text{if} \quad i = j-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

$$[\mathcal{M}^m]_{ij} = \begin{cases} \frac{x_{j+1}(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}{3}, & \text{if} \quad i = j \\ \frac{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}{6}, & \text{if} \quad i = j+1 \\ \frac{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}{6}, & \text{if} \quad i = j-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

$$[Q^m]_{ji} = \begin{cases} \int_{x_{j-1}(t_{m-1/2})}^{x_j(t_{m-1/2})} \left[\frac{1}{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}\right]\left[\sum_{r=2}^{N}(x_r^m - a_r^{m-1})\alpha_r^{m-1/2}\right]\left[\frac{x - x_{j-1}(t_{m-1/2})}{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}\right]dx \\ + \int_{x_j(t_{m-1/2})}^{x_{j+1}(t_{m-1/2})} \left[\frac{-1}{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}\right]\left[\sum_{r=2}^{N}(x_r^m - x_r^{m-1})\alpha_r^{m-1/2}\right]\left[\frac{x_{j+1}(t_{m-1/2}) - x}{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}\right]dx, & \text{if} \quad i = j \\ \int_{x_j(t_{m-1/2})}^{x_{j+1}(t_{m-1/2})} \left[\frac{1}{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}\right]\left[\sum_{r=2}^{N}(x_r^m - x_r^{m-1})\alpha_r^{m-1/2}\right]\left[\frac{x_{j+1}(t_{m-1/2}) - x}{x_{j+1}(t_{m-1/2}) - x_j(t_{m-1/2})}\right]dx, & \text{if} \quad i = j+1 \\ \int_{x_{j-1}(t_{m-1/2})}^{x_j(t_{m-1/2})} \left[\frac{-1}{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}\right]\left[\sum_{r=2}^{N}(x_r^m - x_r^{m-1})\alpha_r^{m-1/2}\right]\left[\frac{x - x_{j-1}(t_{m-1/2})}{x_j(t_{m-1/2}) - x_{j-1}(t_{m-1/2})}\right]dx, & \text{if} \quad i = j-1 \\ 0 & , \quad \text{otherwise} \end{cases}$$

$$[Q^m]_{ij} = \begin{cases} \frac{x_{j-1}^m - x_{j-1}^{m-1} - x_{j+1}^m + x_{j+1}^{m-1}}{6}, & \text{if} \quad i = j \\ \frac{-2x_j^m + 2x_j^{m-1} - x_{j-1}^m + x_{j-1}^{m-1}}{6}, & \text{if} \quad i = j+1 \\ \frac{2x_j^m - 2x_j^{m-1} + x_{j+1}^m - x_{j+1}^{m-1}}{6}, & \text{if} \quad i = j-1 \\ 0 & , \quad \text{otherwise.} \end{cases}$$

Using these matrices in (5.22), the moving-mesh FEM algebraic system, for each $m = 1, \ldots, M_{\Delta t}$, can be written as

$$\left[\mathrm{i}\mathcal{M}^m - \frac{\mathrm{i}}{2}\mathcal{Q}^m - \frac{\bar{\alpha}(k)\Delta t}{2}\mathcal{A}^m\right]\underset{\sim}{\Psi}^m = \left[\mathrm{i}\mathcal{M}^m + \frac{\mathrm{i}}{2}\mathcal{Q}^m + \frac{\bar{\alpha}(k)\Delta t}{2}\mathcal{A}^m\right]\underset{\sim}{\Psi}^{m-1}. \tag{5.24}$$

Using the initial data $\underset{\sim}{\Psi}^0$, we can use direct complex algebraic system solvers for (5.24), to compute the unknown coefficients in (5.16) and hence simulate the approximate wave function $\psi^*_{h_m}(\cdot, t_m; \mathbf{x}^m) = \psi^*_{h_m}(\cdot, t_m) \in V_{h(t_m)}$, for $m = 1, \ldots, M_{\Delta t}$.

### 5.2.2 Numerical experiments: Exact Deterministic simulation

In this section, we consider the deterministic version of the model and make comparisons against an exact solution. To make this comparison, we consider the PDE given by (5.25) and (5.26). In particular, we consider the moving-mesh FEM model for $0 < t \leq 1.5$ on $[\beta, \gamma(t)]$ induced by a moving shutter with speed $\gamma(t) = acon * t^2$, $acon = 0.25$, and the wavenumber dependent initial state induced at $\beta = -2$. Thus, the initial computational domain at $t = 0$ is $[-2, 0]$ and grows until the computational domain is $[-2, 0.5625]$, at the final time $T = 1.5$. Additionally, we will let $\alpha = c/(2\nu)$, $w = \alpha k^2$, and $k_c$ be an integer. For this particular configuration we will choose the initial state to be given by (5.26), which allows us to make a direct comparison against the exact solution (5.27). The forcing function $F(x, t)$ is then found by plugging in (5.27) into the left hand side of (5.25).

$$\mathrm{i}\frac{\partial}{\partial t}\psi(x,t) + \alpha\frac{\partial^2}{\partial x^2}\psi(x,t) \quad = \quad F(x,t), \quad x \in D(t), \quad 0 < t \leq T, \tag{5.25}$$

$$\psi(x,0) = e^{\mathrm{i}kx}\sin(k_c\pi(x - \gamma(0))(x - \beta)), \qquad x \in \overline{D(0)}. \tag{5.26}$$

$$\psi^{exact}(x,t) = e^{\mathrm{i}(kx - wt)}\sin(k_c\pi(x - \gamma(t))(x - \beta)) \tag{5.27}$$

Using the scenario outlined above, we now conduct a convergence study. For this study we compute the relative error using the full solution at the mesh points. We consider a uniform mesh of the time interval $[0, T]$ with mesh-width $\Delta t = T/M_{\Delta t}$ and denote the resulting discrete time

points as $t_i,\ i = 0, \ldots, M_{\Delta t}$ with $t_0 = 0$ and $t_{M_{\Delta t}} = T$. For each $m = 0, \ldots, M_{\Delta t}$, we choose a uniform spatial mesh parameter $h(t_m) := \dfrac{\gamma(t_m) - \beta}{N_h}$ and partition the time-dependent spatial domain $\overline{D(t_m)}$ into equally spaced points $\mathrm{x}_j(t_m),\ j = 1, \ldots, N_h + 1$ so that $\mathrm{x}_1(t_m) = \beta$ and $\mathrm{x}_{N_h+1}(t_m) = \gamma(t_m)$.

Let $t$ and $x$ be the mesh points created as described above, $\psi_{h,\Delta t}(x, t)$ be the finite element solution at these points, $\psi^{exact}(x, t)$ be the exact solution at the mesh points, and the relative error for $N_h$ and $M_{\Delta t}$ be given by:

$$||e_{h,\Delta t}||_2 = \frac{||\psi_{h,\Delta t}(x, t) - \psi^{exact}(x, t)||_2}{||\psi^{exact}(x, t)||_2}. \tag{5.28}$$

This provides the results in Table 5.2, Table 5.3, Table 5.5, Table 5.6, Table 5.8, and Table 5.9. From these results, we can verify that we are obtaining the correct order of convergence for all of the wave numbers. One important thing to note from these results is that as $k$ increases, the relative error tends to increase. This is due to the fact that larger $k$ values increase the complexity of the real and imaginary parts of the solution. For this reason, to maintain a small error one would need to apply finer grids in both the spatial and temporal domains. The complexity of the solution for different $k$ values can bee seen in Figure 5.3, where we compare $k = 1$ vs. $k = 10$ at the final time $T = 1.5$. The run times for these grid sizes using a highly optimized C++ code are shown in Table 5.10.

Table 5.1: Convergence of the moving-mesh FEM DiT model using $\nu = 0.5, c = 1.0$, and $k_c = 1.0$.

Table 5.2: $k = 1$

| $N_h$ | $M_{\Delta t}$ | $\|e_{h,\Delta t}\|_2$ | EOC of $\|e_{h,\Delta t}\|_2$ |
|---|---|---|---|
| 20 | 20 | 4.363680e-03 | |
| 40 | 40 | 1.055748e-03 | 2.0473 |
| 80 | 80 | 2.589717e-04 | 2.0274 |
| 160 | 160 | 6.423185e-05 | 2.0114 |
| 320 | 320 | 1.600215e-05 | 2.0050 |
| 640 | 640 | 3.993246e-06 | 2.0026 |
| 1280 | 1280 | 9.973648e-07 | 2.0014 |

Table 5.3: $k = 2$

| $N_h$ | $M_{\Delta t}$ | $\|e_{h,\Delta t}\|_2$ | EOC of $\|e_{h,\Delta t}\|_2$ |
|---|---|---|---|
| 20 | 20 | 1.360732e-02 | |
| 40 | 40 | 3.274745e-03 | 2.0549 |
| 80 | 80 | 8.132506e-04 | 2.0096 |
| 160 | 160 | 2.031017e-04 | 2.0015 |
| 320 | 320 | 5.077913e-05 | 1.9999 |
| 640 | 640 | 1.269783e-05 | 1.9997 |
| 1280 | 1280 | 3.174994e-06 | 1.9998 |

Table 5.4: Convergence of the moving-mesh FEM DiT model using $\nu = 0.5, c = 1.0$, and $k_c = 1.0$.

Table 5.5: $k = 4$

| $N_h$ | $M_{\Delta t}$ | $\|\|e_{h,\Delta t}\|\|_2$ | EOC of $\|\|e_{h,\Delta t}\|\|_2$ |
|---|---|---|---|
| 20 | 20 | 7.575651e-01 | |
| 40 | 40 | 1.428884e-01 | 2.4065 |
| 80 | 80 | 3.026698e-02 | 2.2391 |
| 160 | 160 | 7.199047e-03 | 2.0719 |
| 320 | 320 | 1.777467e-03 | 2.0180 |
| 640 | 640 | 4.429835e-04 | 2.0045 |
| 1280 | 1280 | 1.106675e-04 | 2.0010 |

Table 5.6: $k = 6$

| $N_h$ | $M_{\Delta t}$ | $\|\|e_{h,\Delta t}\|\|_2$ | EOC of $\|\|e_{h,\Delta t}\|\|_2$ |
|---|---|---|---|
| 20 | 20 | 8.612781e+00 | |
| 40 | 40 | 1.474075e+00 | 2.5467 |
| 80 | 80 | 5.192423e-01 | 1.5053 |
| 160 | 160 | 1.277621e-01 | 2.0230 |
| 320 | 320 | 3.147773e-02 | 2.0211 |
| 640 | 640 | 7.834649e-03 | 2.0064 |
| 1280 | 1280 | 1.956400e-03 | 2.0017 |

Table 5.7: Convergence of the moving-mesh FEM DiT model using $\nu = 0.5, c = 1.0$, and $k_c = 1.0$.

Table 5.8: $k = 8$

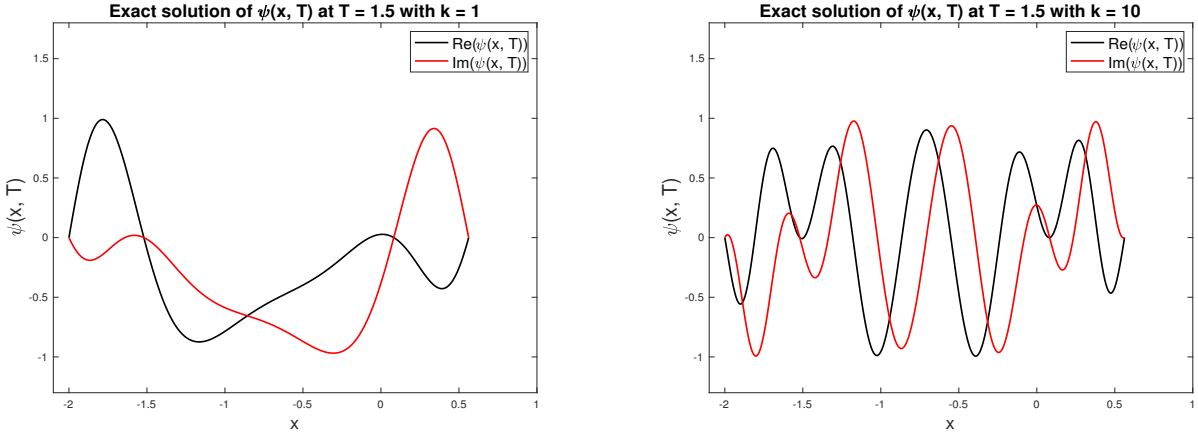
| $N_h$ | $M_{\Delta t}$ | $\|\|e_{h,\Delta t}\|\|_2$ | EOC of $\|\|e_{h,\Delta t}\|\|_2$ |
|---|---|---|---|
| 20 | 20 | 1.564787 | |
| 40 | 40 | 9.220694 | -2.5589 |
| 80 | 80 | 6.748159e-01 | 3.7723 |
| 160 | 160 | 3.368284e-01 | 1.0025 |
| 320 | 320 | 1.022383e-01 | 1.7201 |
| 640 | 640 | 2.678702e-02 | 1.9324 |
| 1280 | 1280 | 6.774839e-03 | 1.9833 |

Table 5.9: $k = 10$

| $N_h$ | $M_{\Delta t}$ | $\|\|e_{h,\Delta t}\|\|_2$ | EOC of $\|\|e_{h,\Delta t}\|\|_2$ |
|---|---|---|---|
| 20 | 20 | 1.180621e+01 | |
| 40 | 40 | 1.728772e+00 | 2.7717 |
| 80 | 80 | 2.998958e+00 | -0.7947 |
| 160 | 160 | 7.365811e-01 | 2.0256 |
| 320 | 320 | 1.417142e-01 | 2.3779 |
| 640 | 640 | 2.769555e-02 | 2.3553 |
| 1280 | 1280 | 7.159549e-03 | 1.9517 |

Table 5.10: Run times of FEM solution for various grid sizes. Data for this table is generated using a highly optimized C++ version of the FEM solution.

| $N_h$ | $M_{\Delta t}$ | Run time (Seconds) |
|---|---|---|
| 20 | 20 | less than 0.01 |
| 40 | 40 | less than 0.01 |
| 80 | 80 | 0.03 |
| 160 | 160 | 0.12 |
| 320 | 320 | 0.46 |
| 640 | 640 | 1.83 |
| 1280 | 1280 | 7.35 |




(a) (b)

Figure 5.3: Real and imaginary solutions of the exact solution (5.27) at T=1.5. In the left panel (a) we provide the exact solution with $k = 1$ and the right panel (b) displays $k = 10$.

### 5.2.3 Numerical experiments: Deterministic simulation

In this section, using the KL expansion in (5.4) with $d = 64$, we demonstrate the accuracy of the moving-mesh FEM model to compute $\psi^*_{h_m}(x, t_m; \mathbf{x}^m)$ in (5.16), for $0 < t \leq 2$ and for one realization sample $\widetilde{\omega}^* = (\omega^*, \omega^*) \in \widetilde{\mathbf{\Omega}} \subset \mathbb{R}^{65}$ with each of the 64 components of $\boldsymbol{\omega}^*$ being a uniformly distributed random variable in $[-0.5, 0.5]$ and $\omega^*$ is the non-deterministic speed of the shutter, which is a uniformly distributed random variable in $[1, 2]$. Using the resulting deterministic setting, we approximate and simulate the FEM discrete version of the deterministic continuous model (5.7)–(5.9) on $[\beta, \gamma(t)]$ induced by a moving shutter with speed $\gamma\,'(t) = 2\omega^* t$ and the

wavenumber dependent initial state induced at $\beta = -2$. Thus, the initial computational domain at $t = 0$ is $[-2, 0]$ and grows until the computational domain is $[-2, 4\omega^*]$, at the final time $T = 2$. For the randomly perturbed initial state in (5.8), we choose wavenumber $k = \pi$ so that $g_k(x) = \exp(\mathrm{i}kx)$ and $\overline{\alpha}(k) = k^2$, and to impose the homogeneous Dirichlet boundary conditions at the boundaries of $D^*(0)$, we choose $\mu(x) = (x - \beta)(x - \gamma(0))$.

For the convergence study of our moving-mesh FEM computational DiT model, the analytical solution of the continuous model is unknown, for this reason we choose a very fine-grid based solution of the computational model as the reference solution $\psi_{h_m}^{*,fine}(x, t_m; \mathbf{x}^m)$, for $m = 1, \ldots, M_{\Delta t}$, with $M_{\Delta t} = M_{\Delta t}^{fine} = N_h^{fine} = 20 * 2^{10} = 20480$ being the number of fine-grid spatial moving-points, for each $m = 1, \ldots, M_{\Delta t}$.

Fixing the time-mesh and for several values of $N_h$, we compute the nodal error, for $n = 1, \ldots, N_h + 1, \quad m = 1, \ldots, M_{\Delta t}^{fine}$,

$$Err(n, m) = \left| \psi_{h_m}^*(\mathrm{x}_n^m, t_m; \mathbf{x}^m) - \overset{*}{\psi}_{h_m}^{,fine}(\mathrm{x}_n, t_m; \mathbf{x}^m) \right|, \tag{5.29}$$

and hence compute the maximum nodal error with respect to both the time and spatial grid:

$$Err^{max}(N_h; M_{\Delta t}) = \max \{ Err(n, m) : n = 1, \ldots, N_h + 1, \quad m = 1, \ldots, M_{\Delta t} \}. \tag{5.30}$$

We start with the coarse spatial mesh and double the mesh size until $N_h = 10 * 2^{10}$ and using the associated $Err^{max}(N_h)$ in (5.30), we compute the estimated order of convergence (EOC) of the moving-mesh FEM algorithm. We note that the theoretical FEM convergence rates are usually w.r.t. the maximum mesh size, and not w.r.t. the degrees of freedom used in the computation. Given that we fix $N_h$ for all increasing time-dependent sizes of the domain (and hence change the maximum mesh size) we will compute the EOC with respect to the spatial degrees of freedom $N_h$. Results in Table 5.11 demonstrate the accuracy of our moving-mesh FEM computational DiT model for a fixed realization of the initial state and shutter speed in the 65-dimensional stochastic space. In Figure 5.4, we visualize the computed density profile of the simulated solution $\psi$ of the deterministic Schrödinger model on the moving domain, for three discrete time steps.

Table 5.11: Convergence of the moving-mesh FEM DiT model using a reference solution with $M_{\Delta t}^{fine} = N_h^{fine} = 20480$.

| $N_h$ | $Err^{max}(N_h; 20480)$ | EOC of $Err^{max}$ |
|---|---|---|
| 80 | 3.1601e-01 | |
| 160 | 1.3607e-01 | 1.2156 |
| 320 | 5.3584e-02 | 1.3445 |
| 640 | 2.1350e-02 | 1.3276 |
| 1280 | 1.0176e-02 | 1.0691 |
| 2560 | 4.9566e-03 | 1.0377 |
| 5120 | 1.7975e-03 | 1.4633 |
| 10240 | 3.8505e-04 | 2.2229 |

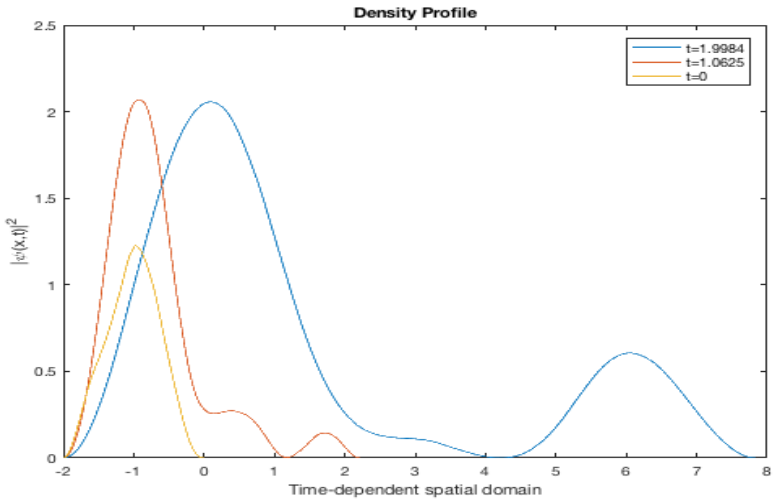



Figure 5.4: Simulated density profiles at three discrete times steps

## 5.3 MC and MLMC simulation of the expected value of QoI

In the following sections we will again describe MC and MLMC for completeness and describe these methods as they apply to DiT model. Key statistical moment metrics to understand the dependence of the QoI in (5.5) on $\widetilde{\omega}$ are its expected value and variance as provided in (3.2) and (3.3), respectively. For each sample $\widetilde{\omega}_j$, the QoI $Q(\widetilde{\omega}_j)$ can be computed by a deterministic moving-mesh DiT FEM model, with a fixed configuration described by $\widetilde{\omega}_j$. The disadvantage of the MC approximation error is in its slow $O(N_{MC}^{-1/2})$ convergence rate. Hence, depending on the variance of the QoI, a large $N_{MC}$ number of samples are required to obtain a few matching digits

of accuracy as the approximate $\mathbb{E}_{\text{MC}}[Q; N_{MC}]$ values are computed, using an increasing number of samples. In addition, the terms in $Q(\widetilde{\omega}_j)$ in (3.2) need to be further approximated by, say $Q_{\Delta t, h_{\Delta t}}(\widetilde{\omega}_j)$, using the moving-mesh FEM algorithm described in the last section, with mesh parameters $M_{\Delta t}$ and $N_h$. The approximation $Q_{\Delta t, h_{\Delta t}}(\widetilde{\omega}_j)$ is then obtained by replacing $\psi(x, T; \widetilde{\omega_j})$ in (5.5) with $\psi_{\Delta t, h_{\Delta t}}(x, T; \widetilde{\omega_j})$, for $j = 1, \ldots, N_{MC}$.

Based on the results in Table 5.11, to ensure that the moving-mesh FEM error is less than the MC error, we took the fine space-time grid with $M_{\Delta t}^{fine} = N_h^{fine} = 20480$ and performed the MC simulation for various values of $N_{MC}$ as described in Table 5.12. In particular, we computed approximations to $\mathbb{E}_{\text{MC}}[Q; N_{MC}]$ as

$$\mathbb{E}_{\text{MC}}\left[Q; N_{MC}; M_{\Delta t}^{fine}; N_h^{fine}\right] = \frac{1}{N_{MC}} \sum_{j=1}^{N_{MC}} Q_{\Delta t^{fine}, h_{\Delta t}^{fine}}(\widetilde{\omega}_j), \tag{5.31}$$

to obtain the results in Table 5.12. These results with certain matching digits in approximate values of $\mathbb{E}_{\text{MC}}\left[Q; N_{MC}; M_{\Delta t}^{fine}; N_h^{fine}\right]$ demonstrate slow convergence of the MC algorithm for tens to hundreds of thousands of MC realizations and for approximate variance error accuracy values $\epsilon = [N_{MC}]^{-1/2}$.

Table 5.12: $\mathbb{E}_{\text{MC}}[Q; N_{MC}]$ values obtained using the standard MC method.

| $N_{MC}$ | $10,000$ | $50,000$ | $100,000$ | $500,000$ |
|---|---|---|---|---|
| $\epsilon = [N_{MC}]^{-1/2}$ | 0.0100 | 0.0045 | 0.0032 | 0.0014 |
| $\mathbb{E}_{\text{MC}}[Q; N_{MC}]$ | 4.6156 | 4.6199 | 4.6282 | 4.6289 |

We note that the MC algorithm does not provide an approach to choose an accuracy $\epsilon$, for which the algorithm adaptively selects various numbers of samples to achieve a similar accuracy. The $\epsilon$ values in Table 5.12 are based on the estimated theoretical rate of convergence, assuming the variance of the QoI (occurring in the order constant) is of moderate size so that MC approximations achieve $\epsilon$ accuracy.

We could achieve such a large number of MC based realizations results in Table 5.12 because of our message passing interface (MPI) based parallel code written in modern versions of Fortran (90+). We ran the MC-FEM parallel code using multiple cluster nodes with each node composed

of dual octa-core Intel X5670 2.93GHz processors. In order to compare the actual computational cost of our naturally parallel MC simulation with a relatively cheaper MLMC algorithm (described in Chapter 3), we provide the total core computational cost (in seconds) in Figure 5.7, which reflects the cost of the results in Table 5.12 for various values of $\epsilon$. Figure 5.7 also demonstrates substantial reduction in the cost by instead using the MLMC algorithm.

### 5.3.1 Multilevel Monte Carlo simulation

Since a thorough explanation of MLMC is available in Chapter 3, in this section we skip a majority of the MLMC theory and focus on the particular aspects of MLMC that should be changed for the physics model in this chapter. In particular, at level $\ell = 0$ the MLMC algorithm is based on simulations using the largest number of adaptively chosen $N^{(\ell)}_{\text{MLMC}}$ samples and obtains an initial approximation similar to the representation in (3.9) using coarse level mesh parameters $[M^{(\ell)}_{\Delta t}; N^{(\ell)}_h]$, and the associated mesh-size $\Delta t^{(\ell)} = T/(M^{(\ell)}_{\Delta t})$ and, for $m = 0, \ldots, M^{(\ell)}_{\Delta t}$, $h^{(\ell)}_{\Delta t}(t_m) = \dfrac{\gamma(t_m) - \beta}{N^{(\ell)}_h}$.
For $\ell = 0, \ldots, L$, using (5.5), we define the $\ell$-th level QoI as

$$Q^{(\ell)}(\widetilde{\omega}) = \int_{D(T;\omega)} |\psi_{\Delta t^{(\ell)}, h^{(\ell)}_{\Delta t}}(x, T; \widetilde{\omega})|^2 dx, \qquad \widetilde{\omega} \in \widetilde{\mathbf{\Omega}}, \tag{5.32}$$

where we use the coarse to finer space-time mesh for each realization of the deterministic computation of $\psi_{\Delta t^{(\ell)}, h^{(\ell)}_{\Delta t}}$ using (as determined by the tabulated values in Table 5.11)

$$M^{(\ell)}_{\Delta t} = N^{(\ell)}_h = 640 * 2^{\ell}, \qquad \ell = 0, 1, 2, 3.$$

In Figure 5.5 we demonstrate the expected reduction in variances of the levels (using a semi-logy plot and $L = 4$) for our moving-mesh FEM based QoI and for demonstration purposes, the variance values were computed using a large number of test samples. Because of the substantially reduced order of the variance, the MC approximations for $\mathbb{E}[Q^{(\ell)} - Q^{(\ell-1)}]$ require substantially fewer $N^{(\ell)}_{\text{MLMC}}$ samples compared to the $N^{(0)}_{\text{MLMC}}$ samples used at the initial level, as demonstrated in Figure 5.6. Results in Figure 5.5, with very low variances of the difference integrands, also suggests that $L = 3$ is sufficient for our MLMC simulation.

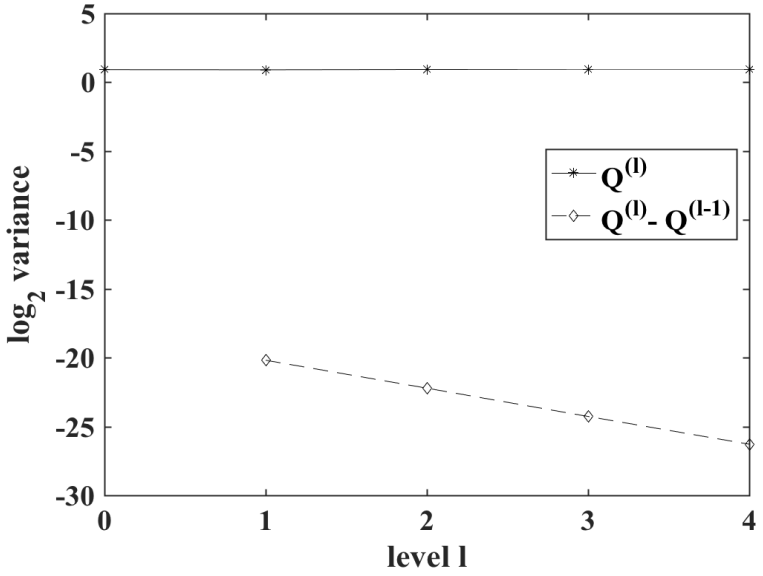



Figure 5.5: Variance for four levels of $Q^{(\ell)}$ and $Q^{(\ell)} - Q^{(\ell-1)}$

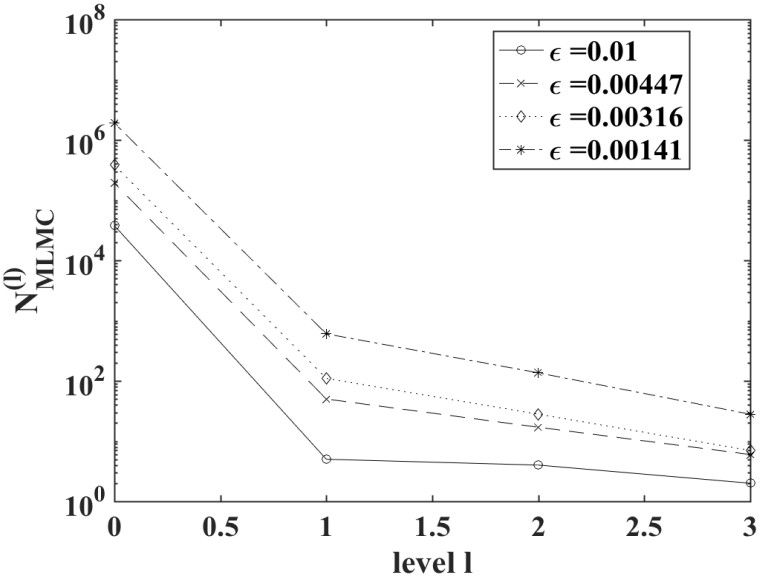



Figure 5.6: Adaptively chosen values of $N^{(\ell)}_{\text{MLMC}}$, $\ell = 0, 1, 2, 3$ for four distinct choices of $\epsilon$ as in Table 5.13.

We conclude this section with results in Table 5.13 that demonstrate the power of our FEM-MLMC algorithm to compute approximate expected values of the QoI with several matching digits at lower computational cost as demonstrated in Figure 5.7. Based on our further

MLMC simulations for other values of $\epsilon = 0.01 * 2^{-j}$ for $j = 0, 1, 2, 3, 4$, we observed an approximate value of $\mathbb{E}[Q]$ is 4.6276, as demonstrated in Table 5.13 for the case $\epsilon = 0.0014$.

Table 5.13: $\mathbb{E}^{(3)}_{\text{MLMC}}\left[Q^{(3)}\right]$ values obtained using the MLMC algorithm with level dependent space-time mesh parameters as stated in (5.33).

| $\epsilon$ | 0.0100 | 0.0045 | 0.0032 | 0.0014 |
|---|---|---|---|---|
| $\mathbb{E}^{(3)}_{\text{MLMC}}\left[Q^{(3)}\right]$ | 4.6256 | 4.6307 | 4.6276 | 4.6276 |

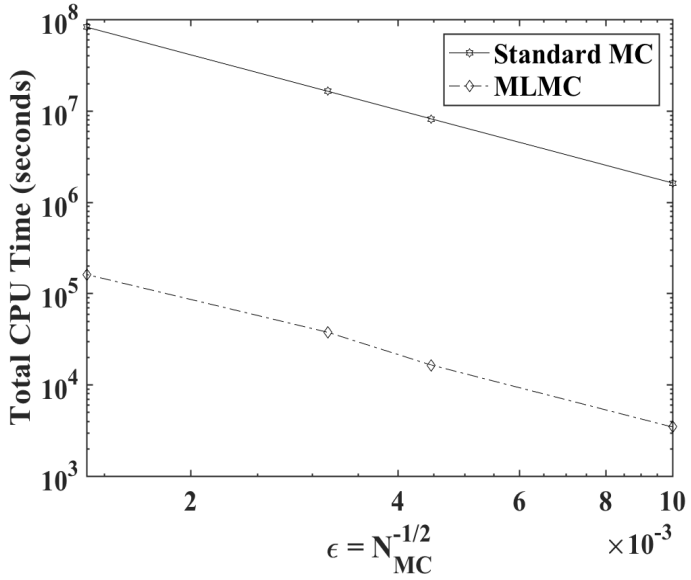



Figure 5.7: Total CPU time to simulate the expected value of the QoI using the MC and MLMC algorithms with $\epsilon = [N_{MC}]^{-1/2}$.

## 5.4 Conclusion

Throughout this chapter we have constructed and solved a stochastic physics based problem that extended the classic stationary shutter problem introduced by Moshinsky. Along with this, we also provided methodology that leads to efficient computation of the physical QoI. Although MLMC computes moments of the QoI in an extremely short amount of time, the question remains, will quasi-random points improve the results we already have in a significant way? To consider this question, we first refer to Figure 5.8. In this figure, it is obvious that quasi-random sampling can improve the convergence rate of MC. However, given the precision of our numerical

solve and the error obtained by MLMC, it is unnecessary to explore quasi-random sampling for this particular model. Although this is the case for this physics based model, quasi-random sampling can improve the convergence of more complex models, as we demonstrated in Chapter 4.

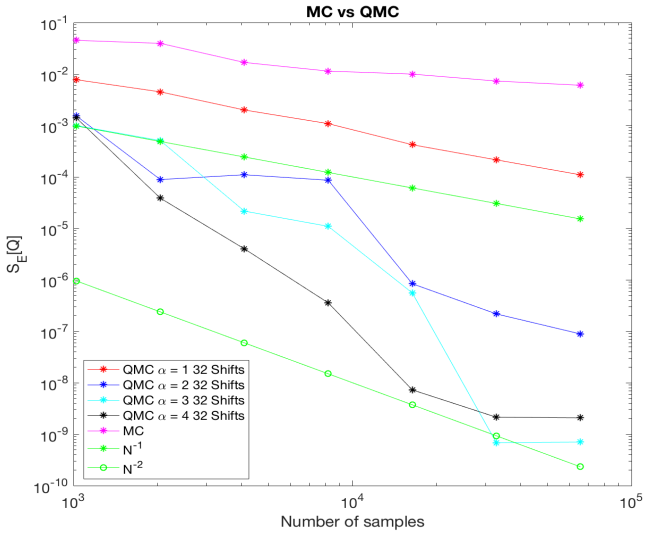



Figure 5.8: QMC results using 32 shifts.

# CHAPTER 6

# NEURAL NETWORK BASED COMPUTATIONAL METHODS FOR PDES

Numerical techniques such as finite difference methods [106], finite element methods [107], finite volume methods, spectral methods, and other state of the art techniques have been developed [108] for the solution of differential equations. These methods have shown to be extremely effective at solving complex differential equation models, in addition to having established convergence, consistency, and stability criteria. We have demonstrated such numerical techniques in the previous chapters, see for example Chapters 2, 4, and 5. In recent years, data-driven neural network (NN) approximation based solutions of PDEs have grown in popularity thanks to user friendly APIs such as Tensorflow and the additional benefit of distributed computing using Graphic Processing Units (GPUs). In this chapter, we explore NN based approximations of the PDEs considered in the previous chapters and access their performance.

One very promising NN technique for PDEs is that of Physics Informed NNs (PINNs), which has been shown to be an effective way to construct solutions to complex PDE models without requiring a substantial development time, if one takes advantage of well established NN APIs. PINNs achieve their approximation by incorporating the PDE and its boundary and initial conditions into the network's loss function. The idea of NNs that incorporate the full differential equation model into the loss function has been studied for several decades. For example, the work of Lee in [109] presents a neural algorithm for solving differential equations by representing the differential equation as finite difference equations and then uses a neural minimization to solve these finite difference equations. In addition to this work, another notable paper is that of [110], which is one of the first attempts to approximate PDEs with irregular boundaries using NNs by incorporating the model into the loss function as many modern PINNs do.

Thanks to this earlier work, this unification of mathematical models and NNs has been shown to reliably solve PDE-governed forward and inverse problems [51]. This innovative strategy has

numerous advantages such as efficiently learning through data-driven approximations and data-driven discovery of unknown quantities of PDEs [51], reduction in the amount of data needed to sufficiently describe observed phenomena [52], and utilization of NNs as memory efficient approximations of PDEs. This ability to model PDEs using PINNs is possible as NNs are universal function approximations [111]. This property in short says that NNs are capable of approximating a large class of functions to a sufficient degree of accuracy, provided an appropriate number of hidden units are available [111]. The efficiency of using PINNs as solutions to PDEs is made possible by automatic differentiation, which allows for accurate evaluation of derivatives of numeric functions expressed as computer programs [112]. Additionally, these PINN approximations are becoming more practical as the performance and efficiency of GPUs increases for tensor-specific operations.

Although PINNs have been used to solve a class of PDEs described on stationary domains, a well-known issue is that the PINN framework does not allow one (*a priori*) to obtain highly accurate (if not exact) fulfilment of boundary and initial conditions of the full space-time PDE models. This is due to the fact that the standard PINN loss function (involving three residuals which represent the PDE, boundary, and initial conditions) can produce an uneven focus on particular parts of the loss function. For example, the loss function may primarily try to optimize the PDE on the interior of the domain, rather than the initial or boundary conditions. To provide a more consistent minimization on each of the loss function's portions, it has been suggested in several articles [113–117] to apply a scaling parameter to the loss function components. Although useful, these particular approaches are not always adequate as the choice of the scaling parameter is problem specific and making it a learnable variable can introduce significant computation time or produce unsatisfactory results.

To combat this downside of PINNs, there have been attempts to construct a NN solution that automatically satisfies the boundary and initial conditions. The earliest of these is that of [118], which presents a method that constructs a trial solution of the differential equation as the sum of two parts. The first of these parts, *a priori*, satisfies the initial and/or boundary conditions and the

second part involves a NN that is trained using a loss function that incorporates the differential equation. Although this approach has shown some promise, it has only been applied to simple domain configurations. The recent approach outlined in [53] aims to incorporate the boundary and initial conditions into the PINN approximation such that the conditions are *a priori* satisfied and the method can be deployed on irregular shaped domains. Throughout this chapter, we will refer to this approach as a Physics Constrained NN (PCNN).

The ability to automatically satisfy the boundary and initial condition data can result in approximations that are more accurate than a standard PINN. Although this approach has tremendous upside, this approach has not been thoroughly explored. To the best of our knowledge, PCNNs have been considered only on stationary domains [53]. However, the approach has not been considered for a PDE defined within a moving domain, such as the DiT model governed by the Schrödinger that was numerically investigated in the last chapter. In this chapter, we consider how PCNNs can be applied to the class of moving domain Schrödinger models of Chapter 5 and compare them against the FEM performance and also generalize the model to two- and three-dimensions. The latter generalization is the key advantage of our proposed frequency-dependent PCNN for the moving domain model as it does not require substantial moving-mesh based complexities for generalizing the model (as in the FEM-based approximation). Before considering the moving domain model, in this chapter we develop details in several stages, including how variants of the PINN approach can be used to simulate the heat equation (which is crucial for simulating the A-C nonlinear PDE model investigated in Chapter 4) and also a nonlinear inverse problem associated with the A-C model.

To begin, in Section 6.1 we develop the NN approximation of interest, the multilayer perceptron (MLP), which is based on a nonlinear activation function acting on the affine form of the data-driven learning of dependent weight and bias parameters. Using the established MLP, in Section 6.2 we introduce the framework of a PINN using the heat equation and highlight the downsides and upsides of the standard PINN approach. Additionally, in this section we introduce a PINN with scaling in the loss function and demonstrate its usefulness over the standard PINN

approach. In Section 6.4 we establish the PCNN and show its advantages over both PINN methods using the 2D heat equation. Lastly, in Section 6.5 we develop a PCNN based approach for simulating a class of moving domain PDE models and showcase the method using the exact solution formulated in Section 5.2.2 for the moving domain Schrödinger model. Additionally, we show how the proposed method can be easily extended into 2D and 3D without making significant changes.

## 6.1 Multilayer Perceptron

Since their inception, NNs have taken on many forms. With the advent of powerful techniques in conjunction with specialized hardware (such as GPUs) their use has become one of the most sought after areas of research for both applied scientists and mathematicians alike. It is this drive that has brought to light many innovations such as feedforward NNs, Convolutional NNs, Recurrent NNs (RNNs), and Residual NNs (ResNets). These network types have proven to be invaluable in many areas of research, such as: image generation/classification, speech recognition, classification, etc. [119, 120]. In this chapter, we will be concerned with deep feedforward NNs, associated optimization-based back propagation and their application to NN based approximations of PDEs that constrain the optimization process. To begin, we will first develop the ideas behind the construction and training of a deep feedforward NN.

In a deep feedforward NN or multilayer perceptron (MLP), the network architecture is as depicted in Figure 6.1 (for further information about MLPs, see [121]). In an $L$-layer MLP we transform an input vector $\mathbf{x} \in \mathbb{R}^d$, with elements $x_i$ for $i = 1, \ldots, d$ ($d$ as the dimension of the input), into the output $\mathbf{y} \in \mathbb{R}^o$, with elements $y_i$ for $i = 1, \ldots, o$ ($o$ as the dimension of the output), using $L$ hidden layers. It is the MLP that defines the parameterized mapping $\mathcal{NN}(\mathbf{x}; \boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}^o$, where $\boldsymbol{\theta} \in \mathbb{R}^k$ is a vector with $k$ entries representing the weights and biases of the network. The weights and biases of the $\ell$-th layer are represented as a matrix $W^{[\ell]} \in \mathbb{R}^{m \times p}$ and vector $\mathbf{b}^{[\ell]} \in \mathbb{R}^p$, respectively, where $m$ is the number of nodes of the previous layer and $p$ is the number of nodes of the current layer for $\ell = 1, \ldots, L+1$. We denote the elements of $W^{[\ell]}$ as $w_{ij}^{[\ell]}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, p$,

and the elements of the vector $\mathbf{b}^{[\ell]}$ as $b_i^{[\ell]}$ for $i = 1, \ldots, p$. The $i$-th hidden layer is denoted as $\mathbf{h}^{[i]} \in \mathbb{R}^p$ for $i = 0, \ldots, L+1$, with elements $h_j^{[i]}$ for $j = 1, \ldots, p$, where the initial and final hidden layers retain the values $\mathbf{h}^{[0]} = \mathbf{x}$ and $\mathbf{h}^{[L+1]} = \mathbf{y}$, respectively. The hidden-layers for $i = 1, \ldots, L$ are a result of the activation function $\mathbf{f}(\cdot) : \mathbb{R}^p \to \mathbb{R}^p$, which performs element wise operations on a vector. It should be noted that in practice the activation functions can take on many forms. Some celebrated activation functions in the machine learning literature are the *tanh*, *sigmoid*, Rectified Linear Units (ReLUs), and swish functions, which are illustrated Figure 6.2.

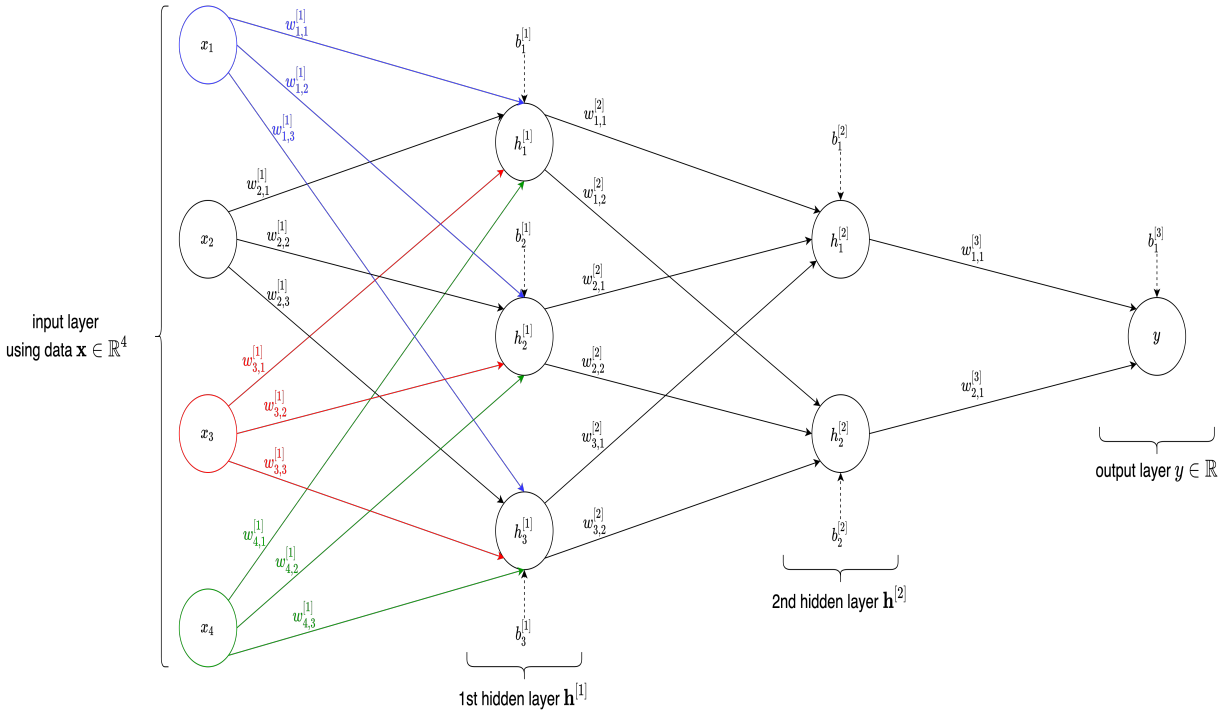



Figure 6.1: Visual representation of a 2-layer MLP with the 1st and 2nd hidden layer consisting of three and two nodes, respectively.

To provide even further clarity on the components of a MLP, we will consider the specific architecture of Figure 6.1, a 2-layer MLP, thus $L = 2$. From this figure, we see that the input layer has four data points providing $\mathbf{x} \in \mathbb{R}^4$. The MLP has the 1st hidden layer composed of three nodes and the 2nd hidden layer consist of three nodes. This provides the following weight matrices and

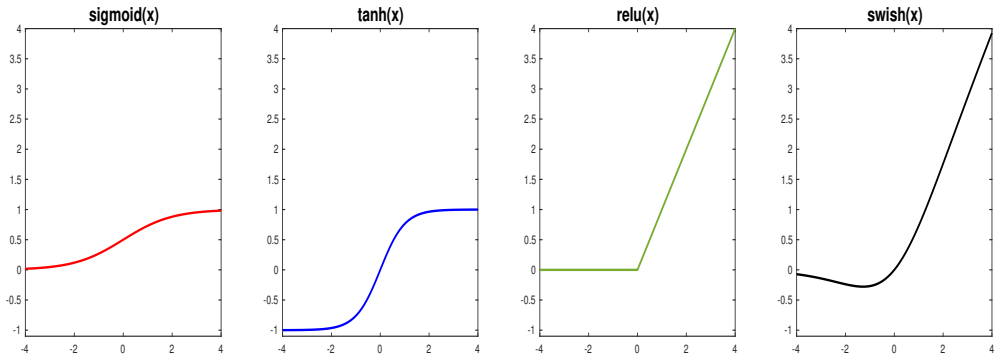



Figure 6.2: Visual representations of celebrated NN-based activation functions.

bias vectors:

$$W^{[1]} = \begin{bmatrix} w_{1,1}^{[1]} & w_{1,2}^{[1]} & w_{1,3}^{[1]} \\ w_{2,1}^{[1]} & w_{2,2}^{[1]} & w_{2,3}^{[1]} \\ w_{3,1}^{[1]} & w_{3,2}^{[1]} & w_{3,3}^{[1]} \\ w_{4,1}^{[1]} & w_{4,2}^{[1]} & w_{4,3}^{[1]} \end{bmatrix}, \quad W^{[2]} = \begin{bmatrix} w_{1,1}^{[2]} & w_{1,2}^{[2]} \\ w_{2,1}^{[2]} & w_{2,2}^{[2]} \\ w_{3,1}^{[2]} & w_{3,2}^{[2]} \end{bmatrix}, \quad W^{[3]} = \begin{bmatrix} w_{1,1}^{[3]} \\ w_{2,1}^{[3]} \end{bmatrix},$$

$$\mathbf{b}^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{bmatrix}, \quad \mathbf{b}^{[2]} = \begin{bmatrix} b_1^{[2]} \\ b_2^{[2]} \end{bmatrix}, \quad \mathbf{b}^{[3]} = \begin{bmatrix} b_1^{[3]} \end{bmatrix}.$$

Once we have defined the weights and biases, we can use the activation function $\mathbf{f}(\cdot)$ to compute the hidden layers. For example, if we let $\mathbf{f}(\cdot)$ be defined by **tanh** and let the input be $\mathbf{z} \in \mathbb{R}^3$ then

$$\mathbf{f}(\mathbf{z}) = \mathbf{tanh}(\mathbf{z}) = \begin{bmatrix} tanh(z_1) \\ tanh(z_2) \\ tanh(z_3) \end{bmatrix}.$$

Using this idea, we can compute the first hidden layer value, $\mathbf{h}^{[1]} \in \mathbb{R}^3$. Thus, we have the following form for the first hidden layer values:

$$\mathbf{h}^{[1]} = \mathbf{f}(W^{[1]T}\mathbf{x} + \mathbf{b}^{[1]}).$$

In a similar fashion, to compute $\mathbf{h}^{[2]} \in \mathbb{R}^2$, we propagate the first hidden layer values forward using the corresponding weight matrix and vector of biases:

$$\mathbf{h}^{[2]} = \mathbf{f}(W^{[2]T}\mathbf{h}^{[1]} + \mathbf{b}^{[2]}).$$

For the MLP in Figure 6.1 there are only two hidden layers, so we can then compute the output of the network as follows:

$$y = \mathbf{f}(W^{[3]T}\mathbf{h}^{[2]} + \mathbf{b}^{[3]}).$$

It should be noted that the output of the network can be obtained via a linear activation function. Thus, an alternative output could be as follows:

$$y = W^{[3]T}\mathbf{h}^{[2]} + \mathbf{b}^{[3]}.$$

From this architecture we can denote the 2-Layer MLP as the parametrized mapping $\mathcal{NN}(\mathbf{x};\boldsymbol{\theta})$, where $\boldsymbol{\theta} = [w^{[1]}_{1,1}, w^{[1]}_{1,2}, w^{[1]}_{1,3}, w^{[1]}_{2,1}, w^{[1]}_{2,2}, w^{[1]}_{2,3}, w^{[1]}_{3,1}, w^{[1]}_{3,2}, w^{[1]}_{3,3}, w^{[1]}_{4,1}, w^{[1]}_{4,2}, w^{[1]}_{4,3}, w^{[2]}_{1,1}, w^{[2]}_{1,2}, w^{[2]}_{2,1}, w^{[2]}_{2,2}, w^{[2]}_{3,1}, w^{[2]}_{3,2}, w^{[3]}_{1,1}, w^{[3]}_{2,1}, b^{[1]}_1, b^{[1]}_2, b^{[1]}_3, b^{[2]}_1, b^{[2]}_2, b^{[3]}_1]^T \in \mathbb{R}^{26}$.

### 6.1.1 MLP paramter training

As MLPs are primarily data based approximations, once the architecture has been established, one can then update $\boldsymbol{\theta}$ via optimization such that the MLP output more closely reflects the data through an approximate minimization process. The first step in this process is taking the gradient of the loss function w.r.t. $\boldsymbol{\theta}$. The parameterized loss function, denoted as $\mathcal{L}(\boldsymbol{\theta})$ represents the difference between the MLP approximation and the desired constraints such as the data or PDE

systems. For minimization of the loss function, in standard update methods, this action is completed using the method known as backpropagation and for related details we refer to [122]. The simplest approach for updating $\boldsymbol{\theta}$ for an MLP is to take the gradient of the loss function w.r.t. $\boldsymbol{\theta}$ and move the current values of $\boldsymbol{\theta}$ in the direction opposite of this gradient. This produces the very well-known approach of gradient descent and results in a stepping direction that often leads to the local minimum of the loss function. In practice, the loss function can take on various forms. In the case of PINNs, the loss function is composed of residuals arising from PDE, boundary, and initial condition comparisons, as we will see in Section 6.2. One of the simplest loss functions is motivated by the least-squares type residual and is provided in (6.1), where $M$ is the number of data points and $y_{pred}(x_i;\boldsymbol{\theta})$ is the predicted $y$ value for a given data point $x_i$ provided by the MLP.

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M}(y(x_i) - y_{pred}(x_i;\boldsymbol{\theta}))^2 \tag{6.1}$$

To provide some clarity on updating $\boldsymbol{\theta}$ we will showcase the update using the Adam optimization routine. The Adaptive Moment Estimation or Adam algorithm is an algorithm for first-order gradient-based optimization of stochastic objective functions and is based on adaptive estimates of lower-order moments [123]. The algorithm updates $g_{s,i}$, the partial derivative of the objective function w.r.t. the parameter $\theta_i$ at step $s$, by forming the first and second moments of the gradients $\mathbf{m}_s \in \mathbb{R}^{|\boldsymbol{\theta}|}$ and $\mathbf{v}_s \in \mathbb{R}^{|\boldsymbol{\theta}|}$, respectively as in (6.2).

$$\begin{aligned}\mathbf{m}_s &= \beta_1\mathbf{m}_{s-1} + (1-\beta_1)\mathbf{g}_s \\ \mathbf{v}_s &= \beta_2\mathbf{v}_{s-1} + (1-\beta_2)\mathbf{g}_s^2\end{aligned} \tag{6.2}$$

Where $\mathbf{m}_0 = 0 = \mathbf{v}_0$ and the values for the coefficients $\beta_1$ and $\beta_2$ are predetermined. In Tensorflow the default values for the coefficients $\beta_1$ and $\beta_2$ are set to 0.9 and 0.999, respectively. To counteract biases that are observed, the first and second moments are then corrected as in (6.3).

$$\begin{aligned}\hat{\mathbf{m}}_s &= \frac{\mathbf{m}_s}{1-\beta_1} \\ \hat{\mathbf{v}}_s &= \frac{\mathbf{v}_s}{1-\beta_2}\end{aligned} \tag{6.3}$$

Lastly, Adam updates the parameter $\boldsymbol{\theta}$ by using the update rule (6.4), where $\eta$ is a user prescribed learning rate and $\epsilon$ is a smoothing term that prevents division by zero, which is typically set to $10^{-8}$.

$$\boldsymbol{\theta}_{s+1} = \boldsymbol{\theta}_s - \frac{\eta}{\sqrt{\hat{\mathbf{v}}_s} + \epsilon}\hat{\mathbf{m}}_s \tag{6.4}$$

In standard NN nomenclature, if the full data set is used during the update process to inform the change in the parameter $\boldsymbol{\theta}$, then the act of going from $\boldsymbol{\theta}_s$ to $\boldsymbol{\theta}_{s+1}$ is termed an epoch.
For many of the PINNs we consider, we will often opt to instead use the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) optimizer [124] to update $\boldsymbol{\theta}$. The L-BFGS optimizer is a limited memory quasi-Newton method that estimates the curvature of the parameter space via an approximation of the Hessian. In the machine learning community, it is often observed that L-BFGS performs better than most optimization routines when the data being considered is relatively small. This attribute makes it attractive when considering PINNs as one often wants to use sparse data to learn the MLP approximation.

### 6.1.2 Linear Regression Example

To concretely establish the framework of the MLP, we will consider a standard linear regression problem. In this example, we will be concerned with only a single input parameter $x \in \mathbb{R}$. In this scenario, we will use the standard 1-layer and 1-node MLP provided by Figure 6.3, where we use the identity activation function $f(x) = x$ to obtain the first layer. To conduct linear regression, we

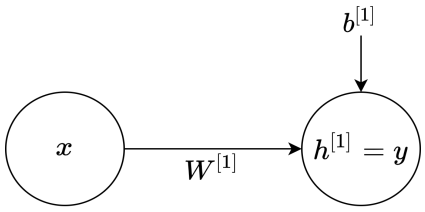



Figure 6.3: 1-layer MLP architecture for the linear regression example.

consider data produced by a line defined on $[0, 3]$ with slope $m$ and intercept $b$. We create the data using $M$ equally spaced points $x_i$, $i = 1, \ldots, M$ with $x_1 = 0$ and $x_M = 3$. For each of these points

we generate $y(x_i)$, $i = 1, \ldots, M$, by adding noise to the line, where $m = 2$, $b = 0.9$, and the noise is generated from $\mathcal{N}(0, 0.3)$. Using these specifications and $M = 120$, we obtain the data in Figure 6.4.

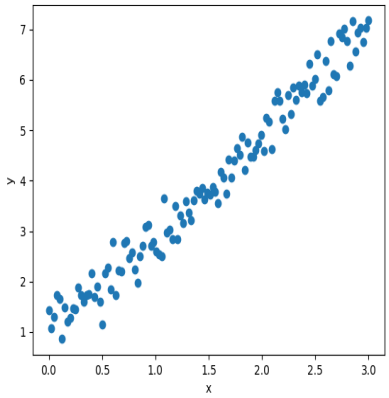



Figure 6.4: Generated data for linear regression example.

To conduct linear regression we must now specify the loss function we wish to minimize. The loss function used will be motivated by the standard least-squares residual as defined in (6.1), where again $M$ is the number of data points and $y_{pred}(x_i; \boldsymbol{\theta})$ is the predicted $y$ value for a given data point $x_i$ provided by the MLP. In addition to this, we initialize the weights and biases for the 1-layer MLP by randomly choosing values between 0 and 1. For these specifications, we complete 1000 epochs of training for the 1-layer MLP using the Adam optimizer with learning rate 1e-3, $\beta_1 = 0.9, \beta_2 = 0.999$, and no weight decay. This provides the results in Figure 6.5, which show that the 1-layer MLP is able to accurately predict the true underlying exact solution.

## 6.2 Physics Informed Neural Networks

Although utilization of NNs as classification models for observed data have been a powerful technique that has proven to be extremely useful in various applications, modifications are required in the loss function to incorporate background physics of the desired process governed by a PDE. Simple data-only constrained NN approximations do not in general satisfy the underlying PDE even to 1% accuracy, and out interest in this chapter is to achieve NN-based

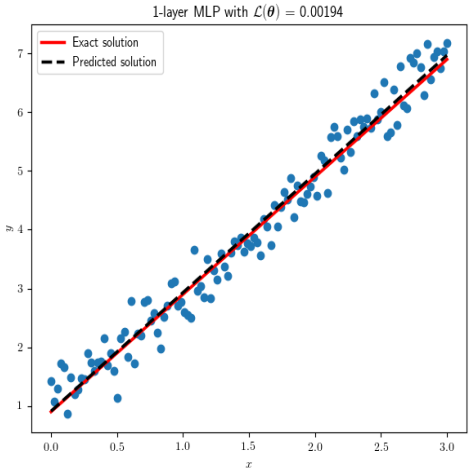



Figure 6.5: Linear regression results using a 1-layer MLP.

approximations that satisfy much higher accuracy even for complex processes. It is this area that the PINN and related variant based methodologies address. In this section, we establish the standard definition of a PINN as presented in [51] and then continue by demonstrating the PINN solution and also highlighting its weaknesses and how to improve its approximation based on variations of the approach. In this chapter, we will be concerned with PINN-based solutions to PDEs. Thus, we are particularly interested in PDEs of the form (6.5):

$$\mathcal{J}(\mathbf{u}(\mathbf{x},t)) = F(\mathbf{x},t), \quad \mathbf{x} \in \Omega, \quad t \in (0,T] \tag{6.5a}$$

$$\mathcal{BC}(\mathbf{u}(\mathbf{x},t)) = \mathcal{B}(\mathbf{x},t), \quad \mathbf{x} \in \partial\Omega, \quad t \in (0,T] \tag{6.5b}$$

$$\mathcal{IC}(\mathbf{u}(\mathbf{x},0)) = \mathcal{I}_0(\mathbf{x}), \quad \mathbf{x} \in \bar{\Omega}, \quad t = 0. \tag{6.5c}$$

Where $\mathbf{u}(\mathbf{x},t)$ is the unknown solution of the PDE system governed by a space-time second-order differential operator $\mathcal{J}(\cdot)$, a boundary operator $\mathcal{BC}$, an initial condition operator $\mathcal{IC}$ and a forcing function $F$. For example: With $\Delta$ denoting the second-order Laplace operator, $\mathcal{J} = \frac{\partial}{\partial t} - \alpha\Delta$ is the differential operator governing the heat equation with diffusion coefficient $\alpha$. For Dirichlet and Neumann boundary conditions, $\mathcal{BC}$ is, respectively, the identity and $\mathbf{n} \cdot \nabla$ operator (with outward unit normal $\mathbf{n}$) and various combinations of these can be used for other types of boundary

conditions, such as the Robin case or mixed boundary conditions. If $\mathcal{J}$ contains only a first order operator in time, then $\mathcal{IC}$ is an identity operator and in the second-order derivative in the time variable occuring in $\mathcal{J}$, typically, an additional first-order derivative condition at the initial state is added to the $\mathcal{IC}$ operator. It is standard to allow for non-zero initial conditions and this facilities, without loss of generality, taking the boundary condition to be zero. For all our examples in this chapter, we take $\mathcal{BC}$ to denote the homogeneous Dirichlet boundary condition, and design reference and NN-based solutions accordingly.

In a PINN, the goal is to construct a solution of (6.5) by approximating $\mathbf{u}(\mathbf{x}, t)$ using an $L$-layer MLP parameterized mapping $\mathcal{NN}(\mathbf{r}; \boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$, where $d$ is the dimension of the input, as presented in Section 6.1. For example, if we are considering a time-dependent PDE with one spatial dimension, then we have $d = 2$, in 2D we have $d = 3$. Constructing this approximation is accomplished by minimizing a loss function that is composed of (6.5a - 6.5c). The most common form of this loss function is that of (6.6), where the collocation points are defined as follows: $\mathbf{x}_{F,i} \in \Omega$ and $t_{F,i} \in (0, T]$ for $i = 1, \ldots, N_F$, $\mathbf{r}_{F,i} = [\mathbf{x}_{F,i}, t_{F,i}]^T$, $\mathbf{x}_{BC,i} \in \partial\Omega$ and $t_{BC,i} \in (0, T]$ for $i = 1, \ldots, N_{BC}$, $\mathbf{r}_{BC,i} = [\mathbf{x}_{BC,i}, t_{BC,i}]^T$, $\mathbf{x}_{0,i} \in \partial\bar{\Omega}$ and $t_{0,i} = 0$ for $i = 1, \ldots, N_0$, $\mathbf{r}_{0,i} = [\mathbf{x}_{0,i}, t_{0,i}]^T$. For these collocation points, the notation $\mathbf{r} = [\mathbf{x}, t]^T$ denotes the construction of the vector $\mathbf{r}$ by concatenating the scalar $t$ onto the end of the vector (or a scaler for d=1) $\mathbf{x}$.

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N_F} \sum_{i=1}^{N_F} \Big( \mathcal{J}(\mathcal{NN}(\mathbf{r}_{F,i}; \boldsymbol{\theta})) - F(\mathbf{r}_{F,i}) \Big)^2 \tag{6.6a}$$

$$+ \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \Big( \mathcal{NN}(\mathbf{r}_{BC,i}; \boldsymbol{\theta}) - \mathcal{B}(\mathbf{r}_{BC,i}) \Big)^2 \tag{6.6b}$$

$$+ \frac{1}{N_0} \sum_{i=1}^{N_0} \Big( \mathcal{NN}(\mathbf{r}_{0,i}; \boldsymbol{\theta}) - \mathcal{I}_0(\mathbf{r}_{0,i}) \Big)^2. \tag{6.6c}$$

There exist several differing strategies for constructing the collocation points. One method that has shown the most promise is that of constructing the collocation points using Latin Hypercube Sampling (LHS). LHS is a technique developed in 1979 [125] and is a strategic way to sample a given interval that has impressive computational advantages when compared to simple uniform

random sampling. The principle is to independently stratify each of the $d$ input dimensions $x = (x_1, x_2, \ldots, x_d)^T$ into $N$ equi-possible intervals of probability $1/N$, $N$ being the number of samples. For this chapter, we choose to construct our collocation points using the LHS technique. To do this, we utilize the Python package PyDOE2 [126].

The proposed PINN approach presents many advantages over standard numerical techniques such as the finite difference or time-stepping finite element method. Once the loss function (6.6) has been sufficiently minimized, the approximation $\mathcal{NN}(\mathbf{r}; \boldsymbol{\theta})$ is a closed form continuous approximation. Lastly, this method is easily parallelizable and a significant amount of the computation can be deployed on GPUs using common APIs such as Tensorflow, in comparison to other numerical techniques that require a significant amount of attention to perform a similar parallelization.

### 6.3 Performance of PINNs and Variant Approximations of the Heat Equation

In the subsequent sections we will compare the standard PINN, a PINN with scaling, and a PCNN. To compare these three approaches, we consider the heat equation on a unit square, with diffusion coefficient $\alpha = 0.5$ and for precise performance comparison we choose the homogeneous Dirichlet boundary condition and the initial state in such a way that the exact solution of the heat equation model is:

$$u(x, y, t) = sin(\pi x) sin(\pi y) e^{-\pi^2 t} \quad (x, y) \in [0, 1] \times [0, 1] \quad \text{for all} \quad t \in [0, 0.5]. \tag{6.7}$$

The main problem of interest is NN approximations to simulate the heat flow in the square plate. More precisely, how the NN-based predicted solution $u_{pred}$ compares with the above exact solution at any set of $(x, y, t)$, where the points need not be part of the collocation points used in the NN's construction (see Figure 6.6 for an example set of collocation points).

We construct the maximum absolute error in the NN-based approximations by first considering the absolute difference (6.8) at $N^2$ uniformly determined spatial points $(x_m, y_n)$ (with mesh-width $h = \frac{1}{N-1}$), for $m, n = 0, \ldots, N-1$ and $K$ uniform temporal mesh points $t_k$ with width $\Delta t = \frac{T}{K-1}$, for $k = 0, \ldots, K-1$. Similar to the FEM solutions, the NN approximation $u_{pred}$ is then obtained at

these points. Accordingly, we can compute the errors

$$Err(m,n,k) = |u_{pred}(x_m, y_n, t_k) - u(x_m, y_n, t_k)|, \tag{6.8}$$

$$Err_{max,t_k} = max\{Err(m,n,k) : m,n = 0,\ldots,N-1\}, \tag{6.9}$$

and hence the maximum absolute error:

$$Err_{max} = max\{Err_{max,t_k} : k = 0,\ldots,K-1\}. \tag{6.10}$$

### 6.3.1 PINN Solution of the Heat Equation

In this section, we consider a standard PINN approach to the above described heat equation. To provide the NN solution, we approximate $u(x,y,t)$ with a parameterized MLP $\mathcal{NN}(\mathbf{r};\boldsymbol{\theta}) : \mathbb{R}^3 \rightarrow \mathbb{R}$ that consists of 8 hidden-layers each with 60 nodes that contain biases and activation functions specified by $f(\mathbf{r}) = tanh(\mathbf{r})$, where $\mathbf{r} = [x,y,t]^T$. The output node of this MLP will contain one node with a bias and a linear activation function. We also initialize all biases to zero and use the well-known Xavier initialization for the weights [127].

To train the MLP we use the loss function specified by (6.11), making $\mathcal{NN}(\mathbf{r};\boldsymbol{\theta})$ a PINN.

$$\mathcal{L}(\boldsymbol{\theta}) = MSE_F(\boldsymbol{\theta}) + MSE_0(\boldsymbol{\theta}) + MSE_{BC}(\boldsymbol{\theta}), \tag{6.11}$$

The terms of (6.11) are provided by (6.12) where $\mathcal{J}(\mathcal{NN}(\mathbf{r};\boldsymbol{\theta}))$ is the differential operator specified by (6.13). Equation (6.12a) ensures that the main PDE of the heat equation is satisfied by using the collocation points $x_{F,i}, y_{F,i}, t_{F,i}$ for $i = 1,\ldots,N_F$, where $\mathbf{r}_{F,i} = [x_{F,i}, y_{F,i}, t_{F,i}]^T$. Equation (6.12c) makes sure that the initial condition is satisfied by using initial condition points $x_{0,i}, y_{0,i}$, for $i = 1,\ldots,N_0$, where $\mathbf{r}_{0,i} = [x_{0,i}, y_{0,i}, 0]^T$. Lastly, equation (6.12b) makes sure that the boundary conditions are satisfied by using boundary condition points $x_{BC,i}, y_{BC,i}, t_{BC,i}$ for $i = 1,\ldots,N_{BC}$, where $\mathbf{r}_{BC,i} = [x_{BC,i}, y_{BC,i}, t_{BC,i}]^T$.

$$MSE_F = \frac{1}{N_F}\sum_{i=1}^{N_F}\left(\mathcal{J}(\mathcal{NN}(\mathbf{r}_{F,i};\boldsymbol{\theta}))\right)^2 \tag{6.12a}$$

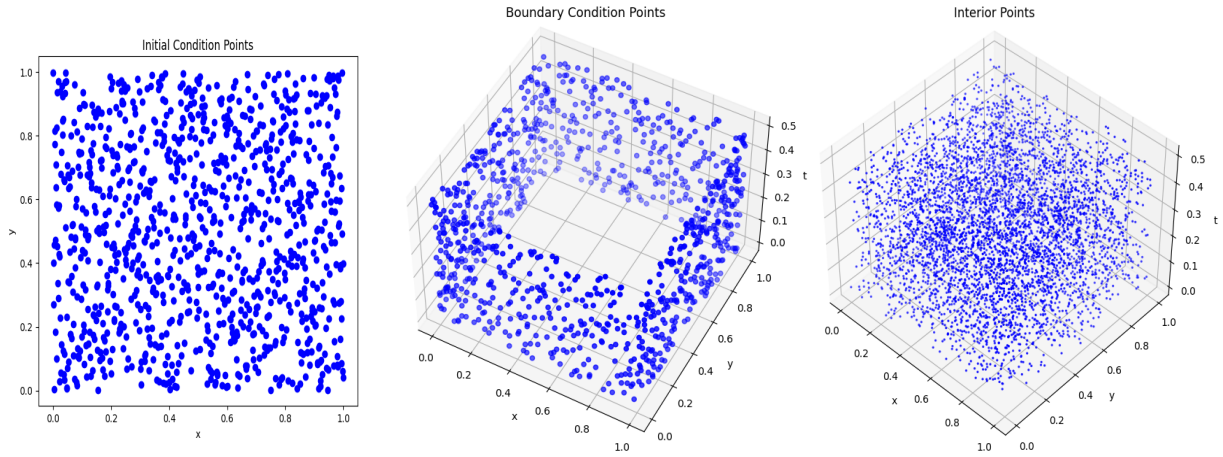
$$MSE_{BC} = \frac{1}{N_{BC}}\sum_{i=1}^{N_{BC}}\left(\mathcal{NN}(\mathbf{r}_{BC,i};\boldsymbol{\theta}) - u(\mathbf{r}_{BC,i})\right)^2, \tag{6.12b}$$

$$MSE_0 = \frac{1}{N_0}\sum_{i=1}^{N_0}\left(\mathcal{NN}(\mathbf{r}_{0,i};\boldsymbol{\theta}) - u(\mathbf{r}_{0,i})\right)^2. \tag{6.12c}$$

$$\mathcal{J}(\mathcal{NN}(\mathbf{r};\boldsymbol{\theta})) = \frac{\partial \mathcal{NN}(\mathbf{r};\boldsymbol{\theta})}{\partial t} - 0.5\Delta\mathcal{NN}(\mathbf{r};\boldsymbol{\theta}) \tag{6.13}$$

To obtain all collocation points, we utilize LHS. Specifically, we generate the initial condition points $x_{0,i}, y_{0,i}$ using LHS with input dimension of 2 and $N_0 = 1,000$ and set the time points to zero. The initial condition points are visually represented in panel (a) of Figure 6.6. The boundary condition points $x_{BC,i}, y_{BC,i}, t_{BC,i}$ are constructed by using LHS with input dimension of 3 and 250 samples for each side of the plate and then combining all points produced into a matrix of dimension $3 \times N_{BC}$, where $N_{BC} = 1,000$. These boundary condition points are shown in panel (b) of Figure 6.6. Lastly, the interior points are constructed by first creating 1,000 evenly spaced points between 0 and $T$, then for each of these time points $x_{F,i}, y_{F,i}$ are generated using LHS with input dimension of 2 and 50 samples. We then let $t_{F,i}$ be a vector of length 50 fixed to the corresponding time point. This process produces the points $x_{F,i}, y_{F,i}, t_{F,i}$ where $N_F = 50,000$. These points are presented visually in panel (c) of Figure 6.6 where, for better visualization purposes, we use 100 evenly spaced time points instead of the stated 1,000.

Using the framework we have specified for the MLP, the weights $\boldsymbol{\theta}$ are found by minimizing the loss function (6.11) using the L-BFGS optimizer with 20,000 iterations. After completing this process we consider the maximum absolute difference (6.8) for each $k$ with $N = K = 1,000$. This produces Figure 6.7, which clearly shows that a significant amount of error is produced by the initial condition and the error is inconsistent throughout time. Additionally, we obtain the loss and error values in Table 6.1.




(a) (b) (c)

Figure 6.6: Collocation points for the 2D heat equation. In the left panel (a) we provide the initial condition points, the middle panel (b) displays the boundary condition points, and the right panel (c) provides a visualization of the interior points using 100 evenly spaced time points.


Figure 6.7: Maximum PINN error, $Err_{max,t_k}$, for $k = 0, \ldots, 999$.

Table 6.1: PINN solution values for the 2D heat equation.

| Method | $MSE_F$ | $MSE_{BC}$ | $MSE_0$ | $Err_{max}$ |
|---|---|---|---|---|
| PINN | 9.1504e-7 | 8.1439e-7 | 2.2278e-7 | 2.8909e-3 |

### 6.3.2 PINN with Scaling

From Figure 6.7, we found that the maximum error is large at the initial time and inconsistent throughout time. These attributes are primarily caused by the initial and boundary conditions. To overcome this issue, a standard generalization of the PINN is to add a weight factor to the individual terms of (6.11) following similar ideas employed, say, in Collocation Least-Squares Finite Element Methods, see for example [128]. In addition to this approach, there have been several other choices in scaling for the PINN framework, see for example, [113–117]. For this section, we consider the weighting of the loss function as specified by (6.14), where $\lambda$ is a hyper-parameter to be determined.

$$\mathcal{L}(\boldsymbol{\theta}) = MSE_F(\boldsymbol{\theta}) + \lambda MSE_0(\boldsymbol{\theta}) + \lambda MSE_{BC}(\boldsymbol{\theta}) \tag{6.14}$$

To showcase the effectiveness of the loss function (6.14), we will again consider the 2D heat equation as specified in Section 6.3. For a fair comparison against the unmodified loss function (6.11) we will use the MLP architecture and the choice of collocation points provided in Section 6.3.1. Similarly, we minimize the loss function (6.14) using the L-BFGS optimizer with 20,000 iterations. To find the hyper-parameter $\lambda$ we considered the values $\lambda = 10, 100, 1000$, and 10000. It was found that $\lambda = 1000$ produced the most appropriate attention to the boundary and initial conditions.

The results of the scaled PINN approach with loss (6.14) and $\lambda = 1000$ are provided in Figure 6.8 and Table 6.2. From these results, one can see that the Scaled PINN has a significant decrease in the maximum error as well as a more consistent maximum error throughout time. This is the result of minimizing $MSE_{BC}$ and $MSE_0$ further as confirmed by Table 6.2. For a comparison of a PINN with an alternatively scaled PINN for a nonlinear PDE, see Appendix C. This Appendix section additionally shows how scaling can lead to a better extrapolation of the predicted solution outside of the sampled time points. To see an example where the scaled PINN solution can be used as a memory efficient approximation of a PDE, see Appendix D

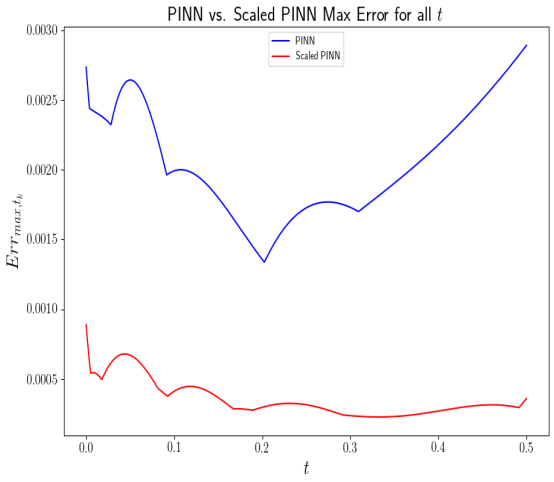



Figure 6.8: Maximum error, $Err_{max,t_k}$, with $k = 0, \ldots, 999$, for PINN vs. Scaled PINN.

Table 6.2: PINN solution values for the 2D heat equation.

| Method | $MSE_F$ | $MSE_{BC}$ | $MSE_0$ | $Err_{max}$ |
|---|---|---|---|---|
| PINN | 9.1504e-7 | 8.1439e-7 | 2.2278e-7 | 2.8909e-3 |
| Scaled PINN | 6.1049e-5 | 1.4135e-8 | 9.3932e-9 | 8.9010e-4 |

## 6.4 PCNN Approximations

In Section 6.3.1 we observed that the errors at the boundaries and initial condition are large for the PINN. To counteract this, we applied a scaling parameter $\lambda$ to the loss function as in (6.14). Although this technique can prove to be effective by creating a more consistent error throughout the solution, there still remains a significant amount of error at the boundaries and initial condition. Additionally, choosing the scaling parameter may require testing several different $\lambda$ values until one finds the parameter that is most appropriate. This is due to the fact that choosing the scaling parameter to be a learnable variable often results in poor approximations of the solution, especially for complex PDE models. In the recent work of [129], the approach developed attempts to adjust this scaling parameter adaptively with a decent amount of success. However, as stated in this work, this continues to be an open problem that requires further study, especially in relation to how this adaptivity should be changed or tailored for specific optimization routines.

This difficulty in choosing the scaling parameter and the persistence of large error at the boundaries and initial condition becomes unmanageable when considering complex PDEs and often leads to incorrect solutions. For this reason, we would like to consider a PINN variant enhanced with *a priori* exact (or approximate, but highly accurate) representations of the boundary and initial conditions, before seeking an NN approximation to appropriately satisfy the PDE in the interior. To accomplish this, we follow the ideas in [53], which were first established in [118]. For the sake of being succinct, we will call the constructed approach a Physics Constrained NN (PCNN).

For a PCNN, we are interested in PDEs of the form (6.5). To create an approximation of $\mathbf{u}(\mathbf{x}, t)$ that automatically satisfies the boundary and initial condition data, one can choose the PCNN approximation to be of the form (6.15). Where $\odot$ represents element-wise multiplication.

$$\mathbf{u}(\mathbf{x}, t) \approx \mathcal{U}(\mathbf{x}, t) = \mathcal{U}_{IB}(\mathbf{x}, t) + \mathcal{D}(\mathbf{x}, t) \odot \mathcal{U}_G(\mathbf{x}, t) \tag{6.15}$$

In this approximation, $\mathcal{U}_{IB}$ is a function that satisfies the initial conditions on $\bar{\Omega} \times \{t = 0\}$ and the boundary conditions on $\partial\Omega$, as provided in (6.16), where $\dot{\mathcal{U}}_{IB}(\mathbf{x}, t)$ represents the temporal derivative of $\mathcal{U}_{IB}$. It should be noted that $\mathcal{U}_{IB}$ on the remaining portions of the domain must take on any nonzero value for training to be effective.

$$\mathcal{U}_{IB} = \begin{cases} \mathcal{B}(\mathbf{x}, t), & \mathbf{x} \in \partial\Omega, \;\; t \in (0, T] \\ \mathcal{I}_0(\mathbf{x}), & \mathbf{x} \in \bar{\Omega}, \;\; t = 0. \end{cases} \tag{6.16}$$

In (6.15), $\mathcal{D}(\mathbf{x}, t)$ is a distance function representing the distance to the boundary and initial condition data, and is zero where the boundaries and initial conditions are, as given in (6.17). It should be noted that if one wants to enforce the initial conditions for a second-order in time operator, then $\mathcal{D}(\mathbf{x}, t)$ must be modified. For example, if we want the derivative of the initial state to be zero, then $\dot{\mathcal{D}}(\mathbf{x}, t)$ must be set to zero for $(\mathbf{x}, t) \in (\bar{\Omega} \times \{t = 0\})$. This is due to the fact that the

temporal derivative of the approximation takes on the form (6.18).

$$\mathcal{D}(\mathbf{x},t) = \begin{cases} 0, & \text{for}(\mathbf{x},t) \in (\partial\Omega \times (0,T]) \cup (\bar{\Omega} \times \{t=0\}) \\ \text{nonzero}, & \text{otherwise} \end{cases} \tag{6.17}$$

$$\dot{\mathcal{U}}(\mathbf{x},t) = \dot{\mathcal{U}}_{IB}(\mathbf{x},t) + \dot{\mathcal{D}}(\mathbf{x},t) \odot \mathcal{U}_G(\mathbf{x},t) + \mathcal{D}(\mathbf{x},t) \odot \dot{\mathcal{U}}_G(\mathbf{x},t) \tag{6.18}$$

Finally, $\mathcal{U}_G(\mathbf{x},t)$ is a NN approximation to the PDE system.

Now that we have our PCNN approximation, we now need to describe how this approximation is learned. Although not necessary, in practice one can represent $\mathcal{U}_{IB}(\mathbf{x},t)$ and $\mathcal{D}(\mathbf{x},t)$ as MLPs and pre-train them before considering the full PCNN minimization. To construct an MLP approximation of $\mathcal{U}_{IB}(\mathbf{x},t)$, one simply constructs a loss function composed of the terms (6.6b - 6.6c), where $\mathcal{NN}(\mathbf{r};\boldsymbol{\theta})$ is a MLP approximating $\mathcal{U}_{IB}(\mathbf{x},t)$. To create a MLP approximation of $\mathcal{D}(\mathbf{x},t)$ one needs to first create collocation points, say $\mathbf{r}_D$, throughout the entire domain, which includes the time points, the simplest way to do this is by using LHS. Once the data points are created, the minimum distance between all of these points needs to be computed. Although this task can be compute intensive, there exist efficient algorithms to compute such distances in several dimensions. See for example the SciPy function *cdist*, which efficiently computes several different types of distances between each pair of the two collections of inputs [130]. Once the minimum distance values are calculated, say $\mathcal{D}_{data}$, one can construct the approximation of $\mathcal{D}(\mathbf{x},t)$ using the loss function (6.19), where the inclusion of the term (6.19b) is included if one wants the derivative of the initial state to be zero.

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N_D}\sum_{i=1}^{N_D}\left(\mathcal{D}(\mathbf{r}_{D,i};\boldsymbol{\theta}) - \mathcal{D}_{data}(\mathbf{r}_{D,i})\right)^2 \tag{6.19a}$$

$$+ \frac{1}{N_D}\sum_{i=1}^{N_D}\left(\dot{\mathcal{D}}(\mathbf{r}_{D,i};\boldsymbol{\theta})\right)^2 \tag{6.19b}$$

Although approximating $\mathcal{U}_{IB}$ and $\mathcal{D}$ with MLPs is sufficient for some domain configurations and data, one can construct exact functional representations. When possible, this is the preferred option as it reduces compute time. Once functional representations of $\mathcal{U}_{IB}$ and $\mathcal{D}$ have been

established, the last task in training the PCNN is constructing the approximation of $\mathcal{U}_G(\mathbf{x}, t)$. One can accomplish this by minimizing the loss function given in (6.20), where $\mathbf{x}_{F,i}$ and $t_{F,i}$ are the spatial and temporal collocation points, respectively over $\Omega \times (0, T]$, and $\mathbf{r}_{F,i} = [\mathbf{x}_{F,i}, t_{F,i}]^T$.

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N_F} \sum_{i=1}^{N_F} \Big( \mathcal{J}(\mathcal{U}(\mathbf{r}_{F,i}; \boldsymbol{\theta})) - F(\mathbf{r}_{F,i}) \Big)^2 \tag{6.20}$$

It should be noted that one can sample these points over $\bar{\Omega} \times [0, T]$ if the initial and boundary conditions satisfy the forcing function $F(\mathbf{x}, t)$. As in a PINN, these collocation points are often constructed using LHS points. Additionally, for the PCNN we often choose to employ the L-BFGS optimizer to minimize (6.20).

### 6.4.1 PCNN Solution of the Heat Equation

To showcase the effectiveness of this approach in comparison to the standard PINN and the scaled PINN, we will consider the 2D heat equation described in Section 6.3. To construct the PCNN approximation (6.15), we let $\mathcal{D}(x, t) = tx(1 - x)y(1 - y)$ and $\mathcal{U}_{IB} = sin(\pi x)sin(\pi y)$. By inspection, one can see that $\mathcal{D}(x, t)$ is zero at the boundaries and initial condition, and $\mathcal{U}_{IB}$ satisfies both the initial and boundary conditions. We then approximate $\mathcal{U}_G$ with the parameterized MLP $\mathcal{NN}(\mathbf{r}; \boldsymbol{\theta}) : \mathbb{R}^3 \to \mathbb{R}$.

As in the standard and scaled PINN approaches, we let the MLP $\mathcal{NN}(\mathbf{r}; \boldsymbol{\theta})$ consists of 8 hidden-layers each with 60 nodes that contain biases and activation functions specified by $f(\mathbf{r}) = tanh(\mathbf{r})$, where $\mathbf{r} = [x, y, t]^T$. The output node of this MLP will contain one node with a bias and a linear activation function. We also initialize all biases to zero and use the well-known Xavier initialization for the weights. For fair comparison, we use the collocation points of Section 6.3.1, however, we do not include the boundary and initial condition collocation points as they are unnecessary for this approach. To minimize the appropriate loss function of the form (6.20), we use the L-BFGS optimizer with 20,000 iterations.

This setup, which is nearly identical to the standard and scaled PINN approaches, produces the results in Figure 6.9 and Table 6.3. These results showcase the true power of the PCNN approach. From the figure, we see that the error produced is significantly less than the scaled PINN

approach. Additionally, the error is more consistent for all of time in comparison to both approaches.

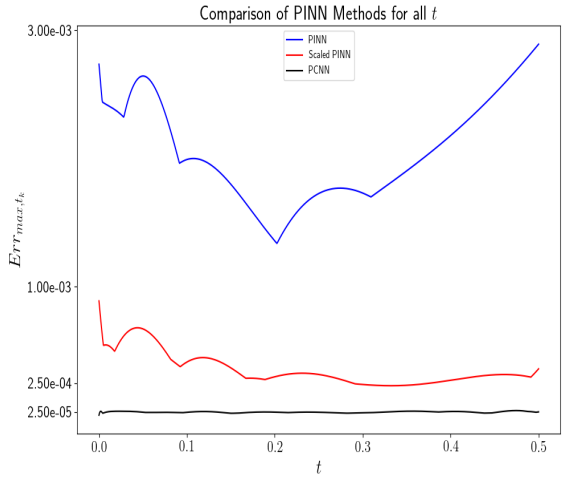



Figure 6.9: Maximum error, $Err_{max,t_k}$, with $k = 0, \ldots, 999$, for all presented PINN approaches.

Table 6.3: Various PINN method solution values for the 2D heat equation.

| Method | $MSE_F$ | $MSE_{BC}$ | $MSE_0$ | $Err_{max}$ |
|---|---|---|---|---|
| PINN | 9.1504e-7 | 8.1439e-7 | 2.2278e-7 | 2.8909e-3 |
| Scaled PINN | 6.1049e-5 | 1.4135e-8 | 9.3932e-9 | 8.9010e-4 |
| PCNN | 8.4978e-7 | – | – | 3.5851e-5 |

### 6.4.2 Allen-Cahn PDE Inverse Model: PCNN approximations

In this subsection, we validate the PCNN approach for solving an inverse problem related to a model discussed earlier. Specifically we demonstrate how, once provided data from say experiments, one can infer parameter values that coincide with a particular solution form. To show this, we use the deterministic A-C system (6.21)–(6.23), as outlined in Chapter 4.

$$\frac{\partial}{\partial t}c(x,t) = \frac{c(x,t)-c^3(x,t)}{(\epsilon_{tw})^2} + \Delta c(x,t), \;\; x \in D, \; 0 < t \leq T, \tag{6.21}$$

$$c(x,0) = \frac{1}{2}\left(1 - \tanh\left(\frac{x}{2\sqrt{2}\epsilon_{tw}}\right)\right), \; x \in D, \tag{6.22}$$

$$\frac{\partial c}{\partial x} = 0 \text{ on } \partial D. \tag{6.23}$$

Additionally, we will consider the exact solution of this system given by (6.24), which was introduced in subsection 4.2.2.

$$c_{exact}(x,t) = \frac{1}{2}\left(1 - \tanh\left(\frac{x-st}{2\sqrt{2}\epsilon_{tw}}\right)\right) \tag{6.24}$$

Here the right traveling wave distance translation value $s$ depends on the gradient energy parameter $\epsilon_{tw}$ and for the above example it is given by $s = 3/(\sqrt{2}\epsilon_{tw})$. The gradient energy is a crucial parameter for experimental purposes. The A-C inverse problem we consider, to demonstrate the PCNN approximations, is motivated by the idea that in general only data arising from such modeled systems are known and the interest is to determine the gradient energy and the related wave translation value. To this end, in this section we consider the inverse problem of finding the parameters $s$ and $\epsilon_{tw}$ from noisy solution data such that (6.24) satisfies the system (6.21)–(6.23) and best fits the provided data generated from the A-C modeled physical process. To generate the synthetic data, which represents experimental data and doubles as our collocation points, we add the noise $\omega$ via the normal distribution with mean 0 and standard deviation 0.1 to the exact solution (6.24). To denote this data, we use $c_{exact}(x,t;\omega)$, which is notation for the exact solution with some added noise $\omega$. The sampling of data occurs over 1000 evenly spaced time points over $[0,T]$ and 100 spatial points generated using LHS for each time point. For the

example that we consider in this section, we let $D = [-0.5, 5.5]$, $\epsilon_{tw} = 0.0197$, $s = 3/(\sqrt{2}\epsilon_{tw})$, and $T = 2/s$. Using these parameter values, we obtain Figure 6.10, which depicts the produced data at three different time points.

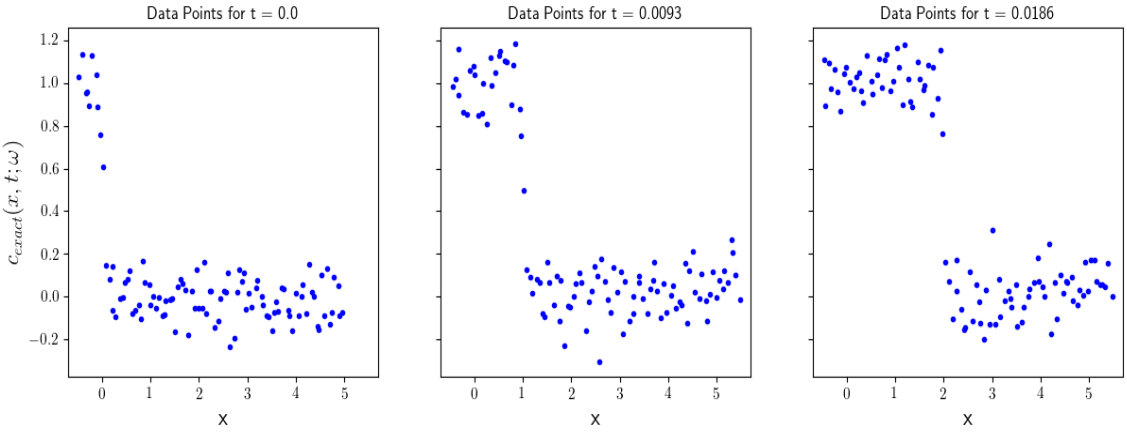



Figure 6.10: Data points for the 1D AC travelling wave PCNN parameter approximation.

To learn the variable $s$ and $\epsilon_{tw}$ of the exact solution and PDE model, we construct the PCNN approximation using (6.15), where $\mathcal{U}_{IB}(x, t)$ will be replaced by the exact solution (6.24), $\mathcal{D}(\mathbf{x}, t) = t(x + 0.5)(5.5 - x)$, and $\mathcal{U}_G(x, t)$ will be the parameterized MLP $\mathcal{NN}(\mathbf{r}; \boldsymbol{\theta}) : \mathbb{R}^2 \to \mathbb{R}$ that consists of 4 hidden-layers each with 40 nodes that contain biases and activation functions specified by $f(\mathbf{r}) = tanh(\delta\mathbf{r})$, where $\mathbf{r} = [x, t]^T$ and $\delta$ is a learnable parameter of the activation function. The output node of this neural network will contain one node with a bias and a linear activation function. We also initialize all biases to zero and use the well-known Xavier initialization for the weights.

In this particular case, the MLP $\mathcal{U}_G(x, t)$ acts as a connection between the inferred solution (6.24) and the domain under consideration. We then utilize the loss function (6.25), which incorporates the PDE model (6.25a), initial condition (6.25b), and data points for $t > 0$ (6.25c). Additionally, this loss function contains the scaling parameter $\lambda$ in (6.25b) and (6.25c), which allows one to put more emphasis on the data points. For our simulations, we let $\lambda = 10$. As stated before, we use the experimental data as collocation points. In particular, $x_{F,i}, t_{F,i}$ are all data points with $N_F = 100,000$, $\mathbf{r}_{F,i} = [x_{F,i}, t_{F,i}]^T$, $x_{0,i}$ are initial condition points with $N_0 = 100$, and

$\mathbf{r}_{0,i} = [x_{0,i}, 0]^T$.

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N_F}\sum_{i=1}^{N_F}\Big(\mathcal{J}(\mathcal{U}(\mathbf{r}_{F,i};\boldsymbol{\theta}))\Big)^2 \tag{6.25a}$$

$$+\,\lambda\frac{1}{N_0}\sum_{i=1}^{N_0}\Big(\mathcal{U}(\mathbf{r}_{0,i};\boldsymbol{\theta}) - c_{exact}(\mathbf{r}_{0,i};\omega)\Big)^2 \tag{6.25b}$$

$$+\,\lambda\frac{1}{N_F}\sum_{i=1}^{N_F}\Big(\mathcal{U}(\mathbf{r}_{F,i};\boldsymbol{\theta}) - c_{exact}(\mathbf{r}_{F,i};\omega)\Big)^2 \tag{6.25c}$$

Using the framework we have specified for the MLP, the weights $\boldsymbol{\theta}$ and parameters $s$, $\epsilon_{tw}$, and $\delta$ are found by minimizing the loss function (6.25), using the L-BFGS optimizer with 3,824 iterations. This produces the results in Table 6.4 where *s error* and $\epsilon_{tw}$ *error* are the absolute relative error of $s$ and $\epsilon_{tw}$, respectively. These results show that the parameters $s$ and $\epsilon_{tw}$ have been sufficiently approximated via the provided data and PCNN approximation. Furthermore, Figure 6.11 provides the PCNN approximation at various times. When coupled, these results provide evidence that not only can this approach solve the inverse problem, but it can also serve as a sufficient approximation of the PDE model.

Table 6.4: Results for 1D AC inverse solution using the PCNN approximation.

| *s error* | $\epsilon_{tw}$ *error* | $\mathcal{L}(\boldsymbol{\theta})$ |
|---|---|---|
| 1.10063e-3 | 1.10246e-3 | 0.18606 |

## 6.5 PCNNs for a Moving Domain Schrödinger Model

As discussed in Chapter 5, since the introduction of the seminal 1952 paper by Moshinsky [49] entitled *Diffraction in Time* (DiT), there has been a substantial interest in understanding more practical variants of the DiT model governed by the Schrödinger equation. One specific application is that of particles impinging on a shutter in conjunction with time-dependent spatial-boundedness. Although some analytical approaches have been developed to solve these types of problems, they often require specific assumptions that are unrealistic such as no

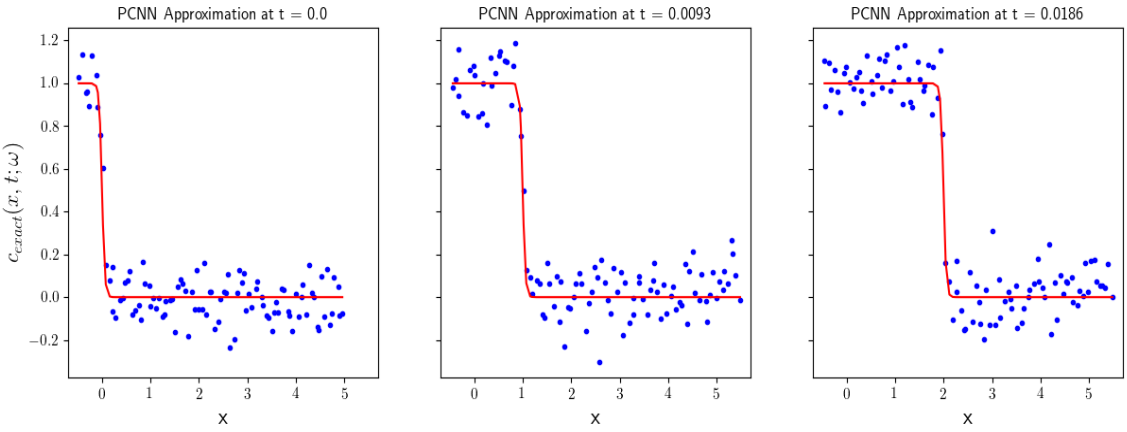



Figure 6.11: Data points (blue dots) and PCNN approximation (red line) for the 1D AC travelling wave at various time points.

boundary and the process occurs in the entire real-line. For this reason, one needs to look beyond analytical representations and instead consider numerical approximations of such scenarios, see for example [21].

Although it is apparent that numerical simulations are necessary, developing these numerical approaches is not simple using standard numerical techniques, such as finite difference or finite element methods. This is due to the fact that the incorporation of the moving domain requires careful modification of the numerical scheme. In the case of the finite element method, this feature requires modification of the basis functions and associated mass, stiffness, and hybrid matrices as seen in Chapter 5 section 5.2. The complexity of this formulation then increases as the dimension of the spatial domain grows. This gives rise to a significant amount of memory and computational cost, especially for moving domain problems. In this section, we explore the applicability of PCNN approximations as an alternative to these well-established numerical methods. For all tests, we aim to achieve relative errors of 0.1% as it is well-known that the approximation of a PINN's achievable relative error tends to stagnate around 0.1–1.0%, irrespective of the number of neurons or layers used to define the underlying network architecture [131]. However, we note that in many applications 0.1% error is often sufficient.

In this section, we consider the deterministic moving domain Schrödinger model (introduced in Chapter 5), induced by a monochromatic beam of particles moving parallel to the $x$-axis impinging on a shutter that moves perpendicular to the beam in its frame:

$$\mathrm{i}\frac{\partial}{\partial t}\psi(\mathbf{x},t) + \bar{\alpha}(k)\Delta\psi(\mathbf{x},t) = F(\mathbf{x},t), \qquad \mathbf{x} \in D(t), \qquad 0 < t \leq T, \tag{6.26a}$$

$$\psi(\mathbf{x},0) = G_k(\mathbf{x}), \quad \mathbf{x} \in \overline{D(0)}, \tag{6.26b}$$

$$\psi(\mathbf{x},t) = 0, \qquad \mathbf{x} \in \partial D(t) \qquad 0 \leq t \leq T. \tag{6.26c}$$

Here, $\mathbf{x}$ denotes the spatial domain (which we consider in 1, 2, and 3D), $T$ is the final time, $G(\mathbf{x})$ is the initial condition, and $F(\mathbf{x},t)$ is the problem specific forcing function. The moving domain of the problem is determined by a moving frame with the deterministic speed $\gamma_{x_i}{}'(t) = \omega_{x_i} f(t)$, where $x_1$, $x_2$, and $x_3$ denote the $x, y$ and $z$ directions, respectively. In the above model $k$ denotes the wavenumber representing the oscillatory nature of the initial state.

For simulation purposes, we choose the moving domain of the problem for a chosen dimension $d$, with $1 \leq d \leq 3$ as $D(t) = \prod_{i=1}^{d}(\beta_{x_i}, \gamma_{x_i}(t))$, where $\beta_{x_1}, \beta_{x_2}$, and $\beta_{x_3}$ denote the initial starting position of the beam of particles in the $x, y$, and $z$ directions, respectively. As discussed earlier, without loss of generality, we chose homogeneous boundary conditions in this chapter and we take the non-zero initial state.

To create a NN approximation of (6.26), we will consider the PCNN approach. We consider the PCNN approach because of its ability to automatically satisfy complex initial and boundary condition data, and its consistent error throughout time (as demonstrated in Section 6.4.1). For this PCNN approach we have to modify the approximation (6.15). This is due to the fact that most widely distributed tensor tools, such as PyTorch and Tensorflow, do not have well-established frameworks for dealing with complex-valued tensors, even though their use has shown advantageous [132]. For this reason, one needs to create approximations of both the real and complex -valued portions of $\psi(\mathbf{x},t)$. Given we are carrying out the approximation in this way, we need to represent (6.26) as a system of real-valued coupled PDEs.

To do this, we first decompose $\psi(\mathbf{x}, t)$ and $F(\mathbf{x}, t)$ into their real and imaginary parts, as provided by (6.27) and (6.28), respectively, where the subscripts $R$ and $I$ denote the real and complex parts, respectively.

$$\psi(\mathbf{x}, t) = \psi_R(\mathbf{x}, t) + \mathrm{i}\psi_I(\mathbf{x}, t) \tag{6.27}$$

$$F(\mathbf{x}, t) = F_R(\mathbf{x}, t) + \mathrm{i}F_I(\mathbf{x}, t) \tag{6.28}$$

Plugging (6.27) and (6.28) into (6.26a) we obtain (6.29).

$$\mathrm{i}\left(\frac{\partial}{\partial t}\psi_R(\mathbf{x}, t) + \mathrm{i}\frac{\partial}{\partial t}\psi_I(\mathbf{x}, t)\right) + \bar{\alpha}(k)\left(\Delta\psi_R(\mathbf{x}, t) + \mathrm{i}\Delta\psi_I(\mathbf{x}, t)\right) = F_R(\mathbf{x}, t) + \mathrm{i}F_I(\mathbf{x}, t) \tag{6.29}$$

Separating (6.29) yields

$$\begin{aligned} \mathrm{i}\left(\frac{\partial}{\partial t}\psi_R(\mathbf{x}, t) + \bar{\alpha}(k)\Delta\psi_I(\mathbf{x}, t)\right) &= \mathrm{i}F_I(\mathbf{x}, t), & (6.30)\\ -\frac{\partial}{\partial t}\psi_I(\mathbf{x}, t) + \bar{\alpha}(k)\Delta\psi_R(\mathbf{x}, t) &= F_R(\mathbf{x}, t). & (6.31) \end{aligned}$$

Thus, (6.26) becomes the system of real-valued coupled PDEs (6.32).

$$\begin{aligned} &\frac{\partial}{\partial t}\psi_R(\mathbf{x}, t) + \bar{\alpha}(k)\Delta\psi_I(\mathbf{x}, t) = F_I(\mathbf{x}, t), m \qquad \mathbf{x} \in D(t), \qquad 0 < t \leq T, && (6.32a)\\ &\frac{\partial}{\partial t}\psi_I(\mathbf{x}, t) - \bar{\alpha}(k)\Delta\psi_R(\mathbf{x}, t) = -F_R(\mathbf{x}, t), \qquad \mathbf{x} \in D(t), \qquad 0 < t \leq T, && (6.32b)\\ &\psi_R(\mathbf{x}, 0) = G_{R,k}(\mathbf{x}), \quad \mathbf{x} \in \overline{D(0)}, && (6.32c)\\ &\psi_I(\mathbf{x}, 0) = G_{I,k}(\mathbf{x}), \quad \mathbf{x} \in \overline{D(0)}, && (6.32d)\\ &\psi_R(\mathbf{x}, t) = 0 = \psi_I(\mathbf{x}, t), \qquad \mathbf{x} \in \partial D(t) \qquad 0 \leq t \leq T. && (6.32e) \end{aligned}$$

Now that the Schrödinger model is represented in terms of real-valued PDEs, we can commence with our PCNN approximation. Similar to (6.15), we can create PCNN approximations of both the real and complex parts as in (6.33) and (6.34), respectively.

$$\psi_R(\mathbf{x}, t) \approx \hat{\psi}_R(\mathbf{x}, t; \boldsymbol{\theta}) = \psi_{R,IB}(\mathbf{x}, t) + \mathcal{D}(\mathbf{x}, t) \odot \psi_{R,G}(\mathbf{x}, t; \boldsymbol{\theta}) \tag{6.33}$$

$$\psi_I(\mathbf{x}, t) \approx \hat{\psi}_I(\mathbf{x}, t; \boldsymbol{\theta}) = \psi_{I,IB}(\mathbf{x}, t) + \mathcal{D}(\mathbf{x}, t) \odot \psi_{I,G}(\mathbf{x}, t; \boldsymbol{\theta}) \tag{6.34}$$

For the PCNN approximations (6.33) and (6.34), we can construct an exact functional representation of $\mathcal{D}(\mathbf{x}, t)$ in a dimension $d$ as provided by (6.35). By inspection, one can see that the function in (6.35) is zero at the initial and boundary conditions while being nonzero otherwise.

$$\mathcal{D}(\mathbf{x}, t) = t \prod_{i=1}^{d} (x_i - \beta_{x_i})(\gamma_{x_i}(t) - x_i) \tag{6.35}$$

The functions $\psi_{R,IB}(\mathbf{x}, t)$ and $\psi_{I,IB}(\mathbf{x}, t)$ will also be represented exactly in the following subsections. Lastly, the functions $\psi_{R,G}(\mathbf{x}, t)$ and $\psi_{I,G}(\mathbf{x}, t)$ will be approximated using MLPs. Specifically, we will use the parameterized MLP $\mathcal{NN}(\mathbf{r}; \boldsymbol{\theta}) : \mathbb{R}^{d+1} \to \mathbb{R}$, where $d$ is the spatial dimension of the problem. The depth and width of the network will be adjusted in conjunction with the number of spatial dimensions, where each hidden layer node will contain biases and activation functions, along with an output node that will contain one node with a bias and a linear activation function. For these MLPs, we will initialize all biases to zero and use the well-known Xavier initialization for the weights.

To train the PCNN approximation we will construct specialized collocation points. These collocation points will be distributed such that more attention is focused on the beginning portion of the time interval. This is motivated by the fact that most PINN approximations accumulate larger error at the start of the interval and this error propagates throughout the rest of the interval, as seen in Figure 6.9. By concentrating more points toward the beginning of the interval, we can reduce this accumulation of error. More specifically, we separate the time interval $[0, T]$ into three sections. To produce this separation, we take the total number of time points $N_t$ and place $3floor(N_t/6)$ points in the interval $[0.0, T/3)$, $2floor(N_t/6)$ points in the interval $[T/3, 2(T/3)]$, and $floor(N_t/6) + r$ points in the interval $(2(T/3), T]$, where $r = N_t \bmod 6$. Then, for each time point we generate spatial points using LHS with input dimension $d$ and $N_s$ sampling points [125]. A visual representation of these points can be seen in Figure 6.12.

Once the collocation points have been produced, we then need to minimize a loss function. For this example, we can construct the loss function as in (6.36), with real and complex losses (6.37) and (6.38). For the constructed loss functions $N_F = N_t * N_s$. It should be noted that the use of this

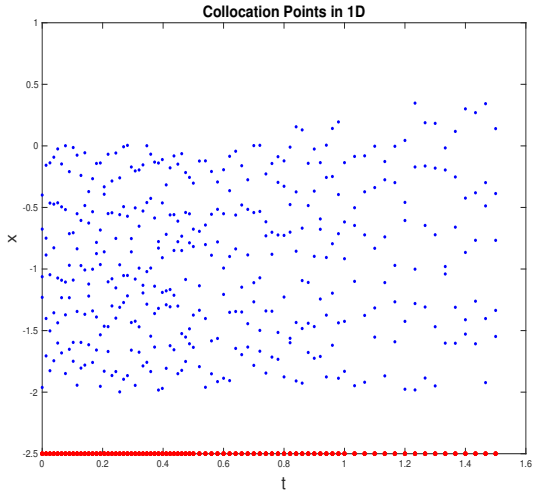



Figure 6.12: Collocation points (in blue) using $d = 1$ with $T = 1.5$, $N_t = 80$, and $N_s = 80$, where the red-dots on the $t$-axis show a visual representation of the time points.

loss function further couples the system of real-valued PDEs.

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_R(\boldsymbol{\theta}) + \mathcal{L}_I(\boldsymbol{\theta}) \tag{6.36}$$

$$\mathcal{L}_R(\boldsymbol{\theta}) = \frac{1}{N_F}\sum_{i=1}^{N_F}\left(\left[\frac{\partial}{\partial t_{F,i}}\hat{\psi}_I(\mathbf{r}_{F,i};\boldsymbol{\theta}) - \bar{\alpha}(k)\Delta\hat{\psi}_R(\mathbf{r}_{F,i};\boldsymbol{\theta})\right] + F_R(\mathbf{r}_{F,i})\right)^2 \tag{6.37}$$

$$\mathcal{L}_I(\boldsymbol{\theta}) = \frac{1}{N_F}\sum_{i=1}^{N_F}\left(\left[\frac{\partial}{\partial t_{F,i}}\hat{\psi}_R(\mathbf{r}_{F,i};\boldsymbol{\theta}) + \bar{\alpha}(k)\Delta\hat{\psi}_I(\mathbf{r}_{F,i};\boldsymbol{\theta})\right] - F_I(\mathbf{r}_{F,i})\right)^2 \tag{6.38}$$

To minimize (6.36), for all simulations we will be using the L-BFGS optimizer [124] with the number of iterations determined by the spatial dimension of the problem.

In order to gauge the effectiveness of the PCNN approach for the Schrödinger model, we compute the maximum relative error. More specifically, we first compute the relative error as stated in (6.39), where $\|\cdot\|_F$ is the vector 2-norm and for matrices is the Frobenius norm, $t_\ell$ denote the time grid points $t_\ell = t_0 + \ell * \Delta t$ for $\ell = 0, \ldots, K-1$ and $\Delta t = \frac{T}{K-1}$, $u_{pred}(\mathbf{x}, t_\ell; \boldsymbol{\theta})$ is the predicted PCNN solution, and $u(\mathbf{x}, t_\ell)$ is the exact or comparison solution.

$$e_u(\ell) = \frac{\|u_{pred}(\mathbf{x}, t_\ell; \boldsymbol{\theta}) - u(\mathbf{x}, t_\ell)\|_F}{\|u(\mathbf{x}, t_\ell)\|_F} \tag{6.39}$$

We represent the 1D spatial grid points as $x_m$, 2D grid points as $x_m, y_n$, and the 3D grid points as $x_m, y_n, z_p$, where $x_m = x_0 + m * h_x$, $y_n = y_0 + n * h_y$, and $z_p = z_0 + p * h_z$ for $m, n, p = 0, \ldots, N-1$ and $h_x = \frac{\gamma_x(t_\ell)-\beta_x}{N-1}, h_y = \frac{\gamma_y(t_\ell)-\beta_y}{N-1}, h_z = \frac{\gamma_z(t_\ell)-\beta_z}{N-1}$. The maximum relative error is then given by (6.40).

$$\max \|e_u\|_F = max\{e_u(\ell) : \ell = 0, \ldots, K-1\} \tag{6.40}$$

For all 1D and 2D simulations, we let $N = K = 1281$, and in 3D we let $N = K = 201$.

To construct the established PCNN approximation of the Schrödinger model, we utilize Tensorflow version 1. Tensorflow version 1 was chosen for its ability to easily allow for the use of the LFBGS optimizer with little overhead cost. Additionally, Tensorflow was chosen for its simplicity in deploying the tensor computation to GPUs. For all experiments in this section, we use one GPU on IBM Power8 S822LC nodes that contain NVIDIA Tesla K80 GPUs.

For demonstrating our PCNN approach for the above moving domain model in $1 \le d \le 3$ dimensions, we choose the initial state to be a modulated plane wave impinging from the unit direction vector $\hat{\mathbf{d}}$, with each of its $d$ components taking the value $v_d = 1/\sqrt{d}$, and the wavenumber $\tilde{k} = \sqrt{d}k$ for several values of the scaled reference wavenumber $k \ge 1$. The modulation function $m(\mathbf{x}, t) = \sin\left(\pi \prod_{i=1}^{d}(x_i - \gamma_{x_i}(t))(x_i - \beta_{x_i})\right)$ is chosen to satisfy the homogeneous boundary condition. For validating the PCNN based solution using this initial state, we choose the exact solution to be the associated time-harmonic wave with frequency $w$. More specifically, in this chapter we take $F$ in (6.26a) so that

$$\psi(\mathbf{x}, t) = e^{\mathrm{i}(\,\tilde{k}\mathbf{x}\cdot\hat{\mathbf{d}} - wt\,)} m(\mathbf{x}, t), \qquad \mathbf{x} \in \overline{D(t)}, \quad t \in [0, T], \tag{6.41}$$

is the unique solution of the of moving domain model (6.26a)–(6.26c). For simulation we consider the above model with time-harmonic frequency $w = k^2$, $\bar{\alpha} = 1$, $T = 1.5$, and choose the moving frame speed in all directions to be $\gamma_{x_i}(t) = t^2/4$, and $\beta_{x_i} = -2$ for $i = 1, 2, 3$ so that, for $d = 1, 2, 3$ the diameter of the initial domain grows from $2\sqrt{d}$ to approximately $2.6\sqrt{d}$. That is, with $\tilde{k} = \sqrt{d}k$, the wavelength (and hence complexity) of the problem grows from $(dk)/(\pi)$ to approximately $(2.6dk)/(2\pi)$. That is, in terms of complexity of the model, the case $k = 1$ for $d = 3$ can be considered similar to that for $k = 5$ in the one-dimensional case.

As we shall see below, unlike in the FEM case investigated in the previous chapter for the same moving domain problem (with a single CPU core run time of less than 10 seconds, see Table 5.10), using NN-approximations for $k > 5$ in the one-dimensional moving domain case is computationally impractical even when utilizing GPU computations, as demonstrated using a GPU with about 5000 cuda cores (Tesla K80). Further, it is well known that relative errors for purely NN-based approximations for the PDE governed solutions stagnate around 0.1% [131] while, as observed in earlier chapters, FEM approximations facilitate high-order accuracy. However, we choose NN-based formulation in this chapter as constructing moving-mesh finite element based approximations in higher dimensions lead to algebraic system setup and high memory usage complexities. For this numerical experiment model, we consider the PCNN approximation using representations (6.33) and (6.34), where $\mathcal{D}(x,t)$ will be chosen as in (6.35). Additionally, with initial condition induced by $\psi$ in (6.41), we choose $\psi_{R,IB}(\mathbf{x},t)$ and $\psi_{R,IB}(\mathbf{x},t)$ to be respectively the real and imaginary parts of $e^{\mathrm{i}(\tilde{k}\mathbf{x}\cdot\hat{\mathbf{d}})}m(\mathbf{x},t)$. We let $\psi_{R,G}(\mathbf{x},t;\boldsymbol{\theta})$ and $\psi_{I,G}(\mathbf{x},t;\boldsymbol{\theta})$ both be parameterized MLPs $\mathcal{NN}(\mathbf{r};\boldsymbol{\theta}) : \mathbb{R}^{d+1} \to \mathbb{R}$ initialized as described in the previous section.

### 6.5.1 PCNN Moving Domain Schrödinger Model in 1D

Using the above established algorithmic and numerical experimental framework for the PCNN approximation, we will first investigate the $d = 1$ case. For comparison, we first direct the reader's attention to Section 5.2.2, where we consider the moving-mesh FEM approximation against this exact solution. From these results, it is apparent that the moving-mesh FEM approximation is extremely good at providing an approximation of the presented problem. In particular, the FEM approach produces consistent convergence for a variety of $k$ values. Although, as demonstrated in Table 5.2 - Table 5.9, the larger $k$ values, e.g. $k > 6$, do begin to produce relative errors that increase as $k$ does. This is due to the fact that large $k$ values produce solutions that are highly oscillatory as demonstrated in Figure 5.3.

To determine the most appropriate activation functions for these MLPs in 1D and in higher dimensions, we consider several different activation functions. For the initial test, we utilized $tanh(x)$ activation functions. The results of these activation functions are shown in Table 6.5 for $k \leq 5$, and they demonstrate sufficient relative error for all $k$ values considered. In our second test, we use the slightly modified $swish(\delta x)$ activation function, which is a hybrid function that is the combination of a sigmoid and the ReLU activation functions [133], where the $\delta$ parameter is a trainable variable. We consider the *swish* activation function as it has shown promise in improving the performance of deep neural networks [133]. The results of the *swish* activation function are shown in Table 6.6 and unfortunately, for the moving domain model, produce worse relative error in comparison to *tanh* activation functions. Additionally, the *swish* activation functions induce extra computation that significantly increases the runtime.

In the same spirit of the *swish* activation function, we consider the slightly modified activation function $tanh(\delta x)$, where again the $\delta$ parameter is a trainable variable. This modified activation function produces the results in Table 6.7 and shows to be advantageous for smaller $k$ values and consistent with the results of larger $k$ values, when compared to the unmodified *tanh* activation function. Thus, in future simulations we will consider this modified *tanh* activation function. Plots comparing the learned $tanh(\delta x)$ activation functions against $tanh(x)$ are presented in Figure 6.13 for $k = 1$ and $k = 5$. Plots demonstrating the PCNN approximation against the exact solution are also provided in Figure 6.14 and Figure 6.15 for $k = 1$ and $k = 5$, respectively, for various times.

Table 6.5: PCNN Schrödinger model: Relative error with $tanh(x)$ activation function.

| $k$ | $N_s$ | $N_t$ | Nodes | max $\|\|e_u\|\|_2$ | max $\|\|e_v\|\|_2$ | Iterations | Runtime | $\mathcal{L}(\boldsymbol{\theta})$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 500 | 20 | 1.9513e-3 | 1.2609e-3 | 15,000 | 10 min | 8.7623e-6 |
| 2 | 10 | 1000 | 40 | 1.4802e-3 | 1.8363e-3 | 30,000 | 33.4 min | 3.6152e-5 |
| 3 | 15 | 1500 | 60 | 1.6905e-3 | 1.6936e-3 | 60,000 | 2.4 hrs | 1.2294e-4 |
| 4 | 20 | 2000 | 80 | 1.2547e-3 | 1.4544e-3 | 120,000 | 10.6 hrs | 9.7344e-5 |
| 5 | 25 | 2500 | 100 | 1.8785e-3 | 1.8304e-3 | 240,000 | 42 hrs | 2.0679e-4 |

Table 6.6: PCNN Schrödinger model: Relative error with $swish(\delta x)$ activation function.

| $k$ | $N_s$ | $N_t$ | Nodes | max $\|\|e_u\|\|_2$ | max $\|\|e_v\|\|_2$ | Iterations | Runtime | $\mathcal{L}(\boldsymbol{\theta})$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 500 | 20 | 8.1779e-4 | 4.6838e-4 | 15,000 | 13.5 min | 6.4883e-6 |
| 2 | 10 | 1000 | 40 | 1.6684e-2 | 1.9248e-2 | 30,000 | 50.9 min | 1.7635e-3 |
| 3 | 15 | 1500 | 60 | 5.4914e-3 | 6.0960e-3 | 60,000 | 4.3 hrs | 4.2118e-4 |
| 4 | 20 | 2000 | 80 | 3.4612e-2 | 4.0863e-2 | 120,000 | 19.4 hrs | 1.3883e-2 |
| 5 | 25 | 2500 | 100 | 1.1302e-2 | 1.0555e-2 | 240,000 | 75.89 hrs | 8.2847e-3 |

Table 6.7: PCNN Schrödinger model: Relative error with $tanh(\delta x)$ activation function.

| $k$ | $N_s$ | $N_t$ | Nodes | max $\|\|e_u\|\|_2$ | max $\|\|e_v\|\|_2$ | Iterations | Runtime | $\mathcal{L}(\boldsymbol{\theta})$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 500 | 20 | 6.4679e-4 | 4.2552e-4 | 15,000 | 11.4 min | 4.6215e-6 |
| 2 | 10 | 1000 | 40 | 5.2059e-4 | 1.0270e-3 | 30,000 | 36.8 min | 1.4327e-5 |
| 3 | 15 | 1500 | 60 | 1.0119e-3 | 1.1836e-3 | 60,000 | 2.8 hrs | 4.7449e-5 |
| 4 | 20 | 2000 | 80 | 2.0678e-3 | 1.9877e-3 | 120,000 | 12.7 hrs | 1.1988e-4 |
| 5 | 25 | 2500 | 100 | 1.8535e-3 | 1.8989e-3 | 240,000 | 48.9 hrs | 1.3366e-4 |

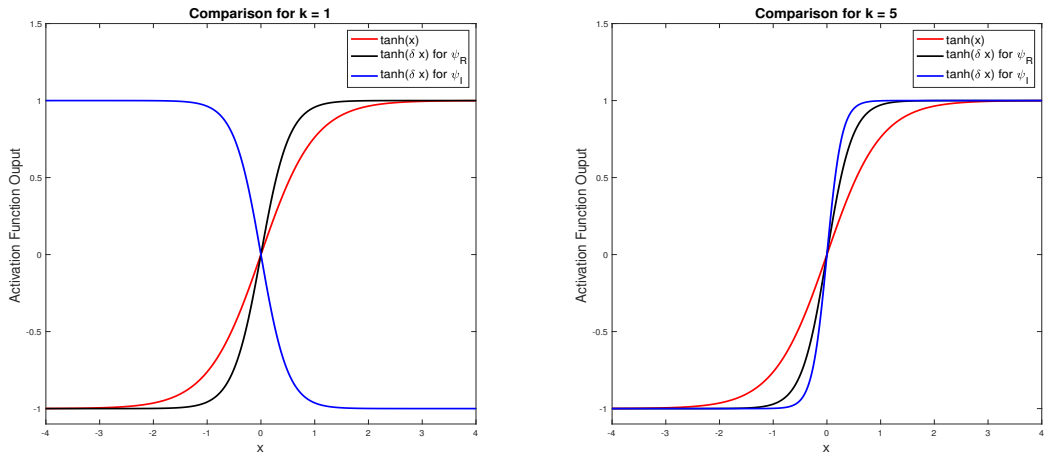



Figure 6.13: Comparison of learned $tanh(\delta x)$ activation functions against $tanh(x)$ for the 1D Schrödinger model PCNN approximation for $k = 1$ (left) and $k = 5$ (right).

To provide further evidence that PCNNs provide more accurate approximations, especially for moving domain models, we will now consider a PINN solution of the moving domain Schrödinger model in one dimension. For this PINN approximation, we will utilize the same interior collocation points as outlined in the previous section. However, as this is a PINN approximation, we now need to construct collocation points at the boundary and initial

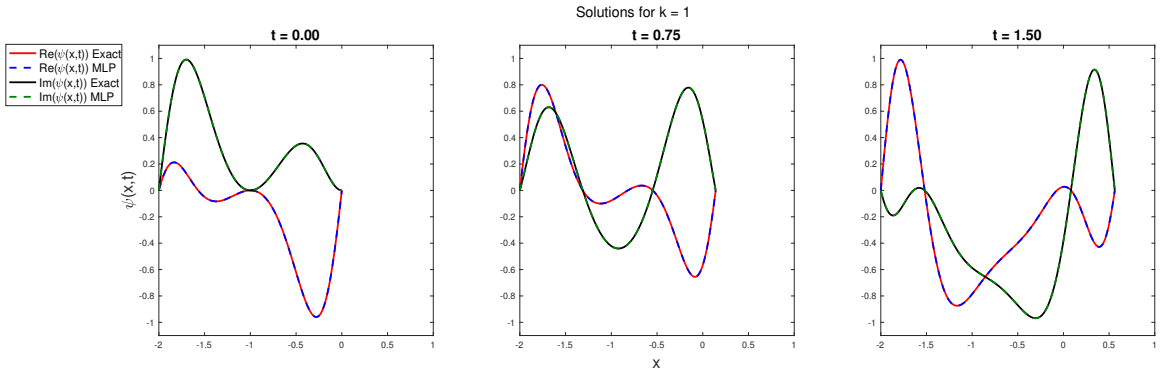



Figure 6.14: Plot of 1D PCNN solution against the exact solution at various times for $k = 1$.

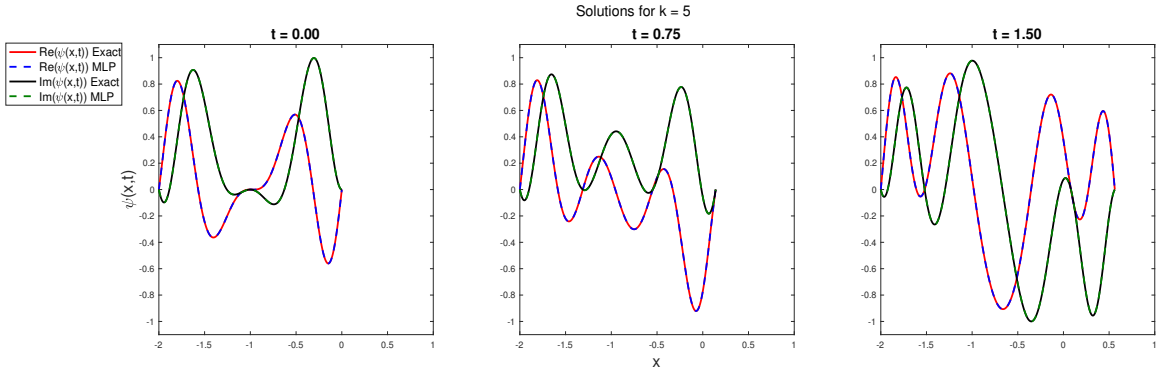



Figure 6.15: Plot of 1D PCNN solution against the exact solution at various times for $k = 5$.

conditions. The initial condition collocation points $x_{0,i}$ for $i = 1, \ldots, N_0$ will be constructed for $D(0)$ using LHS with input dimension of 1 and $N_0 = 1,000$, where the time points are set to zero. The boundary condition collocation points $t_{BC,i}$ for $i = 1, \ldots, 2 * N_{BC}$ will be constructed by first creating time points between zero and $T$ using LHS with input dimension 1 and $N_{BC}$ sampling points. Then, for each time point, the collocation points $x_{BC,i}$ for $i = 1, \ldots, 2 * N_{BC}$ will be found by adding the left and right boundary point at the specified time point.

Using the above collocation points, we then minimize the loss function (6.42), where $\mathcal{L}_R(\boldsymbol{\theta})$ and $\mathcal{L}_I(\boldsymbol{\theta})$ are as in (6.37) and (6.38), respectively, and the additional terms are provided in (6.43) and (6.44). It should be noted that now, both $\hat{\psi}_R(\mathbf{x}, t; \boldsymbol{\theta})$ and $\hat{\psi}_I(\mathbf{x}, t; \boldsymbol{\theta})$ are the parameterized MLP $\mathcal{NN}(\mathbf{r}; \boldsymbol{\theta}) : \mathbb{R}^2 \to \mathbb{R}$, rather than the PCNN approximations (6.33) and (6.34). For all simulations

of the PINN, we let there be four hidden layers as in the PCNN approximation.

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_R(\boldsymbol{\theta}) + \mathcal{L}_I(\boldsymbol{\theta}) + \mathcal{L}_{R,IB}(\boldsymbol{\theta}) + \mathcal{L}_{I,IB}(\boldsymbol{\theta}) \tag{6.42}$$

$$\mathcal{L}_{R,IB}(\boldsymbol{\theta}) = \frac{1}{2N_{BC}} \sum_{i=1}^{2N_{BC}} \left( \hat{\psi}_R(\mathbf{r}_{BC,i}; \boldsymbol{\theta}) - \psi_R(\mathbf{r}_{BC,i}) \right)^2 \tag{6.43a}$$

$$+ \frac{1}{N_0} \sum_{i=1}^{N_0} \left( \hat{\psi}_R(\mathbf{r}_{0,i}; \boldsymbol{\theta}) - \psi_R(\mathbf{r}_{0,i}) \right)^2 \tag{6.43b}$$

$$\mathcal{L}_{I,IB}(\boldsymbol{\theta}) = \frac{1}{2N_{BC}} \sum_{i=1}^{2N_{BC}} \left( \hat{\psi}_I(\mathbf{r}_{BC,i}; \boldsymbol{\theta}) - \psi_I(\mathbf{r}_{BC,i}) \right)^2 \tag{6.44a}$$

$$+ \frac{1}{N_0} \sum_{i=1}^{N_0} \left( \hat{\psi}_I(\mathbf{r}_{0,i}; \boldsymbol{\theta}) - \psi_I(\mathbf{r}_{0,i}) \right)^2 \tag{6.44b}$$

Using the constructed PINN, we utilize $tanh(\delta x)$ activation functions, initialize all biases to zero and use Xavier initialization for the weights, and minimize (6.42) using the L-BFGS optimizer. This Provides the results in Table 6.8 that clearly demonstrate the increase of relative error for the PINN approximation.

Table 6.8: Comparison of PCNN and the standard PINN for the Schrödinger model using $tanh(\delta x)$ activation functions for $k = 1$.

| Method | Nodes | max $\|\|e_u\|\|_2$ | max $\|\|e_v\|\|_2$ | Iterations | Runtime | $\mathcal{L}(\boldsymbol{\theta})$ |
|---|---|---|---|---|---|---|
| PCNN | 20 | 6.4679e-4 | 4.2552e-4 | 15,000 | 11.4 min | 4.6215e-6 |
| PINN | 20 | 4.0131e-3 | 2.2694e-3 | 15,000 | 10.1 min | 3.1856e-5 |

### 6.5.2 PCNN Moving Domain Schrödinger Model in 2D

Extending the 1D version of the model into 2D is relative simple with the PCNN approach. For the MLPs, we let the activation functions be described by $tanh(\delta x)$, where $\delta$ is a learnable parameter. The activation functions $tanh(\delta x)$ were chosen in comparison to $tanh(x)$ activation functions because of their ability to provide more accurate solutions for smaller $k$ values, as demonstrated in Table 6.5 and Table 6.7. However, due to the increased complexity arising from a spatial dimension increase (as explained earlier with wavelength metric), we use double the points

specified by $N_s$ and $N_t$ for $k = 1$, in comparison to the 1-D version. In addition, we then appropriately adjust these points, depending on the wavenumber $k$ as follows:
$N_s = 10 + (k-1) * 5$ and $N_t = 1,000 + (k-1) * 500$. The distribution of these points is maintained and is as described at the beginning of the section.
The results produced by this construction are show in Table 6.9, for $k = 1$ through $k = 3$. Additionally, plots of the predicted solution for $k = 1, 3$ at various times for both the real and imaginary parts are shown in Figure 6.17, Figure 6.16, Figure 6.19, and Figure 6.18, which demonstrate the increase in complexity as $k$ also increases. Results for $k > 3$ were not obtainable in a realistic runtime as the increase in dimension results in a significant increase in run time on 1 GPU (with 5000 Cuda cores), as demonstrated in Table 6.9. Although this method has tremendous upside due to its ease of construction and ability to be deployed on GPUs, the trade-off in comparison to traditional methods is that of increased run time (especially for higher dimensions).

Table 6.9: PCNN Schrödinger model: Relative error for the 2D Schrödinger model with $tanh(\delta x)$ activation function.

| $k$ | $N_s$ | $N_t$ | Nodes | max $\|e_u\|_F$ | max $\|e_v\|_F$ | Iterations | Runtime | $\mathcal{L}(\boldsymbol{\theta})$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 10 | 1000 | 40 | 1.8545e-3 | 1.3198e-3 | 75,000 | 2.9 hrs | 3.4925e-5 |
| 2 | 15 | 1500 | 60 | 1.8887e-3 | 1.9139e-3 | 150,000 | 11.6 hrs | 1.1216e-4 |
| 3 | 20 | 2000 | 80 | 2.3188e-3 | 1.9314e-3 | 300,000 | 49.8 hrs | 2.6377e-4 |

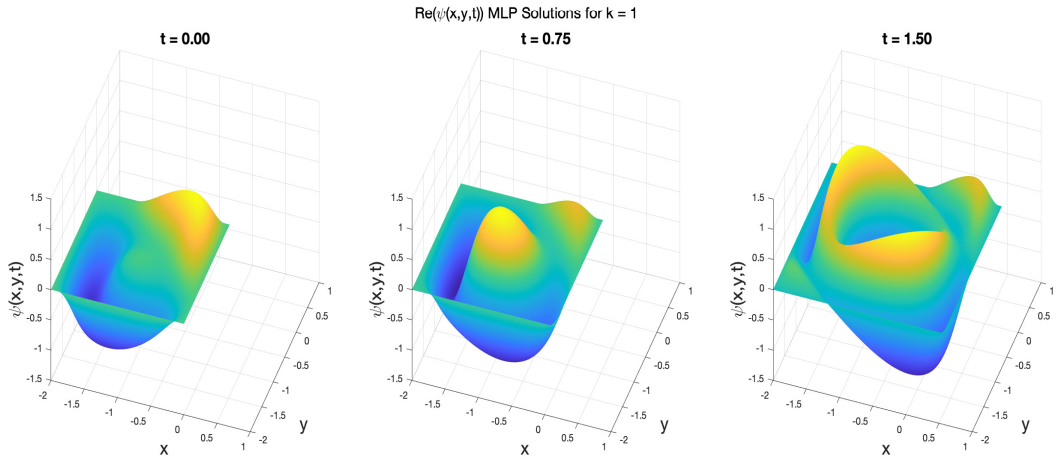



Figure 6.16: Real-part of 2D PCNN solution at various times for $k = 1$.

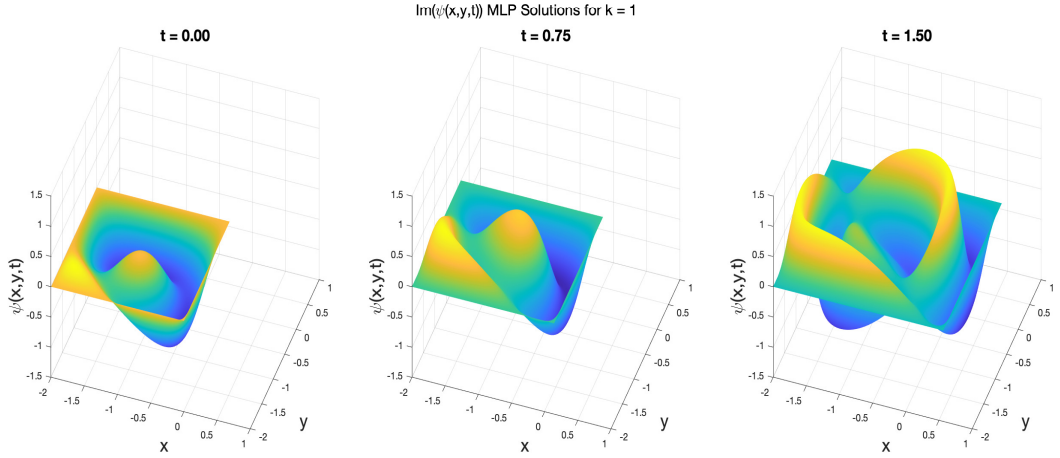



Figure 6.17: Imaginary-part of 2D PCNN solution at various times for $k = 1$.

### 6.5.3 PCNN Moving Domain Schrödinger Model in 3D

In this final section, we will consider the Schrödinger model in three dimensions. The purpose of this section is to show that the PCNN approach for the Schrödinger model can be easily extended into 3D, but this representation is accompanied by a significant increase in runtime. Thus, for simulations we consider only $k = 1$. As explained earlier using the wavelength metric, the 1D case with $k = 5$ is comparable to the 3D case with $k = 1$. To construct the MLPs, we initialize the MLPs as described at the beginning of the section using $tanh(\delta x)$ activation functions. To

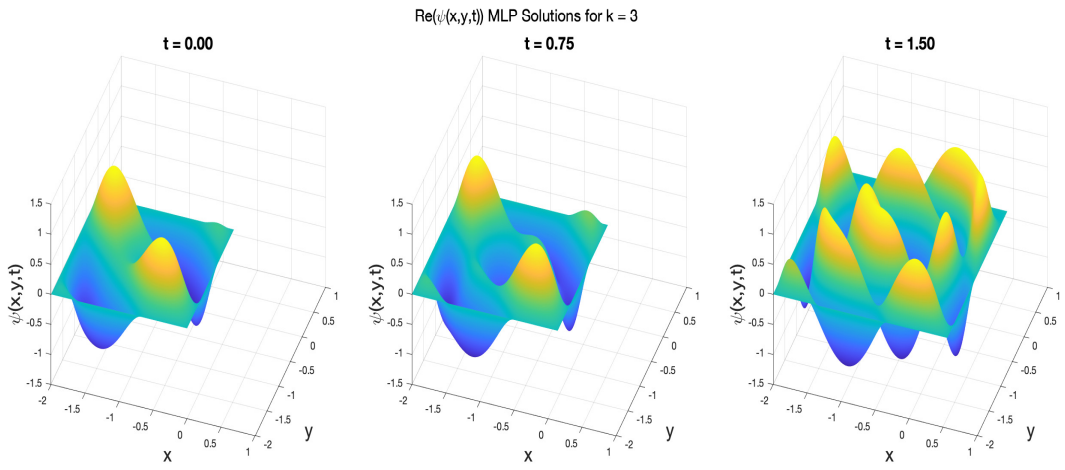



Figure 6.18: Real-part of 2D PCNN solution at various times for $k = 3$.

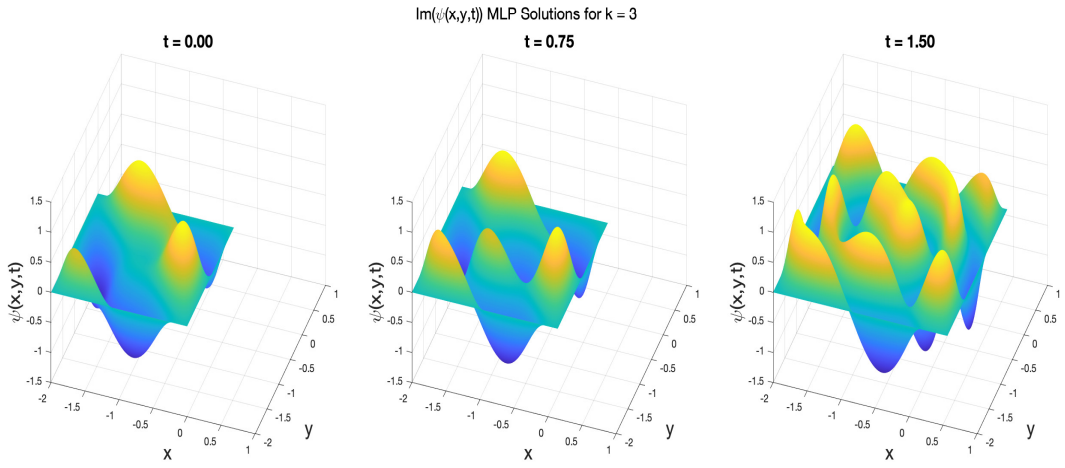



Figure 6.19: Imaginary-part of 2D PCNN solution at various times for $k = 2$.

construct the collocation points, we again increase $N_t$ and $N_s$, which is a result of even further complexity induced by the 3D solution. Lastly, as in all other cases, we use the distribution of points as in the beginning of this section.

Results of the 3D PCNN approximation for $k = 1$ are provided in Table 6.10. As stated previously, the runtime of the process incurs a significant increase in runtime. For this reason, simulations for $k > 1$ are not presented. Figure 6.20 and Figure 6.21 show the real and complex predicted solution, respectively of the PCNN approximation for $k = 1$ and highlight the

complexity of the solution. This is further solidified in Figure 6.22 and Figure 6.23, which show the real and complex exact solutions, respectively for $k = 3$. From this figure, it is evident that the complexity of the 3D simulation increases drastically for $k > 1$.

Table 6.10: PCNN Schrödinger model: Relative error for the 3D Schrödinger model with $tanh(\delta x)$ activation function.

| $k$ | $N_s$ | $N_t$ | Nodes | max $\|\|e_u\|\|_F$ | max $\|\|e_v\|\|_F$ | Iterations | Runtime | $\mathcal{L}(\boldsymbol{\theta})$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 25 | 2500 | 60 | 3.5246e-3 | 3.4308e-3 | 375,000 | 82.4 hrs | 1.7830e-4 |

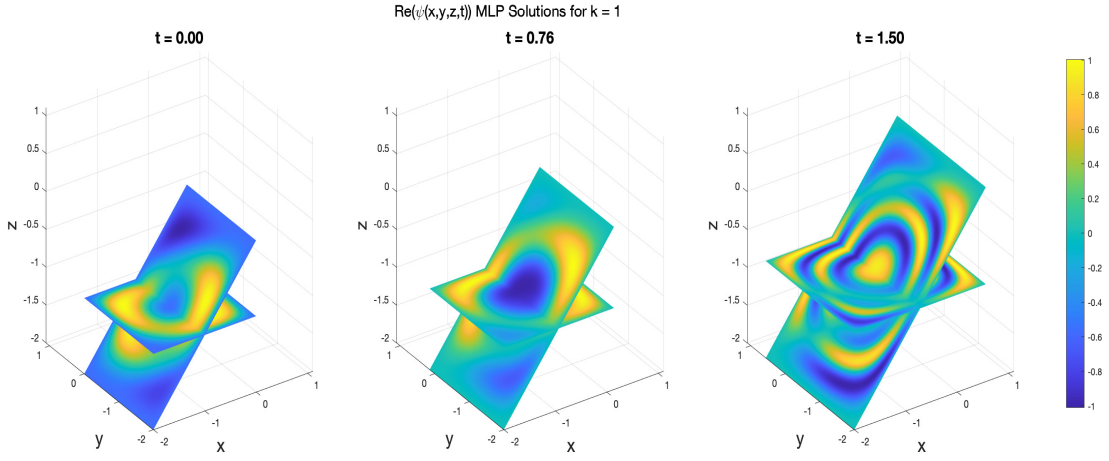



Figure 6.20: Real-part of 3D PCNN solution at various times for $k = 1$.

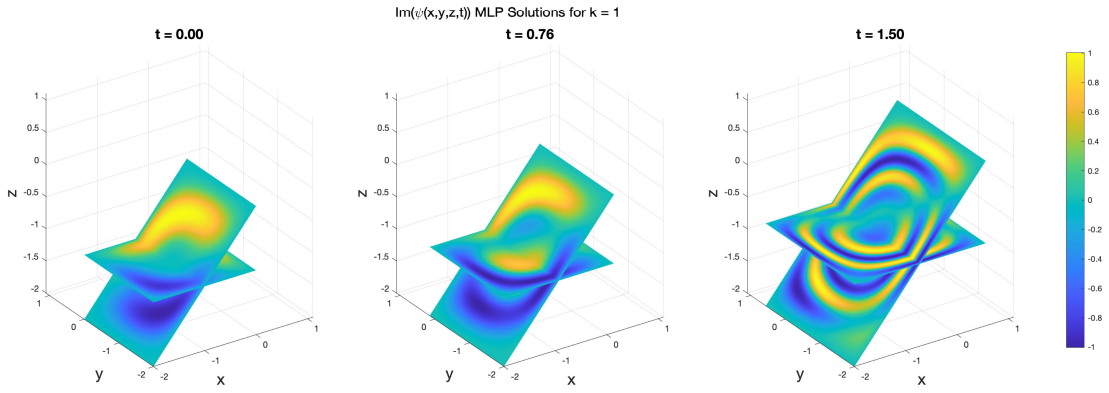



Figure 6.21: Imaginary-part of 3D PCNN solution at various times for $k = 1$.

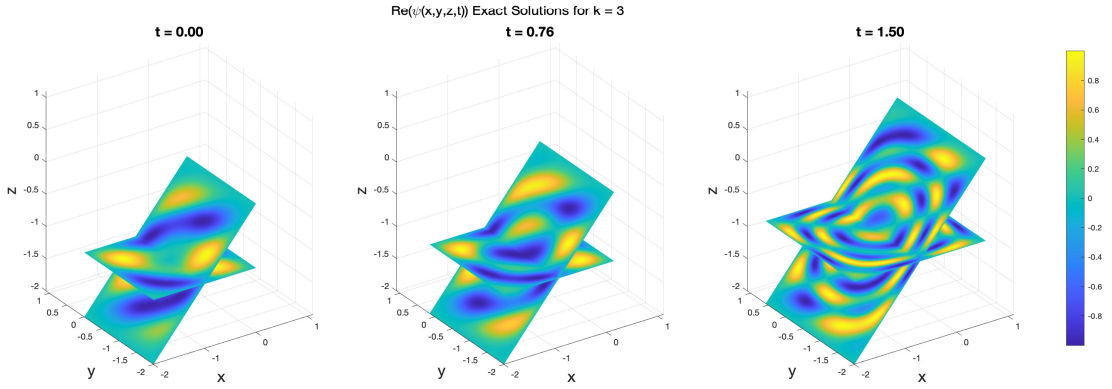



Figure 6.22: Real-part of 3D exact solution at various times for $k = 3$.

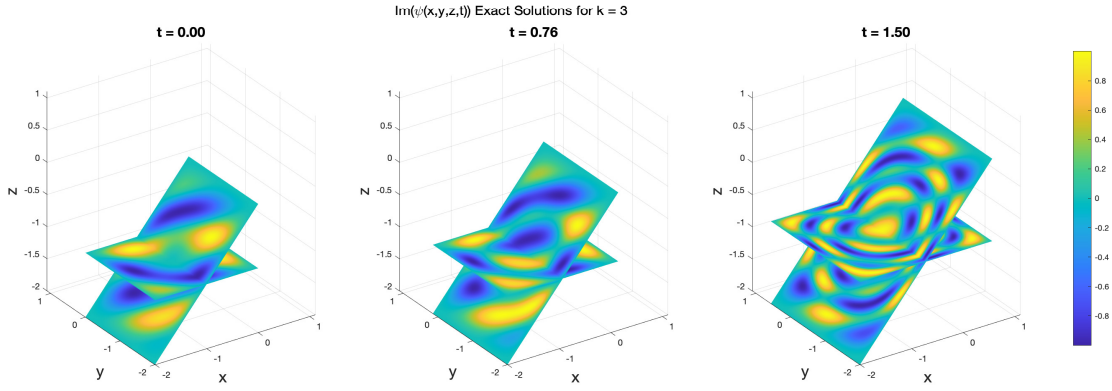



Figure 6.23: Imaginary-part of 3D exact solution at various times for $k = 3$.

# CHAPTER 7

# CONCLUSION

To understand and discover complex biological and physical processes, it is evident that efficient numerical algorithms and HPC simulations are necessary. This is the result of complex time- and space-dependent interactions being dictated by continuous systems of ODEs in time and/or PDEs in space and time, that cannot in general be solved analytically. Thankfully, their counterpart approximate discrete systems can be simulated. Although these discrete systems can be used as a tool to investigate such phenomena, their implementations must be carefully chosen such that the resulting simulations can be effectively utilized by the scientific community. In this thesis, we have constructed several such simulation techniques that address the associated challenges encountered within the simulation of these biological and physical processes. These techniques range from the reduction of computational resources for quantifying uncertainties within stochastic-space-time physical models, PCNNs capable of approximating deterministic moving domain physical models, and the identification of deterministic biological signaling pathways capable of switch-like behavior via an optimization-based method.

To begin, we considered a biochemical process modeled by a system of nonlinear ODEs that characterize signaling pathways in Chapter 2. This particular application was of significant importance as signaling pathways capable of switching between two states are ubiquitous within living organisms and distortion of these signaling pathways with switch-like behavior can manifest in underlying developmental disorders such as cancer, diabetes, and presumably a number of other pathologies. The results of this chapter include the construction of an optimization-based approach that provides an efficient procedure for the detection of switch-like bistability in reaction networks governed by mass action kinetics for any network structure, the Python tool CRNT4SBML that includes the constructed approach and several other key techniques, and a demonstration of the effectiveness of our approach in discovering switch-like

behavior using models describing a biterminal futile signaling cycle and a biterminal prion/double phosphorylation motif.

In subsequent chapters we switched our focus to numerical techniques for PDE based physical models. In order to quantify the uncertainty within the classes of stochastic PDE models considered, we demonstrated that the MC simulation cost becomes computationally prohibitive when the QoI is obtained by a time consuming numerical approximation because of its low-order convergence rate of $O(N^{-1/2})$. To improve upon MC simulation, we considered the QMC method, which has the ability to obtain a convergence rate of $O(N^{1-\epsilon})$ (for some $\epsilon > 0.5$) using high-order approximations through a carefully chosen set of quasi-random sampling points. Although MC simulations can be improved upon if QMC is used, both of these methods are shown to consume a large amount of HPC resources because the order constant in the convergence rate also depends on the variance of the QoI. Additionally, the stochastic-variable regularity of the QoI for the QMC case is tied to how much improvement is observed in the convergence rate. To reduce the computational complexity of MC and QMC, we concluded our exploration of UQ tools by considering MLMC and MLQMC simulations.

Using these established tools, in Chapter 4 we developed and implemented a hybrid high-order multi-level stochastic A-C computational model, in conjunction with finite element in space-time and efficient sampling of the high-dimensional stochastic space, to compute statistical moments of a QoI induced by the A-C output field. The input uncertainty in the model occurs both in the A-C PDE and in the initial state that facilitates the stochastic evolutionary process. Our hybrid deterministic-stochastic approximation framework is based on a space-time finite element method in two and three spatial dimensions, high-order approximations in high-dimensional stochastic variables, and fast evaluations using multi-level and parallel computations via MPI.

In Chapter 5 we considered our second stochastic model, however, this time we were concerned with a physical process on a moving domain with the underlying process governed by the space-time Schrödinger equation. Thus, the main focus of Chapter 5 was in developing and demonstrating an algorithm to compute statistical moments of a QoI, determined by a stochastic

quantum density profile induced by a random initial state. Main results of this chapter included the application of efficient methods for computing statistical moments of the QoI (using techniques established in Chapter 3) and the development of an efficient moving-mesh FEM for solving the associated deterministic Schrödinger model on a moving domain.

In Chapters 4 and 5 of this thesis, we constructed efficient simulations of the associated deterministic PDEs using the classical spline-approximations based FEM. Thanks to substantial interest in the machine learning community using NN approximations, it has been shown that NNs can be used as an alternative for solving PDE models, if an approximate error between 1% and 0.1% is sufficient for applications. Furthermore, these approximations can be constructed in such a way that non-standard domains and extension into multiple dimensions can be easily handled. To this end, we concluded this thesis by exploring NN based approximations. Key results of this chapter include the construction of a framework for moving domain models using the PCNN approach and the extension of the Schrödinger model into 2 and 3 dimensions.

REFERENCES CITED


[1] P. Massoner, M. Ladurner-Rennau, I. E. Eder, and H. Klocker. Insulin-like growth factors and insulin control a multifunctional signalling network of significant importance in cancer. *British Journal of Cancer*, 103:1479 EP –, 10 2010. URL https://doi.org/10.1038/sj.bjc.6605932.

[2] Y. Yarden and M. X. Sliwkowski. Untangling the ErbB signalling network. *Nature Reviews Molecular Cell Biology*, 2(2):127–137, 2001. doi: 10.1038/35052073. URL https://doi.org/10.1038/35052073.

[3] O.M. Fischer, S. Hart, A. Gschwind, and A. Ullrich. EGFR signal transactivation in cancer cells. *Biochemical Society Transactions*, 31(6):1203–1208, 2003. ISSN 0300-5127. doi: 10.1042/bst0311203. URL http://www.biochemsoctrans.org/content/31/6/1203.

[4] M. Feinberg. Chemical reaction network structure and the stability of complex isothermal reactors i. the deficiency zero and deficiency one theorems. *Chemical Engineering Science*, 42(10):2229 – 2268, 1987. ISSN 0009-2509. doi: https://doi.org/10.1016/0009-2509(87)80099-4. URL http://www.sciencedirect.com/science/article/pii/0009250987800994.

[5] F. Horn and R. Jackson. General mass action kinetics. *Archive for Rational Mechanics and Analysis*, 47(2):81–116, Jan 1972. ISSN 1432-0673. doi: 10.1007/BF00251225. URL https://doi.org/10.1007/BF00251225.

[6] P. R. Ellison. *The Advanced Deficiency Algorithm and Its Applications to Mechanism Discrimination*. PhD thesis, The University of Rochester, 1998.

[7] M. Feinberg. Chemical reaction network structure and the stability of complex isothermal reactors ii. multiple steady states for networks of deficiency one. *Chemical Engineering Science*, 43(1):1 – 25, 1988. ISSN 0009-2509.

[8] J. Martín and M. González. Revealing regions of multiple steady states in heterogeneous catalytic chemical reaction networks using Gröbner basis. *Journal of Symbolic Computation*, 80:521 – 537, 2017. ISSN 0747-7171. doi: https://doi.org/10.1016/j.jsc.2016.07.024. URL http://www.sciencedirect.com/science/article/pii/S0747717116300694.

[9] Y. Arkun. Detection of biological switches using the method of Gröebner bases. *BMC Bioinformatics*, 20(1):615, 2019.

[10] R. Bradford, J. H. Davenport, M. England, H. Errami, V. Gerdt, D. Grigoriev, C. Hoyt, M. Košta, O. Radulescu, T. Sturm, and A. Weber. A case study on the parametric occurrence of multiple steady states. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, page 45–52, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350648. doi: 10.1145/3087604.3087622.

[11] C. Conradi and D. Flockerzi. Switching in mass action networks based on linear inequalities. *SIAM J. Appl. Dyn. Syst.*, 11:110–134, 2012.

[12] I. Otero-Muras, P. Yordanov, and J. Stelling. Chemical reaction network theory elucidates sources of multistability in interferon signaling. *PLoS Computational Biology*, 13(4), 2017.

[13] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, 2004.

[14] O. P. Le Maitre and O. M. Kino. *Spectral Methods for Uncertainty Quantification*. Springer, 2010.

[15] M. Ganesh and S. C. Hawkins. A stochastic pseudospectral and T-matrix algorithm for acoustic scattering by a class of multiple particle configurations. *J. Quant. Spectrosc. Radiat. Transfer*, 123:41–52, 2013.

[16] J. Dick, F. Y. Kuo, and I.H. Sloan. High-dimensional integration: the quasi-Monte Carlo way. *Acta Numer.*, 22:133–288, 2013.

[17] M. B. Giles. Multilevel monte carlo methods. *Acta Numerica*, 24:259–328, 2015.

[18] M. Ganesh and S. C. Hawkins. A high performance computing and sensitivity analysis algorithm for stochastic many-particle wave scattering. *SIAM J. Sci. Comput.*, 37: A1475–A1503, 2015.

[19] F. Y. Kuo and D. Nuyens. Application of quasi-monte carlo methods to elliptic pdes with random diffusion coefficients: A survey of analysis and implementation. *Foundations of Computational Mathematics*, 16(6):1631–1696, 2016.

[20] M. Ganesh and S. C. Hawkins. An offline/online algorithm for a class of stochastic multiple obstacle configurations in half-plane. *J. Comp. Appl. Math.*, 307:52–64, 2016.

[21] M. Ganesh, B. Reyes, and A. Purkayastha. An FEM-MLMC algorithm for a moving shutter diffraction in time stochastic model. *Discrete Cont. Dyn.-B*, 24:257–272, 2019.

[22] M. Ganesh, S. C. Hawkins, and D. Volkov. An efficient algorithm for a class of stochastic forward and inverse Maxwell models in $R^3$. *J. Comput. Phys.*, 398:10881, 2019.

[23] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method, 2nd Edition*, volume 77. Wiley, 2009.

[24] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., 2005.

[25] G. Fishman. A first course in monte carlo. *Journal of the American Statistical Association*, 102:758–758, 2007.

[26] S. Joe and F. Y. Kuo. Constructing sobol sequences with better two-dimensional projections. *SIAM. J. Sci. Comput.*, 30(5):2635–2654, 2008.

[27] H. Niederreiter and F. Pillichshammer. Construction algorithms for good extensible lattice rules. *Constructive Approximation*, 30(3):361–393, 2009.

[28] F. Y. Kuo, I. H. Sloan, and H. Woniakowski. Lattice rule algorithms for multivariate approximation in the average case setting. *Journal of Complexity*, 24(2):283–323, 2008.

[29] F. Y. Kuo and D. Nuyens. A practical guide to quasi-monte carlo methods, 2016.

[30] J. Dick and F. Pillichshammer. Multivariate integration in weighted hilbert spaces based on walsh functions and weighted sobolev spaces. *Journal of Complexity*, 21(2):149–195, 2005.

[31] D. Nuyens and R. Cools. Fast component-by-component construction, a reprise for different kernels. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pages 373–387. Springer Berlin Heidelberg, 2006.

[32] J. Dick. Walsh spaces containing smooth functions and quasi-monte carlo rules of arbitrary high order. *SIAM J. Numerical Analysis*, 46:1519–1553, 2008.

[33] T. Goda, K. Suzuki, and T. Yoshiki. Digital nets with infinite digit expansions and construction of folded digital nets for quasi-monte carlo integration. *Journal of Complexity*, 33:30–54, 2016.

[34] J. Dick, F. Y. Kuo, and I. H. Sloan. High-dimensional integration: The quasi-monte carlo way. *Acta Numerica*, 22:133–288, 2013.

[35] M. Ganesh, F.Y. Kuo, and I.H. Sloan. Quasi-monte carlo finite element analysis for wave propagation in heterogeneous random media. *SIAM/ASA J. Uncertainty Quantification*, 9: 106–134, 2021.

[36] S. M. Allen and J. W. Cahn. A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta Metallurgica*, 27(6):1085–1095, 1979.

[37] S. Bartels. *Numerical Methods for Nonlinear Partial Differential Equations*. Springer, New York, 2015.

[38] M. Cheng and J. A. Warren. An efficient algorithm for solving the phase field crystal model. *Journal of Computational Physics*, 227(12):6241–6248, 2008.

[39] M. Alfaro and D. Hilhorst. Generation of interface for an allen-cahn equation with nonlinear diffusion. *Math. Model. Nat. Phenom.*, 5(5):1–12, 2009.

[40] M. Benes, V. Chalupecky, and K. Mikula. Geometrical image segmentation by the allen-cahn equation. *Applied Numerical Mathematics*, 51(2):187–205, 2004.

[41] L. C. Evans, H. M. Soner, and P. E. Souganidis. Phase transitions and generalized motion by mean curvature. *Communications on Pure and Applied Mathematics*, 45(9):1097–1123, 1992.

[42] X. Feng and A. Prohl. Numerical analysis of the allen-cahn equation and approximation for mean curvature flows. *Numerische Mathematik*, 94(1):33–65, 2003.

[43] T. Ohtsuka. Motion of interfaces by an allen-cahn type equation with multiple-well potentials. *Asymptotic Analysis*, 56:87–123, 2008.

[44] Y. Li, H. G. Lee, D. Jeong, and J. Kim. An unconditionally stable hybrid numerical method for solving the allen–cahn equation. *Computers and Mathematics with Applications*, 60(6): 1591–1606, 2010.

[45] D. C. Antonopoulou, G. Karali, and A. Millet. Existence and regularity of solution for a stochastic cahn-hilliard/allen-cahn equation with unbounded noise diffusion. *Journal of Differential Equations*, 260(3):2383–2417, 2016.

[46] E. N. M. Cirillo, N. Ianiro, and G. Sciarra. Allen-cahn and cahn-hilliard-like equations for dissipative dynamics of saturated porous media. *Journal of the Mechanics and Physics of Solids*, 61(2):629 – 651, 2013.

[47] M. Katsoulakis, G. Kossioris, and O. Lakkis. Noise regularization and computations for the 1-dimensional stochastic allen-cahn problem. *Interfaces and Free Boundaries*, 9:1–28, 2011.

[48] A. del Campo, G. Garcia-Calderon, and J. G. Muga. Quantum transients. *Phys. Reports*, 476:1–50, 2009.

[49] M. Moshinsky. Diffraction in time. *Phys. Rev.*, 88(3):625–631, 1952.

[50] A. Goussev. Diffraction in time: An exactly solvable model. *Phys. Review A*, 87:1–4, 2012.

[51] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686 – 707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045.

[52] M. Raissi, P. Perdikaris, and G. Karniadakis. Inferring solutions of differential equations using noisy multi-fidelity data. *Journal of Computational*, 335:736 – 746, 2017. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2017.01.060.

[53] C. Rao, H. Sun, and Y. Liu. Physics informed deep learning for computational elastodynamics without labeled data, 2020.

[54] A. Atay, O.and Doncic and J. M. Skotheim. Switch-like transitions insulate network motifs to modularize biological networks. *Cell Systems*, 3(2):121–132, 2019/07/17 2016. doi: 10.1016/j.cels.2016.06.010. URL `https://doi.org/10.1016/j.cels.2016.06.010`.

[55] B. Schoeberl, C. Eichler-Jonsson, E. D. Gilles, and G. Müller. Computational modeling of the dynamics of the map kinase cascade activated by surface and internalized egf receptors. *Nature Biotechnology*, 20(4):370–375, 2002.

[56] E. Klipp and W. Liebermeister. Mathematical modeling of intracellular signaling pathways. *BMC Neuroscience*, 7(1):S10, 2006.

[57] D. Gilbert, M. Heiner, R. Breitling, and R. Orton. *Computational Modelling of Kinase Signalling Cascades*, pages 369–384. Humana Press, Totowa, NJ, 2010.

[58] J. Gunawardena. Time-scale separation – Michaelis and Menten's old idea, still bearing fruit. *The FEBS Journal*, 281(2):473–488, 2014. doi: 10.1111/febs.12532. URL `https://febs.onlinelibrary.wiley.com/doi/abs/10.1111/febs.12532`.

[59] G. Craciun and M. Feinberg. Multiple equilibria in complex chemical reaction networks: I. the injectivity property. *SIAM Journal on Applied Mathematics*, 65(5):1526–1546, 2005.

[60] G. Shinar and M. Feinberg. Concordant chemical reaction networks and the species-reaction graph. *Mathematical biosciences*, 241, 08 2012. doi: 10.1016/j.mbs.2012.08.002.

[61] E. Feliu and C. Wiuf. A computational method to preclude multistationarity in networks of interacting species. *Bioinformatics*, 29(18):2327–2334, 07 2013. ISSN 1367-4803. doi: 10.1093/bioinformatics/btt400.

[62] C. Conradi, E. Feliu, M. Mincheva, and C. Wiuf. Identifying parameter regions for multistationarity. *PLoS Computational Biology*, 13(10):1–25, 10 2017. doi: 10.1371/journal.pcbi.1005751.

[63] E. Feliu and A. Sadeghimanesh. Kac-Rice formulas and the number of solutions of parametrized systems of polynomial equations, 2020.

[64] B. B. Edelstein. Biochemical model with multiple steady states and hysteresis. *Journal of Theoretical Biology*, 29(1):57 – 62, 1970. ISSN 0022-5193. doi: https://doi.org/10.1016/0022-5193(70)90118-9. URL http://www.sciencedirect.com/science/article/pii/0022519370901189.

[65] M. Feinberg. Lectures on chemical reaction networks. notes of lectures given at the Mathematics Research Center, University of Wisconsin. *https://crnt.osu.edu/LecturesOnReactionNetworks*, 1979.

[66] M. Terzer. *Large scale methods to enumerate extreme rays and elementary modes*. Lulu Press, Incorporated, 2009.

[67] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1991.

[68] C. H. Schilling, D. Letscher, and B. Ø. Palsson. Theory for the systemic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective. *Journal of Theoretical Biology*, 203(3):229 – 248, 2000. ISSN 0022-5193. doi: https://doi.org/10.1006/jtbi.2000.1073.

[69] E. Murabito, E. Simeonidis, K. Smallbone, and J. Swinton. Capturing the essence of a metabolic network: A flux balance analysis approach. *Journal of Theoretical Biology*, 260 (3):445 – 452, 2009. ISSN 0022-5193. doi: https://doi.org/10.1016/j.jtbi.2009.06.013.

[70] R. M. T. Fleming, I. Thiele, G. Provan, and H. P. Nasheuer. Integrated stoichiometric, thermodynamic and kinetic modelling of steady state metabolism. *Journal of Theoretical Biology*, 264(3):683—692, June 2010. ISSN 0022-5193. doi: 10.1016/j.jtbi.2010.02.044.

[71] R. Seydel. *Practical Bifurcation and Stability Analysis*. Interdisciplinary Applied Mathematics. Springer, New York, 2009. ISBN 9781441917409.

[72] C. Chicone. *Ordinary Differential Equations with Applications*. Texts in Applied Mathematics. Springer, New York, 2006. ISBN 9780387307695.

[73] J. A. Egea, D. Henriques, T. Cokelaer, A. F. Villaverde, A. MacNamara, D.-P. Danciu, J. R. Banga, and J. Saez Rodriguez. Benchmarking optimization methods for parameter estimation in large kinetic models. *Bioinformatics*, 35(5):830–838, 2019.

[74] I. Otero-Muras, J. R. Banga, and A. A. Alonso. Characterizing multistationarity regimes in biochemical reaction networks. *PLoS ONE*, 7(7):e39194, 2012. doi: 10.1371/journal.pone.0039194.

[75] Y. Xiang, D. Y. Sun, W. Fan, and X. G. Gong. Generalized simulated annealing algorithm and its application to the Thomson model. *Physics Letters A*, 233(3):216 – 220, 1997. ISSN 0375-9601. doi: https://doi.org/10.1016/S0375-9601(97)00474-X.

[76] Y. Xiang and X. Gong. Efficiency of generalized simulated annealing. *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, 62:4473–6, 10 2000. doi: 10.1103/PhysRevE.62.4473.

[77] Y. A. Kuznetsov. *Elements of applied bifurcation theory*. Springer, New York, 2 edition, 1998.

[78] R. E. Moore. *Reliability in computing: the role of interval methods in scientific computing*. Academic Press Professional Inc, San Diego, 1988.

[79] A. F. Villaverde, F. Frölich, D. Weindl, J. Hasenauer, and Banga J. R. Benchmarking optimization methods for parameter estimation in large kinetic models. *Bioinformatics*, 35 (5):830–838, 2019.

[80] H. P. J. Bolton, A. A. Groenwold, and J. A. Snyman. The application of a unified Bayesian stopping criterion in competing parallel algorithms for global optimization. *Computers and Mathematics with Applications*, 48(3):549 – 560, 2004. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2003.09.030. URL http://www.sciencedirect.com/science/article/pii/S0898122104840768.

[81] J. A. Snyman and L. P. Fatti. A multi-start global minimization algorithm with dynamic search trajectories. *Journal of Optimization Theory and Applications*, 54(1):121–141, 1987. doi: 10.1007/BF00940408. URL https://doi.org/10.1007/BF00940408.

[82] Wolfram Research, Inc. Mathematica, Version 12.0. URL https://www.wolfram.com/mathematica.

[83] E. J. Doedel, R. C. Paenroth, A. R. Champneys, and T. F. Fairgrieve. Auto 2000 : Continuation and bifurcation software for ordinary differential equations (with homcont). *Technical report, California Institute of Technology*, 1997.

[84] E. T. Somogyi, J-M. Bouteiller, J. A. Glazier, M. Konig, J. K. Medley, M. H. Swat, and H. M. Sauro. libRoadRunner: a high performance SBML simulation and analysis library. *Bioinformatics*, 31(20):3315–3321, 06 2015. ISSN 1367-4803. doi: 10.1093/bioinformatics/btv363. URL https://doi.org/10.1093/bioinformatics/btv363.

[85] S. Feng, M. Sáez, C. Wiuf, E. Feliu, and O. Soyer. Core signalling motif displaying multistability through multi-state enzymes. *Journal of the Royal Society Interface*, 13, 10 2016. doi: 10.1098/rsif.2016.0524.

[86] A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, and H. Kitano. Celldesigner 3.5: A versatile modeling tool for biochemical networks. *Proceedings of the IEEE*, 96(8):1254–1265, 8 2008. ISSN 0018-9219. doi: 10.1109/JPROC.2008.925458.

[87] B. J. Bornstein, S. M. Keating, A. Jouraku, and M. Hucka. Libsbml: An api library for sbml. *Bioinformatics*, 24(6), 2008.

[88] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[89] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, Š. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, jan 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL `https://doi.org/10.7717/peerj-cs.103`.

[90] J-F. Hervagault and S. Canu. Bistability and irreversible transitions in a simple substrate cycle. *Journal of Theoretical Biology*, 127(4):439 – 449, 1987. ISSN 0022-5193. doi: https://doi.org/10.1016/S0022-5193(87)80141-8.

[91] F. Ortega, J. L. Garcés, F. Mas, B. N. Kholodenko, and M. Cascante. Bistability from double phosphorylation in signal transduction. *The FEBS Journal*, 273(17):3915–3926, 2006. doi: 10.1111/j.1742-4658.2006.05394.x.

[92] S. Burhenne, D. Jacob, and G. Henze. Sampling based on sobol' sequences for monte carlo techniques applied to building simulations. *Proceedings of Building Simulation 2011: 12th Conference of International Building Performance Simulation Association*, pages 1816–1823, 2011.

[93] I. M. Sobol. Uniformly distributed sequences with an additional uniform property. *USSR Computational Mathematics and Mathematical Physics*, 16(5):236–242, 1976.

[94] T. Hansen and G. L. Mullen. Primitive polynomials over finite fields. *Mathematics of Computation*, 59(200):639–643, 1992.

[95] J. Dick. Explicit constructions of quasi-monte carlo rules for the numerical integration of high-dimensional periodic functions. *SIAM Journal on Numerical Analysis*, 45(5): 2141–2176, 2007.

[96] T. Goda. Fast construction of higher order digital nets for numerical integration in weighted sobolev spaces. *Numerical Algorithms*, volume 69, 2013.

[97] M. B. Giles and B. J. Waterhouse. Multilevel quasi-monte carlo path simulation. *Advanced Financial Modelling Radon Series on Computational and Applied Mathematics*, 8:1–18, 2009.

[98] F. Y. Kuo, R. Scheichl, C. Schwab, I. H. Sloan, and E. Ullmann. Multilevel quasi-monte carlo methods for lognormal diffusion problems. *Mathematics of Computation*, 86(308): 2827–2860, 2017.

[99] M. Ganesh and B. Reyes. An efficient multi-level high-order algorithm for simulation of a class of allen–cahn stochastic systems. *Journal of Computational and Applied Mathematics*, 401:113765, 2022. ISSN 0377-0427. doi: https://doi.org/10.1016/j.cam.2021.113765.

[100] J. Dick, F. Y. Kuo, Q. T. Le Gia, and C. Schwab. Multilevel higher order qmc petrov-galerkin discretization for affine parametric operator equations. *SIAM J. NUMER. ANAL.*, 54:2541–2568, 2016.

[101] Z. Romanowski. Application of h-adaptive, high order finite element method to solve radial schrodinger equation. *Molecular Physics*, 107(13):1339–1348, 2009.

[102] A. Nissen, G. Kreiss, and M. Gerritsen. High order stable finite difference methods for the schrodinger equation. *J. Sci. Comput*, 55:173–199, 2013.

[103] T. Kimura, N. Sato, and S. Iwata. Application of the higher order finite-element method to one-dimensional schrodinger equation. *J. Comput. Chem.*, 9:827–835, 1998.

[104] T. E. Lee, M. J. Baines, and S. Langdon. A finite difference moving mesh method based on conservation for moving boundary problems. *J. Comput. Appl. Math.*, 288:1–17, 2015.

[105] C. J. Budd, W. Huang, and R. D. Russell. Adaptivity with moving grids. *Acta Numerica*, 18:111–241, 2009.

[106] J. W. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods*. Texts in Applied Mathematics. Springer New York, 1998. ISBN 9780387979991.

[107] P. Ŝolín. *Partial Differential Equations and the Finite Element Method*. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. Wiley, 2005. ISBN 9780471764090.

[108] D. Di Pietro, A. Ern, and L. Formaggia. *Numerical Methods for PDEs. State-of the Art Numerical Techniques*. 06 2018. ISBN 978-3-319-94675-7. doi: 10.1007/978-3-319-94676-4.

[109] H. Lee and I. S. Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1):110–131, 1990. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(90)90007-N.

[110] I.E. Lagaris, A.C. Likas, and D.G. Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5): 1041–1049, 2000. doi: 10.1109/72.870037.

[111] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8.

[112] A. Baydin, B. Pearlmutter, A. Radul, and J. Siskind. Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 18:1–43, 04 2018.

[113] R. Van Der Meer, C. Oosterlee, and A. Borovykh. Optimally weighted loss functions for solving pdes with neural networks, 2020.

[114] M. A. Nabian, R. J. Gladstone, and H. Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 36(8):962–977, 2021. doi: https://doi.org/10.1111/mice.12685.

[115] C. L. Wight and J. Zhao. Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks. *Communications in Computational Physics*, 29(3): 930–954, 2021. ISSN 1991-7120. doi: https://doi.org/10.4208/cicp.OA-2020-0086.

[116] S. Wang, X. Yu, and P. Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *ArXiv*, abs/2007.14527, 2020.

[117] S. Wang, Y. Teng, and P. Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *CoRR*, abs/2001.04536, 2020.

[118] I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998. doi: 10.1109/72.712178.

[119] F. Sultana, A. Sufian, and P. Dutta. Advancements in image classification using convolutional neural network. *CoRR*, abs/1905.03288, 2019. URL `http://arxiv.org/abs/1905.03288`.

[120] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE Access*, 7:19143–19165, 2019. doi: 10.1109/ACCESS.2019.2896880.

[121] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[122] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[123] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[124] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989. doi: 10.1007/BF01589116.

[125] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979. ISSN 00401706. URL http://www.jstor.org/stable/1268522.

[126] R. Sjögren A. D. Lee and D. Svensson. Pydoe2. URL https://github.com/clicumu/pyDOE2.

[127] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. JMLR Workshop and Conference Proceedings, 13–15 May 2010.

[128] P. Bochev and M. Gunzburger. *Least-Squares Finite Element Methods*, volume 3. 01 2006. ISBN 978-0-387-30888-3. doi: 10.1007/b13382.

[129] L. D. McClenny and U. M. Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. *CoRR*, abs/2009.04544, 2020.

[130] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL http://www.scipy.org/.

[131] M. Ainsworth and J. Dong. Galerkin neural networks: A framework for approximating variational equations with error control. *SIAM Journal on Scientific Computing*, 43(4), 6 2021.

[132] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. Felipe Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal. Deep complex networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1T2hmZAb.

[133] M. A. Mercioni and S. Holban. P-swish: Activation function with learnable parameters based on swish activation function in deep learning. In *2020 International Symposium on Electronics and Telecommunications (ISETC)*, pages 1–4, 2020. doi: 10.1109/ISETC50328.2020.9301059.

[134] H. Bateman. Some recent researches on the motion of fluids. *Monthly Weather Review*, 43: 163–170.

[135] J.M. Burgers. A mathematical model illustrating the theory of turbulence. 1:171 – 199, 1948. ISSN 0065-2156. doi: https://doi.org/10.1016/S0065-2156(08)70100-5.

[136] T. Nagatani. Density waves in traffic flow. *Phys. Rev. E*, 61:3564–3570, Apr 2000. doi: 10.1103/PhysRevE.61.3564. URL https://link.aps.org/doi/10.1103/PhysRevE.61.3564.

[137] D. E. Panayotounakos and D. Drikakis. On the closed-form solutions of the wave, diffusion and burgers equations in fluid mechanics. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 75 (6):437–447, Jan 1995. ISSN 0044-2267. doi: 10.1002/zamm.19950750604.

[138] M. Vergassola, B. Dubrulle, U. Frisch, and A. Noullez. Burgers' equation, devil's staircases and the mass distribution for large-scale structures. *Astronomy and Astrophysics*, 289:325–356, 08 1994.

# APPENDIX A

# FORMAL DESCRIPTIONS OF A CHEMICAL REACTION NETWORK

In this appendix we provide a more formal description of a chemical reaction network, which is based of the Feinberg lectures [65]. Consider the chemical reaction network in Figure A.1. Note that the reactions in this toy example seem unrealistic in a chemical sense. These reactions are termed ”funny” reactions by Feinberg and are a result of open reactors having an almost identical construction to those in closed systems. For this reason, we allow for these “funny” reactions and refer the reader to the lectures in which Feinberg explains the different ways we can represent open systems as “funny” reaction networks. The diagram presented depicts a reaction network with mass action kinetics. Here, it is important to note that the kinetics that we will consider throughout our work will be mass action kinetics only. Although this is the case, as stated in [65], certain aspects of this theory can also work for an arbitrary kinetics.

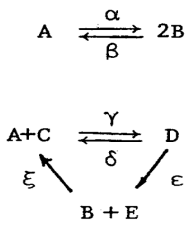



Figure A.1: Toy chemical reaction network.

Referring back to Figure A.1, in this reaction network we have the the following chemical species: A, B, C, D, and E. Where a molecule of A decomposes into two molecules of B with rate $\alpha$, two molecules of B react to form one molecule of A with rate $\beta$, a molecule of A can react with a molecule of C to form a molecule of D with rate $\gamma$, a molecule of D can decompose into a molecule of A and a molecule of C with rate $\delta$, a molecule of D can decompose into a molecule of B and a molecule of E with rate $\epsilon$, and a molecule of B can react with a molecule of E to form a molecule of A and a molecule of C with rate $\xi$. For notational convenience we will use short hand

notation for these reactions. For example, when A decomposes into two molecules of B, we will write this as $A \to 2B$.

For all reaction network we consider, we will presume that we throw various amounts of each of the species into a container. We then require that the contents of this container remain spatially homogeneous for all of time and the contents are maintained at a fixed temperature and volume. Within this setting, we are then interested in the temporal evolution of the concentrations for each of these species, we denote these instantaneous values of the molar concentrations of these species as the functions $c_A(t), c_B(t), c_C(t), c_D(t)$, and $c_E(t)$, where $t$ denotes a dependence on time. Thus, the derivate of $c_A(t)$ with respect to time, will describe the evolution of the molar concentration (we will often refer to this just as concentration) of species $A$, we denote this as $\dot{c}_A$. Assuming an instantaneous occurrence rate for each of the reactions in the network, we can then begin to build our systems of ODEs. To start, let us first consider the temporal evolution of the concentration of species A. To form an expression for $\dot{c}_A$ we must consider every reaction that involves A and ask the question, "Are we losing or gaining molecules of A?". If we are losing molecules of A, this will be reflected in the expression as a negative number of the molecules of A that we lost and if we gain molecules of A, this will be reflected in the expression as an addition of the number of molecules of A that we gained. If we refer to the diagram of the network above, we see that in the reaction $A \to 2B$ we lose a molecule of B at a rate of $\alpha$. For this reason, one term in $\dot{c}_A$ will be $-\alpha c_A$. The reaction $2B \to A$ is a little different, we are gaining a molecule of A with a rate of $\beta$, for this reason this term will be positive, however, this gain of a molecule of A is only possible if two molecules of B encounter each other. To reflect this encounter in $\dot{c}_A$, we will add $\beta(c_B)^2$, if three molecules of B needed to encounter each other, then we would add $\beta(c_B)^3$. The last situation we need to consider are reactions such as $A + C \to D$, where a molecule of A must meet a molecule of C to form a molecule of D. For this situation we will say that the probability of this encounter is proportional to $c_A c_C$. Thus, $A + C \to D$ will contribute $-\gamma c_A c_C$ to the expression for $\dot{c}_A$.

Using this framework for the formation of the differential equations, we obtain the temporal derivative of each of the species' concentrations as below. Where $\alpha, \beta, \gamma, \epsilon, \delta$, and $\xi$ are the positive rate constants for the corresponding reactions.

$$\begin{aligned}
\dot{c}_A &= -\alpha c_A + \beta(c_B)^2 - \gamma c_A c_C + \delta c_D + \xi c_B c_E \\
\dot{c}_B &= 2\alpha c_A - 2\beta(c_B)^2 + \epsilon c_D - \xi c_B c_E \\
\dot{c}_C &= -\gamma c_A c_C + \delta c_D + \xi c_B c_E \\
\dot{c}_D &= \gamma c_A c_C - (\delta + \epsilon) c_D \\
\dot{c}_E &= \epsilon c_D - \xi c_B c_E.
\end{aligned}$$

Further descriptions of this ODE system can be made. We will now provide these descriptions. To begin, we will define those sets that a reaction network is composed of. The first of these is the set of chemical species which will be denoted as $\mathcal{S}$, for our example we have $\mathcal{S} = \{A, B, C, D, E\}$. Second, is the set of objects that appear before and after the reaction arrows, these objects are called the complexes of the network and the set of complexes will be denoted as $\mathcal{C}$, for our example we have $\mathcal{C} = \{A, 2B, A + C, D, B + E\}$. Lastly, we have the set of reactions, denoted as $\mathcal{R}$, for our example, we have

$\mathcal{R} = \{A \to 2B, 2B \to A, A + C \to D, D \to A + C, D \to B + E, B + E \to A + C\}$. Given these sets will define a reaction network, it is appropriate to require that all of these sets be finite dimensional.

Now that we know that these sets are finite, we need to establish what types of vector spaces these sets can live in. First, we will denote $\mathbb{R}$ as the real numbers, $\mathbb{P}$ as the positive real numbers, and $\overline{\mathbb{P}}$ as the non-negative real numbers. We then say that for a set $\mathcal{I}$, $\mathbb{R}^{\mathcal{I}}$ is the vector space of real-valued functions with domain $\mathcal{I}$. Similarly, $\mathbb{P}^{\mathcal{I}}$ are those functions with domain $\mathcal{I}$ that produce positive values and $\overline{\mathbb{P}}^{\mathcal{I}}$ are those functions with domain $\mathcal{I}$ that produce non-negative values. It is in these vector spaces where we will choose to define our finite sets (choices which we will provide later on). For example, we let the concentrations of the species be given by a vector $c \in \overline{\mathbb{P}}^{\mathcal{S}}$ because we only want to consider non-zero and positive concentration values for our species.

Since we are working in a vector space, we then need to define some basic operations in these vectors spaces. If $x$ is a vector in $\mathbb{R}^{\mathcal{I}}$, then $x_i$ will denote that the number $x$ assigns to $i \in \mathcal{I}$. Note that the subscript in $x_i$ does not denote an index in the traditional way. For example, if $c \in \overline{\mathbb{P}}^{\mathcal{S}}$, where $\mathcal{S}$ is the set from our example, then $c_A$ would be the concentration of species $A$ which is non-negative. Now, if $x$ and $y$ are vectors in $\mathbb{R}^{\mathcal{I}}$ then the statement $xy$ will denote a vector in $\mathbb{R}^{\mathcal{I}}$ such that

$$(xy)_i = x_i y_i, \ \ \forall i \in \mathcal{I}.$$

For the same vectors $x$ and $y$ we then define the standard scalar product in $\mathbb{R}^{\mathcal{I}}$ as follows

$$x \cdot y = \sum_{i \in \mathcal{I}} x_i y_i.$$

The support of $x \in \mathbb{R}^{\mathcal{I}}$ will define the subset of $\mathcal{I}$ whose elements are assigned non-zero values of $x$. In a more mathematical notation we have

$$\text{supp } x := \{i \in \mathcal{I} : x_i \neq 0\}.$$

For example, if $c \in \overline{\mathbb{P}}^{\mathcal{S}}$ then supp $c$ will denote the set of all species that have positive concentrations. The standard basis for the vector space $\mathbb{R}^{\mathcal{I}}$ will be given as the set

$$\{\omega_j \in \mathbb{R}^{\mathcal{I}} : j \in \mathcal{I}\}.$$

Here $\omega_j$ is a column vector that is the length of the number of elements of $\mathcal{I}$ and has zero elements everywhere except for the $j^{\text{th}}$ element. For example, using the set $\mathcal{S}$ we have established, $\omega_A$ can be given as follows:

$$\omega_A = [1, 0, 0, 0, 0]^T.$$

Note here that we are assuming some order of the set $\mathcal{S}$, we are stating here that the element $A$ is the first element of the set $\mathcal{S}$. Thus, the standard basis for the vector space $\mathbb{R}^{\mathcal{S}}$ is

$$\{[1,0,0,0,0]^T, [0,1,0,0,0]^T, [0,0,1,0,0]^T, [0,0,0,1,0]^T, [0,0,0,0,1]^T\}.$$

At this point, it is important to note that we often use these basis vectors to our advantage when we are referring to the set of chemical species $\mathcal{S}$. For example, for the set $\mathcal{S} = \{A, B, C, D, E\}$, we will consider the symbol $A + B$ as an abbreviation for $\omega_A + \omega_B \in \mathbb{R}^{\mathcal{S}}$. By using this notation we can consider the adding, subtracting, multiplication by a constant, and scalar product of a complex. By making the complex a vector, we are also permitted to consider the support of a complex, which is just the set of species that appear in that complex. In addition to this, if we have a complex $y \in C$, the component $y_s$, $s \in \mathcal{S}$ corresponds to the stoichiometric coefficient of species $s$ in complex $y$. For our example, if $y = 2B$, then $y_B = 2$, where as $y_s = 0, \forall s \in \mathcal{S}\backslash B$. With all of this notation now established, we can begin to state formal definitions of a reaction network. The first definition we consider is a chemical reaction network.

**Definition A.9.** Chemcial reaction network
A chemical reaction network consists of three sets:

1. a finite set $\mathcal{S}$, elements of which are called the species of the network.
2. a finite set $C$ of distinct vectors in $\overline{\mathbb{P}}^{\mathcal{S}}$ such that

$$\bigcup_{y \in C} \text{supp } y = \mathcal{S}.$$

   Elements of $C$ are called the complexes of the network.

3. a relation $\mathcal{R} \subset C \times C$ such that

   (a) $y \to y \notin \mathcal{R}, \forall y \in C$.

(b) for each $y \in C$ there exists a $y' \in C$ such that

$y' \to y \in \mathcal{R}$ or such that $y \to y' \in \mathcal{R}$.

Elements of $\mathcal{R}$ are called the reactions of the network. Here, $y \to y'$ denotes that complex $y$ reacts to complex $y'$. Where $y$ is called the reactant complex and $y'$ is the product complex of the reaction $y \to y'$.

In this definition, 2. implies that elements in $\mathcal{S}$ appear in at least one complex, 3a. implies that a complex does not have a reaction with itself, and 3b. implies that no complex is isolated. next we establish the kinetics for a reaction network.

**Definition A.10.** A Kinetics

A kinetics for a reaction network $\{\mathcal{S}, C, \mathcal{R}\}$ is an assignment of each reaction $y \to y' \in \mathcal{R}$ of a continuous rate function $\mathcal{K}_{y\to y'} : \overline{\mathbb{P}}^{\mathcal{S}} \to \overline{\mathbb{P}}$ such that

$$\mathcal{K}_{y\to y'}(c) > 0 \text{ if and only if } \operatorname{supp} y \subset \operatorname{supp} c.$$

This definition says that the reaction $y \to y'$ proceeds at non-zero rate if and only if the species appearing in the reactant complex are actually present in the reactor (i.e. container). Given we will only be dealing with mass action kinetics, we will now provide a formal definition of a mass action kinetics.

**Definition A.11.** A Mass Action Kinetics

A kinetics $\mathcal{K}$ for a reaction network $\{\mathcal{S}, C, \mathcal{R}\}$ is mass action if, for each reaction $y \to y' \in \mathcal{R}$, there exists a positive number $k_{y\to y'}$ known as the rate constant of a reaction $y \to y'$ such that

$$\mathcal{K}_{y\to y'}(c) = k_{y\to y'} \prod_{s\in\mathcal{S}} c_s^{y_s}.$$

This established kinetics then gives rise to a vector for of the differential equations that can be expressed as follows:

$$\dot{\mathbf{c}} = \sum_{y\to y' \; in\mathcal{R}} k_{y\to y'} \Big[ \prod_{s\in\mathcal{S}} c_s^{y_s} \Big] (y' - y).$$

This definition of the temporal evolution of the species concentration provides one unique insight, $\dot{c}$ must take on values in the stoichiometric subspace for the underlying reaction network, where the stoichiometric subspace is as given in the definition below.

**Definition A.12.** Stoichiometric Subspace

A stoichiometric subspace for a reaction network $\{\mathcal{S}, \mathcal{C}, \mathcal{R}\}$ is the linear subspace $S \subset \mathbb{R}^{\mathcal{S}}$ defined by

$$S := \text{span}\{y' - y \in \mathbb{R}^{\mathcal{S}} : y \to y' \in \mathcal{R}\}.$$

We can then define the size of this stoichiometric subspace by defining the rank of the network as given below.

**Definition A.13.** Rank of a network

The rank of a reaction network $\{\mathcal{S}, \mathcal{C}, \mathcal{R}\}$ will be denoted as $\bar{s}$ where $\bar{s} = \dim S$.

The idea of a stoichiometric subspace then gives rise to the notion of stoichiometric compatability between two compositions, say $c$ and $c'$. This is given in the next definition.

**Definition A.14.** Stoichiometrically Compatible

Let $\{\mathcal{S}, \mathcal{C}, \mathcal{R}\}$ be a reaction network with stoichiometric subspace $S \subset \mathbb{R}^{\mathcal{S}}$. Two vectors $c \in \overline{\mathbb{P}}^{\mathcal{S}}$ and $c' \in \overline{\mathbb{P}}^{\mathcal{S}}$ are stoichiometrically compatible if $c' - c$ lies in $S$. Stoichiometric compatibility is an equivalence relation that induces a partition of $\overline{\mathbb{P}}^{\mathcal{S}}$ into equivalence classes called the stoichiometric compatibility classes for a network. In particular, the stoichiometric compatibility class containing $c \in \overline{\mathbb{P}}^{\mathcal{S}}$ is the set $(c + S) \cap \overline{\mathbb{P}}^{\mathcal{S}}$.

This notion is proposed because the theory established by Feinberg answers the question of whether or not the differential equations for a reaction network can admit multiple positive equilibria within a stoichiometric compatability class. Using the definition of stoichiometrically compatible, answering this question will tell us if there are two or more positive equilibria which are stoichiometrically compatible with the same initial composition. The deficiency of the network is a non-negative integer that will help us classify if the given network has only one

stable equilibrium value. To begin this discussion of deficiency, we will need to point out some graphical aspects of a reaction network. They are defined as follows.

**Definition A.15.** Linked Complexes

Two complexes $y \in C$ and $y' \in C$ are directly linked if $y \to y'$ or if $y' \to y$. If $y$ and $y'$ are directly linked we write $y \leftrightarrow y'$. These two complexes are then linked if any of the following conditions are satisfied:

1. $y = y'$
2. $y \leftrightarrow y'$
3. $C$ contains a sequence $\{y_1, y_2, \ldots, y_k\}$ such that $y \leftrightarrow y_1 \leftrightarrow y_2 \leftrightarrow \cdots \leftrightarrow y_k \leftrightarrow y'$.

If $y$ and $y'$ are linked we write $y \sim y'$. The equivalence relation $\sim$ induces a partition of $C$ into a family of equivalence classes called the linkage classes of the network. We will let $\ell$ denote the number of linkage classes of a network. We then denote the linkage classes as $L^\theta$ for $\theta = 1, 2, \ldots, \ell$.

For the example we have been discussing, we have $\ell = 2$, where $L^1 = \{A, 2B\}$ and $L^2 = \{A + C, D, B + E\}$.

**Definition A.16.** Ultimately Reacts to

A complex $y \in C$ ultimately reacts to a complex $y' \in C$ if any of the following conditions are satisfied:

1. $y = y'$
2. $y \to y'$
3. $C$ contains a sequence $\{y_1, y_2, \ldots, y_k\}$ such that $y \to y_1 \to y_2 \to \cdots \to y_k \to y'$.

If $y$ ultimately reacts to $y'$ we write $y => y'$. The two complexes are strongly linked if both $y => y'$ and $y' => y$. If $y$ and $y'$ are strongly linked, we write $y \approx y'$. The equivalence relation $\approx$

induces a partition of $C$ into a family of equivalence classes called the strong linkage classes of the network. We will let $\rho$ denote the number of strong linkage classes of a network. We then denote the strong linkage classes as $\Lambda^\theta$ for $\theta = 1, 2, \ldots, \rho$.

From this definition, one can see that every strong linkage class lies within a linkage class. More specifically, every linkage class is the disjoint union of strong linkage classes. We can provide even further classification of a strong linkage class by defining a terminal strong linkage class, as given below.

**Definition A.17.** Terminal Strong linkage class
A terminal strong linkage class is a strong linkage class $\Lambda$ with the property that no complex in $\Lambda$ reacts to a complex not in $\Lambda$; that is, $y \in \Lambda$ and $y \to y'$ imply that $y'$ is in $\Lambda$. We denote the number of terminal strong linkage classes in a network as $\tau$.

From this statement it is implied that for any network, each linkage class contains at least one terminal strong linkage class. These definitions then provide us with the more important classification of a network, weakly reversible.

**Definition A.18.** Weakly Reversible
A reaction network is weakly reversible if any of the following equivalent conditions are satisfied:

1. whenever $y => y'$ we also have $y' => y$
2. whenever $y \to y'$ we also have $y' => y$
3. each linkage class is a strong linkage class
4. each linkage class is a terminal strong linkage class

A more strict form of weakly reversible is reversible. A network is reversible only if $y \to y'$ and $y' \to y$. All of these definitions finally bring us to one of the most important definitions when paired with weakly reversible, that is deficiency. Deficiency is defined as follows.

**Definition A.19.** Deficiency

The deficiency of a reaction network is defined by the positive integer, $\delta$, given by

$$\delta := N - \ell - s,$$

where $N$ is the number of complexes, $\ell$ is the number of linkage classes, and $s$ is the rank of the network.

Using the definitions and notation we have developed we can finally state the deficiency zero and one theorems, which are the heart of Feinberg's work. They are as follows.

**Theorem 4** (The Deficiency Zero Theorem)**.** Let $\{\mathcal{S}, C, \mathcal{R}\}$ be any reaction network of deficiency zero.

1. If the network is not weakly reversible then, for arbitrary kinetics $\mathcal{K}$, the differential equations for the reaction system $\{\mathcal{S}, C, \mathcal{R}\}$ cannot admit a positive equilibrium (i.e. an equilibrium in $\mathbb{P}^{\mathcal{S}}$).

2. If the network is not weakly reversible then, for arbitrary kinetics $\mathcal{K}$, the differential equations for the reaction system $\{\mathcal{S}, C, \mathcal{R}\}$ cannot admit a cyclic composition trajectory containing a positive composition (i.e. a point in $\mathbb{P}^{\mathcal{S}}$).

3. If the network is weakly reversible (in particular, if the network is reversible) then, for any mass action kinetics $k \in \mathbb{P}^{\mathcal{R}}$, the differential equations for the mass action system $\{\mathcal{S}, C, \mathcal{R}, k\}$ have the following properties: There exists within each positive stoichiometric compatibility class precisely one equilibrium; that equilibrium is asymptotically stable; and there cannot exist a nontrivial cyclic composition trajectory in $\mathbb{P}^{\mathcal{S}}$.

**Theorem 5** (The Deficiency One Theorem)**.** Let $\{\mathcal{S}, C, \mathcal{R}\}$ be any reaction network with $\ell$ linkage classes. Let $\delta$ denote the deficiency of the network; let $\delta_\theta$ denote the deficiency of the $\theta^{\text{th}}$ linkage class, $\theta = 1, 2, \ldots, \ell$; and suppose that both of the following conditions are satisfied:.

1. $\delta_\theta \leq 1, \ \ \theta = 1, 2, \ldots, \ell$

2. $\delta = \sum_{\theta=1}^{\ell} \delta_\theta$.

If the network is weakly reversible (in particular, if the network is reversible) then, for any mass action kinetics $k \in \mathbb{P}^{\mathcal{R}}$, the differential equations for the mass action system $\{\mathcal{S}, \mathcal{C}, \mathcal{R}, k\}$ admit precisely one equilibrium in each positive stoichiometric compatibility class.

# APPENDIX B

## ALLEN-CAHN EXTRA PLOTS

In this Appendix, we present additional simulation plots corresponding to numerical experiments described in Section 4.2.3. Figures Figure B.1 and Figure B.2 further demonstrate the evolution of more initial interfaces for the 2D and 3D AC models, respectively.

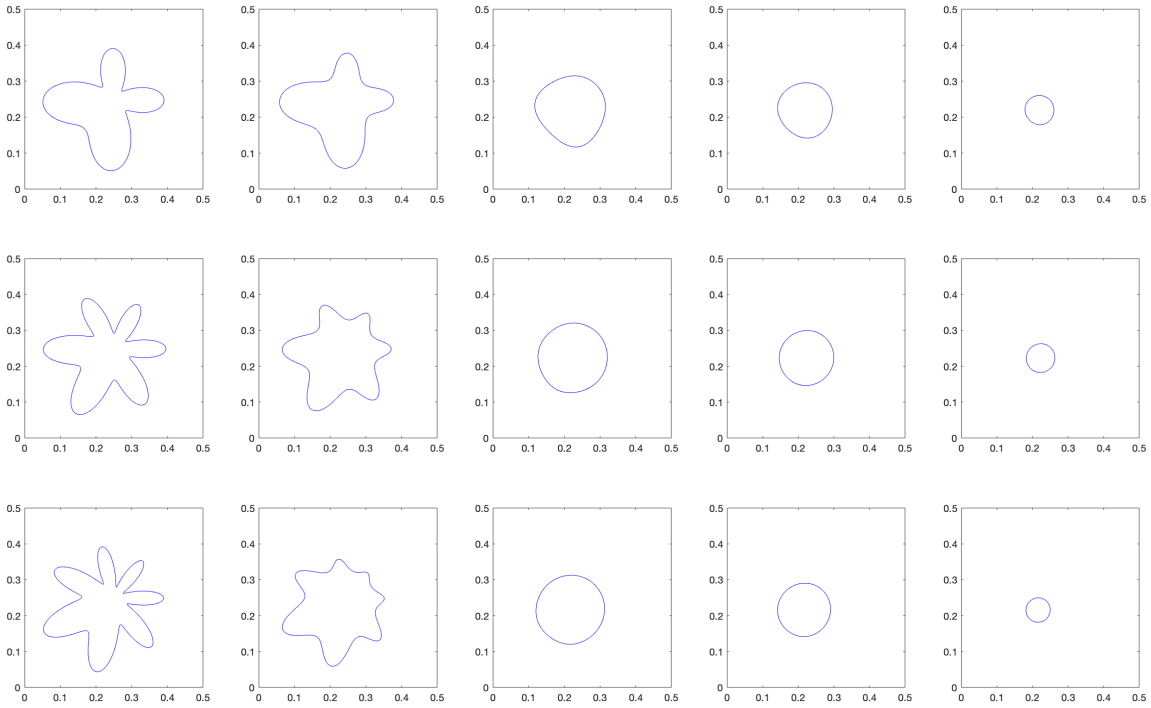


Figure B.1: 2D M-C-N approximate A-C system solutions with $L_{AC} = 16$, $\eta = 4$ and $\epsilon_{AC} \approx 0.5063$ (top row), $\eta = 6$ and $\epsilon_{AC} \approx 0.7430$ (middle row), and $\eta = 7$ and $\epsilon_{AC} \approx 0.8513$ (bottom row).

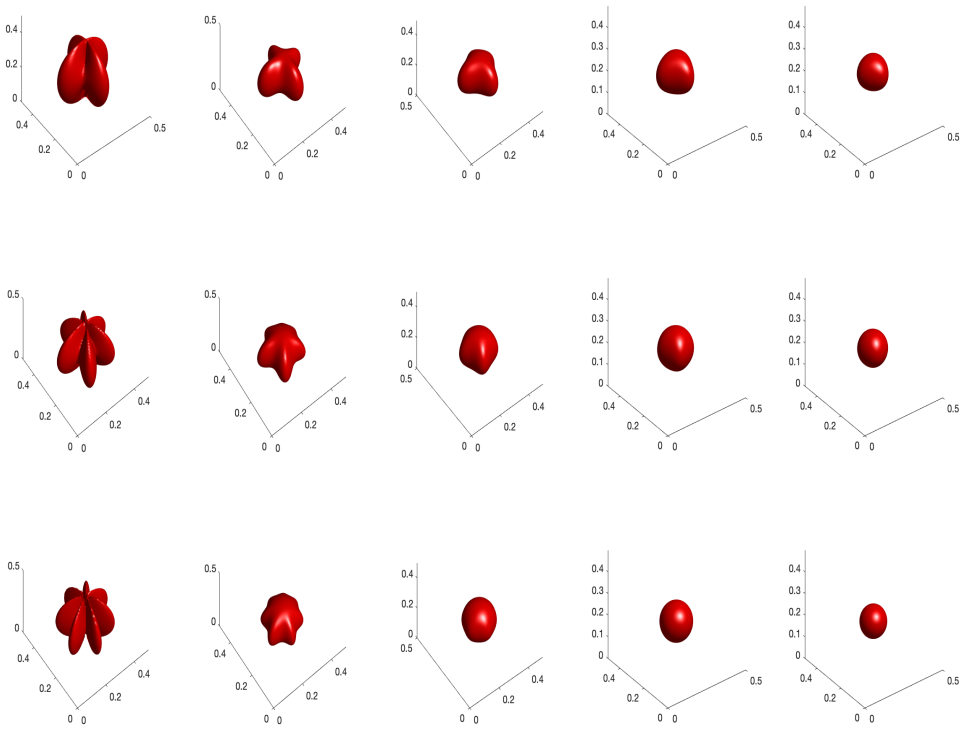


Figure B.2: 3D M-C-N A-C system solutions with $L_{AC} = 8$, $\eta = 4$ and $\epsilon_{AC} \approx 0.5084$ (top row), $\eta = 6$ and $\epsilon_{AC} \approx 0.6876$ (middle row), and $\eta = 7$ and $\epsilon_{AC} \approx 0.5465$ (bottom row).

# APPENDIX C

## STANDARD PINN VS. SCALED PINN USING BURGER'S EQUATION

In this appendix section we investigate the solution of Burger's equation in one space dimension using the standard PINN and scaled PINN approach. The Burger's equation was first introduced in 1915 by Harry Bateman [134] and expanded upon by Johannes Burgers in 1948 [135]. The Burger's equation is a partial differential equation that is a simplification of the Navier-Stokes equation. Since its introduction, the Burger's equation has found its way into multiple applications such as traffic flow [136], gas dynamics [137], and cosmology [138]. The Burger's equation in one dimension is provided by (C.1), where $x$ is the spatial coordinate, $t$ is the temporal coordinate, $u(x,t)$ is the speed of the fluid at a specified $x$ and $y$, $v$ is the viscosity of the fluid, and $F(x,t)$ is an imposed forcing function of the PDE.

$$\begin{aligned} &\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial^2 u}{\partial x^2} = F(x,t), \quad x \in [-1,1], \quad t \in [0,1], \\ &u(x,0) = \sin(\pi x), \\ &u(-1,t) = u(1,t) = 0. \end{aligned} \tag{C.1}$$

Based off of the construction of (C.1), one can formulate an exact solution to the problem. Specifically, if one chooses the exact solution of the PDE to be $u(x,t) = e^{-t}sin(\pi x)$, then the initial and boundary conditions of (C.1) are satisfied and the forcing function $F(x,t)$ is provided as (C.2).

$$F(x,t) = -e^{-t}sin(\pi x) - ve^{-t}\pi^2 sin(\pi x) + e^{-2t}\pi cos(\pi x)sin(\pi x) \tag{C.2}$$

Using the exact solution $u(x,t) = e^{-t}sin(\pi x)$ and $v = (-0.01/\pi)$, we will now compare the solution of (C.1) against its PINN approximation. To provide the Neural Network solution, we approximate $u(x,t)$ with a MLP $\mathcal{NN}(\mathbf{r};\boldsymbol{\theta}) : \mathbb{R}^2 \to \mathbb{R}$ that consists of 9 hidden-layers each with 20 nodes that contain biases and activation functions specified by $f(\mathbf{r}) = tanh(\mathbf{r})$, where $\mathbf{r} = [x,t]^T$. The output node of this neural network will contain one node with a bias and activation function

*tanh*. To train the Neural Network we use the loss function specified by (C.3), making $\mathcal{NN}(\mathbf{r};\boldsymbol{\theta})$ a PINN.

$$\mathcal{L}(\boldsymbol{\theta}) = MSE_F(\boldsymbol{\theta}) + MSE_0(\boldsymbol{\theta}) + MSE_{DB}(\boldsymbol{\theta}), \tag{C.3}$$

The terms of (C.3) are provided by (C.4) where $\mathcal{J}(\mathcal{NN}(\mathbf{r};\boldsymbol{\theta}))$ is the nonlinear differential operator specified by (C.5), (C.4a) ensures that the main PDE of (C.1) is satisfied by using collocation points $x_{F,i}, t_{F,i}$ for $i = 1, \ldots, N_F$ where $\mathbf{r}_{F,i} = [x_{F,i}, t_{F,i}]^T$, (C.4c) makes sure that the initial condition of (C.1) is satisfied by using initial condition points $x_{0,i}$ for $i = 1, \ldots, N_0$ where $\mathbf{r}_{0,i} = [x_{0,i}, 0]^T$, and (C.4b) makes sure that the boundary conditions of (C.1) are satisfied by using boundary condition points $x_{DB,i}, t_{DB,i}$ for $i = 1, \ldots, N_{DB}$ where $\mathbf{r}_{DB,i} = [x_{DB,i}, t_{DB,i}]^T$.

$$MSE_F = \frac{1}{N_F}\sum_{i=1}^{N_F}\Big(\mathcal{J}(\mathcal{NN}(\mathbf{r}_{F,i};\boldsymbol{\theta})) - F(x_{F,i}, t_{F,i})\Big)^2 \tag{C.4a}$$

$$MSE_{DB} = \frac{1}{N_{DB}}\sum_{i=1}^{N_{DB}}\Big(\mathcal{NN}(\mathbf{r}_{DB,i};\boldsymbol{\theta}) - u(x_{DB,i}, t_{DB,i})\Big)^2, \tag{C.4b}$$

$$MSE_0 = \frac{1}{N_0}\sum_{i=1}^{N_0}\Big(\mathcal{NN}(\mathbf{r}_{0,i};\boldsymbol{\theta}) - u(x_{0,i}, 0)\Big)^2. \tag{C.4c}$$

$$\mathcal{J}(\mathcal{NN}(\mathbf{r};\boldsymbol{\theta})) = \frac{\partial \mathcal{NN}(\mathbf{r};\boldsymbol{\theta})}{\partial t} + \mathcal{NN}(\mathbf{r};\boldsymbol{\theta})\frac{\partial \mathcal{NN}(\mathbf{r};\boldsymbol{\theta})}{\partial x} + \nu\frac{\partial^2 \mathcal{NN}(\mathbf{r};\boldsymbol{\theta})}{\partial x^2} \tag{C.5}$$

To obtain the points $x_{F,i}, t_{F,i}$ for $i = 1, \ldots, N_F$, $x_{0,i}$ for $i = 1, \ldots, N_0$, and $x_{DB,i}, t_{DB,i}$ for $i = 1, \ldots, N_{DB}$ we utilize LHS. Specifically, we generate the initial condition points $x_{0,i}$ using LHS with input dimension of 1 and $N_0 = 200$ and set the time points to zero, the boundary condition points $x_{DB,i}$ are constructed by creating a column vector of 100 points on both the left and right boundary and then we concatenate these two column vectors to form a vector of length 200, the corresponding points $t_{DB,i}$ are then created using LHS with input dimension of 1 and $N_0 = 200$, and the interior collocation points $x_{F,i}, t_{F,i}$ are created by generating LHS with input dimension of 1 and $N_f = 10000$ for both the $x_{F,i}$ and $t_{F,i}$ points.

Using the framework we have specified for the MLP, the weights $\boldsymbol{\theta}$ are found by minimizing the loss function (C.3) using first the L-BFGS optimizer [124] with 50 maximum iterations and then

subsequently with the Adam optimizer [123] with a learning rate of 1e-7. Completing this process for 1000 epochs and utilizing the error provided by (C.6) evaluated at evenly spaced points $x_{e,i}$ for $i = 1, \ldots, 351$, we obtain the predicted solution of the MLP at $t = 0.0, 0.5$, and 1.0 as provided in Figure C.1. In addition to this, we extrapolate the solution beyond the learned interval of $t \in [0, 1]$, this provides the extrapolated solution at $t = 1.01, 2.25$, and 3.5 as provided in Figure C.2.

$$\mathbf{e}(\mathbf{x}_e, t) = \mathcal{NN}([\mathbf{x}_e, t]^T; \boldsymbol{\theta}) - u(\mathbf{x}_e, t) \tag{C.6}$$

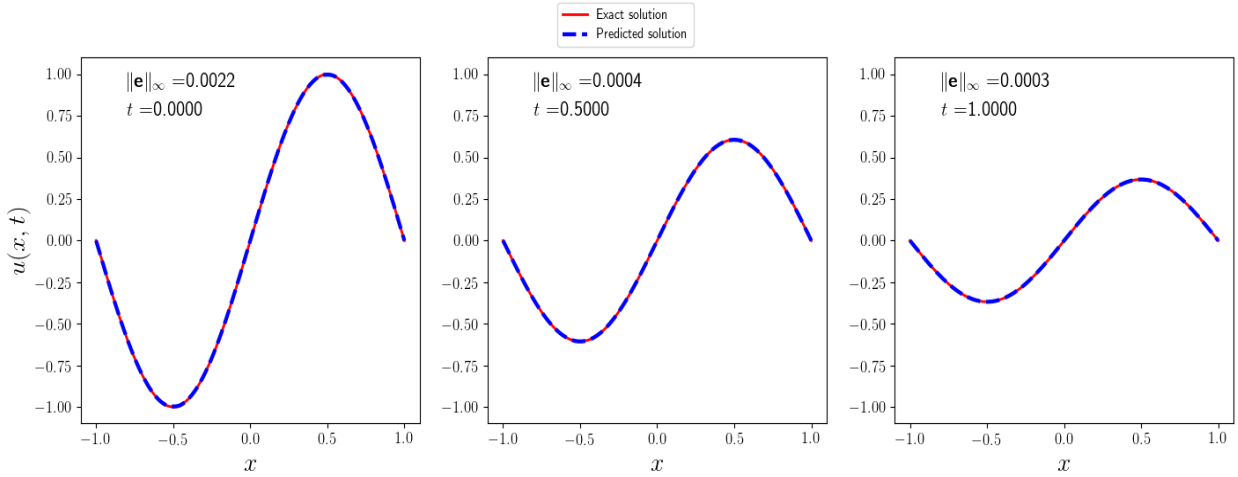



Figure C.1: Exact vs Neural Network prediction for Burger's equation.

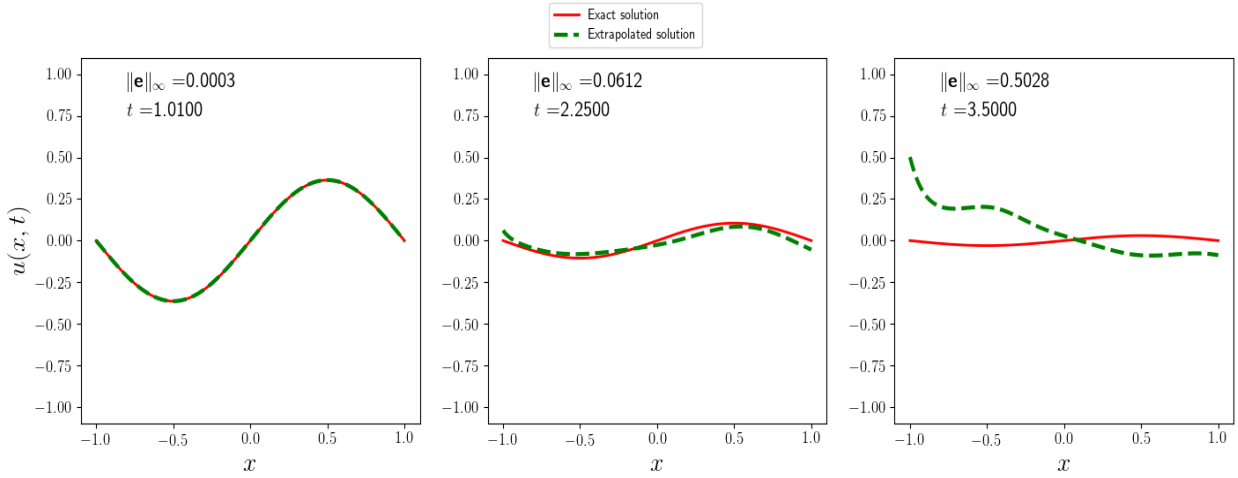



Figure C.2: Exact vs Neural Network extrapolation for Burger's equation.

From Figure C.2 we see that the extrapolated solution quickly begins to deviate from the true solution. This deviation can be caused by an uneven focus on particular parts of the loss function.

For example, the loss function provided by (C.3) may primarily try to optimize the term (C.4a), rather than the initial or boundary conditions. This seems to be the case as $t = 0$ has a larger error than $t = 0.5$ and $t = 1.0$, as seen in Figure C.1. To provide more consistent minimization on each of the loss function's portions, it is suggested in [113] to apply a scaling parameter $\lambda \in (0, 1)$ to the loss function as in (C.7). We demonstrate the effectiveness of this modified loss function using the Burger's equation, letting $\lambda = 0.2$, and using the same framework as previously established for the non-modified loss function. This provides the predicted solution and extrapolated solutions in Figure C.3 and Figure C.4, respectively. From these results, it is clear that this modified loss function provides more consistent results and an improved extrapolated solution of the MLP.

$$\mathcal{L}(\boldsymbol{\theta}) = \lambda MS\, E_F(\boldsymbol{\theta}) + (1-\lambda) MS\, E_0(\boldsymbol{\theta}) + (1-\lambda) MS\, E_{DB}(\boldsymbol{\theta}), \tag{C.7}$$

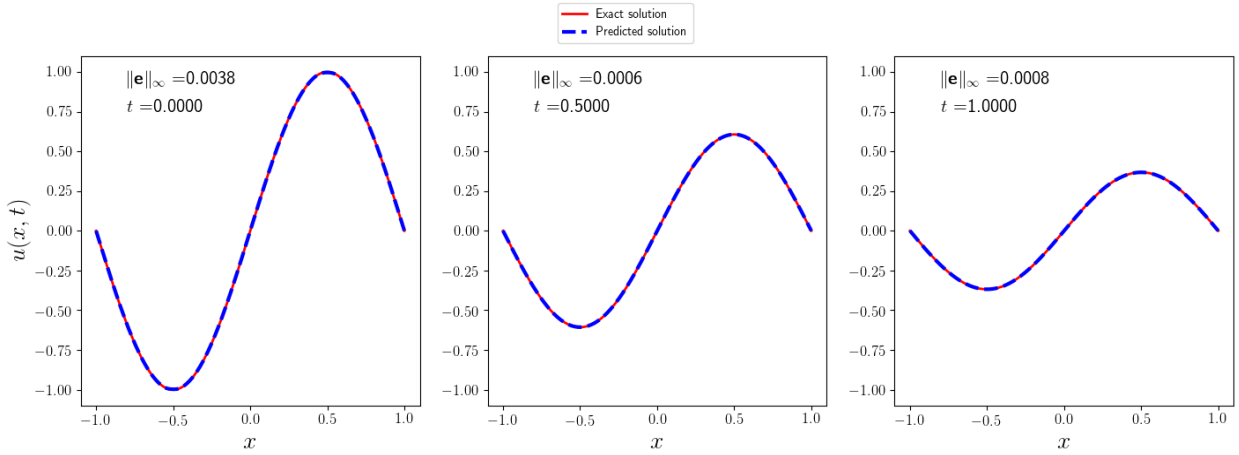



Figure C.3: Exact vs Neural Network prediction for Burger's equation for loss function (C.7) with $\lambda = 0.2$.

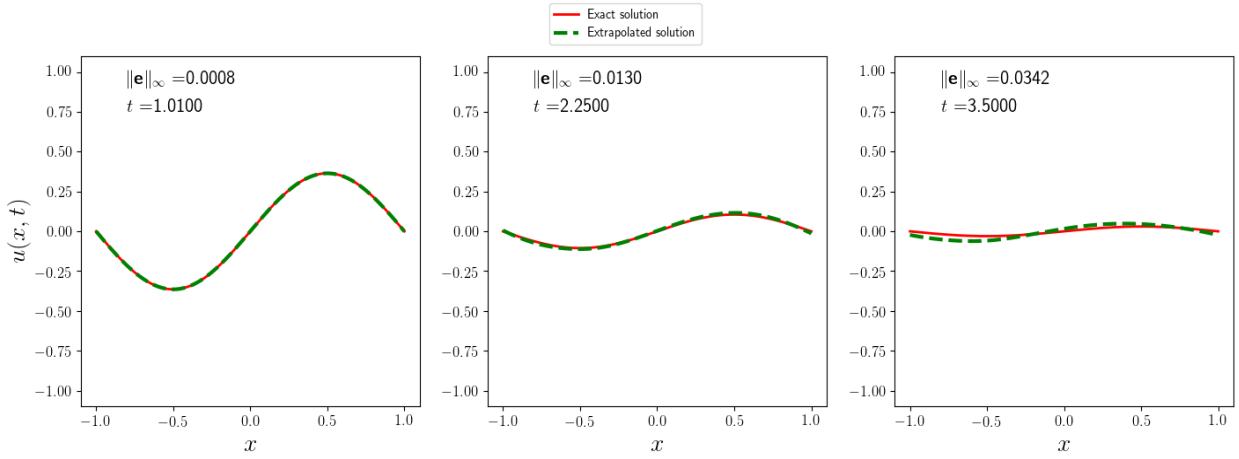



Figure C.4: Exact vs Neural Network extrapolation for Burger's equation for loss function (C.7) with $\lambda = 0.2$.

# APPENDIX D

# COMPARISON OF SCALED PINN AGAINST FEM

In this Appendix section, we compare the Scaled PINN solution against an implicit second-order in space-and-time Crank-Nicolson linear Galerkin spline finite element method. Again, we will be considering the solution of the Heat Equation in 2D as provided in section 6.3 for $c = 1$ and $T = 0.5$. For comparison against traditional numerical techniques, we will consider the FEM solution of the heat equation by using an implicit second-order in space-and-time Crank-Nicolson linear Galerkin spline FEM. A majority of the details for constructing this numerical method can be found in Section 4.2.1, the differences here are that we instead solve the heat equation for all times, rather than including an intermediary step as in the A–C solution. Thus, we have the final representation of the solution in matrix-vector form:

$$[B + \frac{\Delta t}{4}A]\underset{\sim}{U}^k = [B - \frac{\Delta t}{4}A]\underset{\sim}{U}^{k-1} \ , \ k = 0, \ldots, K-1$$

where $\underset{\sim}{U}^0 = U^0_{m,n}$ (initial condition), $B = [B^x \otimes B^y]$, $A = [A^x \otimes B^y] + [B^x \otimes A^y]$, and $\underset{\sim}{U}^{k-1}, \underset{\sim}{U}^k$ are vectors of length $N^2$. As mentioned previously, one can see that the $A$ and $B$ matrices constructed are symmetric and positive definite. Thus, we can create an efficient procedure to compute the solution using the Cholesky Factorization. For the scaled PINN approximation of $u(x, y, t)$ we will use the MLP $\mathcal{NN}(\mathbf{r}; \boldsymbol{\theta}) : \mathbb{R}^3 \to \mathbb{R}$ with 10 hidden-layers each with 20 nodes that contain biases and activation functions specified by $f(\mathbf{r}) = tanh(\mathbf{r})$, where $\mathbf{r} = [x, y, t]^T$. The output node of this neural network will contain one node with a bias and linear function.

Using the framework we have specified for the MLP, we will use the choice of collocation points of section 6.3.1. Similarly, we minimize the loss function (6.14) with $\lambda = 1000$ using the L-BFGS optimizer with 20,000 iterations. The FEM solution is then constructed using $h = \Delta t = 0.025$. These solutions produce the results in Table D.1. The number of values needed in the FEM solution are determined by the number of nonzero elements in the Cholesky Factorization, R, and

the number of nonzero elements in $[B - \frac{\Delta t}{4}A]$. The number of values needed in the PINN solution are found by adding up the number of elements needed to construct each layer in the MLP (including both the input and output layers). From Table D.1 one can see that a PINN can produce a better maximum absolute error using about a nineteenth of the memory required by the FEM.

Table D.1: Comparing the FEM solutions vs. the Scaled PINN solution with $\lambda = 1000$.

| Method | Number of Values Needed | $Err_{max}$ |
|---|---|---|
| FEM | 74,026 | 2.0698e-3 |
| Scaled PINN | 3,881 | 1.3372e-3 |

# APPENDIX E

## COPYRIGHT DOCUMENTATION FOR CHAPTER 4

Terms and conditions

These terms and conditions (“Terms and Conditions”) apply to your use of all Elsevier websites, applications, services and products (“Services” or individually a “Service”) that post a link to these Terms and Conditions and that are provided by any Elsevier group company worldwide (”Elsevier”, ”we”, ”us” or ”our”).

By accessing or using any of the Services, you agree to be bound by these Terms and Conditions. These Terms and Conditions expressly incorporate by reference and include the Service’s Privacy Policy and any guidelines, policies or additional terms or disclaimers that may be posted and/or updated on the Service or on notices that are sent to you. If you do not agree with these Terms and Conditions, please do not use the Services.

Using our services

Unless otherwise set out herein, content comprised within the Services, including text, graphics, user interfaces, visual interfaces, photographs, trademarks, logos, videos, audio, images, applications, programs, computer code and other information (collectively, the ”Content”), including but not limited to the design, layout, “look and feel” and arrangement of such Content, is owned by Elsevier, its licensors or its content providers and is protected by copyright, trademark and other intellectual property and unfair competition laws.

RELX and the RE symbol are trade marks of RELX Group plc, used under license.

Except as otherwise provided in any additional terms for a Service, you may print or download Content from the Services for your own personal, non-commercial, informational or scholarly use, provided that you keep intact all copyright and other proprietary notices.

You may not copy, display, distribute, modify, publish, reproduce, store, transmit, post, translate or create other derivative works from, or sell, rent or license all or any part of the Content, or

products or services obtained from the Services, in any medium to anyone, except as otherwise expressly permitted under these Terms and Conditions, or any relevant license or subscription agreement or authorization by us.

You may not reverse engineer, disassemble, decompile or translate any software in the Content, or otherwise attempt to derive the source code of such software, except to the extent expressly permitted under applicable law, without our prior written permission. You may not engage in the systematic retrieval of Content from the Services to create or compile, directly or indirectly, a collection, compilation, database or directory without our prior written permission.

You may not use any robots, spiders, crawlers or other automated downloading programs, algorithms or devices, or any similar or equivalent manual process, to: (i) continuously and automatically search, scrape, extract, deep link or index any Content; (ii) harvest personal information from the Services for purposes of sending unsolicited or unauthorized material; or (iii) cause disruption to the working of the Services or any other person's use of the Services. If the Services contain robot exclusion files or robot exclusion headers, you agree to honor them and not use any device, software or routine to bypass them. You may not attempt to gain unauthorized access to any portion or feature of the Services, any other systems or networks connected to the Services or to any Elsevier server, or any of the products or services provided on, accessed from or distributed through the Services. You may not probe, scan or test the vulnerability of the Services or any network connected to the Services or breach or attempt to breach the security or authentication measures on the Services or any network connected to the Services.

You may not, without the approval of Elsevier, use the Services to publish or distribute any advertising, promotional material, or solicitation to other users of the Services to use any goods or services. For example (but without limitation), you may not use the Services to conduct any business, to solicit the performance of any activity that is prohibited by law, or to solicit other users to become subscribers of other information services. Similarly, you may not use the Services to download and redistribute public information or shareware for personal gain or distribute multiple copies of public domain information or shareware.

Content provided to us or posted on or through the service

Some of our Services allow you to upload, submit, store, send or receive content. Except as otherwise provided in any additional terms or agreements for or relating to a Service, we do not claim ownership of any content, application or other material that you or third parties provide to us (including feedback and suggestions) or post, upload, input or submit on or through the Services, including our blog pages, message boards, comment or discussion features, chat rooms and forums, for review by Elsevier and its suppliers or licensors, the general public, registered users of the Services, or by the members of any public or private community (”Submission”), and you retain ownership of any intellectual property rights that you hold in the Submission. We are not responsible for the content, accuracy or compliance with relevant laws or regulations of any Submission that contains content that is not owned by Elsevier. However, by posting, uploading, inputting, providing or submitting (”Posting”) a Submission, and except to the extent that any Submission is to be treated as confidential as part of our policies, you grant us (and those we work with) a royalty-free, perpetual, irrevocable, worldwide, non-exclusive right and license to publish, post, reformat, index, archive, make available, link to, and otherwise use such Submission in all forms and media (whether now known or later developed), in connection with the operation of our respective businesses (including, without limitation, the Services) and to permit others to do so. You shall not make any Submission anonymously, and you warrant and represent that any such Submission is original, and has been written by you. We are under no obligation to display or otherwise use any Submission you may provide, and we may remove any Submission at any time in our sole discretion. By Posting such a Submission, you also warrant and represent that you own or otherwise control all of the rights to your Submission including, without limitation, all the rights necessary for granting the permission specified above.

You may not use the Services to publish or distribute any information (including software or other content) which is illegal, which violates or infringes upon the rights of any other person, which is defamatory, abusive, hateful, profane, pornographic, threatening or vulgar, which contains errors, viruses or other harmful components, or which is otherwise actionable at law. Elsevier may at any

time exercise editorial control over the Content of the Services.

We shall have the right, but not the obligation, to monitor Submissions to determine compliance with these Terms and Conditions and any operating rules we establish and to satisfy any law, regulation or authorized government request. We shall have the right in our sole discretion to edit, refuse to post, or remove any Submission.

Notification of infringement

Notifications regarding any alleged intellectual property infringement should be directed to the Elsevier Legal Department by email addressed to DMCA@elsevier.com.

Links

The Services may contain links to third-party sites or resources. We do not endorse and are not responsible or liable for any content, advertising, products or other materials on or available from external sites or resources linked to the Services. Transactions that occur between you and any third party are strictly between you and the third party and are not the responsibility of Elsevier. Elsevier welcomes you to link to the Services. You may establish a hypertext link to the Services, provided that the link does not state or imply any sponsorship or endorsement of your site, company, product and/or service by Elsevier. You may not use on your site any trademarks, service marks or copyrighted materials appearing on the Services, including but not limited to any logos, without the express written consent or an explicit license of the owner of the mark or right. You may not frame or otherwise incorporate into another site any of the Content or other materials on the Services without prior written consent of Elsevier.

Disclaimer of warranties and liability

We provide the Services using a commercially reasonable level of skill and care, but neither Elsevier nor its suppliers or licensors make any specific promises about the Services, including the Content or any Submission therein. For example, we don’t make any commitment that the operation of the Services will be uninterrupted or error-free; that any defects will be corrected; that the Services, including the server that makes them available, are free of viruses or other harmful components; or as to the accuracy, completeness, reliability, availability, suitability,

quality, non-infringement, operation or result obtained from the use of any Content or Submission provided on, accessible from or distributed through the Services.

ELSEVIER AND ITS SUPPLIERS AND LICENSORS PROVIDE THE SERVICES AND ALL CONTENT AND ANY SUBMISSION INCLUDED IN OR ACCESSIBLE FROM THE SERVICES ”AS IS” AND WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND (EXPRESS, IMPLIED AND STATUTORY, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF TITLE AND NONINFRINGEMENT AND THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE), ALL OF WHICH ELSEVIER AND ITS SUPPLIERS AND LICENSORS DISCLAIM TO THE FULLEST EXTENT PERMITTED BY LAW. YOUR USE OF THE SERVICES AND ALL CONTENT AND SUBMISSIONS, INCLUDED IN OR ACCESSIBLE FROM THE SERVICES ARE AT YOUR SOLE RISK.

To the extent permitted under applicable law, neither Elsevier nor its suppliers and licensors assume responsibility for any injury and/or damage to persons, animals or property as a matter of products liability, malpractice, failure to warn, negligence or otherwise, or from any use or operation of any ideas, instructions, methods, tests, products or procedures displayed on the Services or incorporated in the Content or any Submission included in or accessible from the Services. Practitioners and researchers must rely on their own experience, knowledge and judgment in evaluating or applying any information, which remains their professional responsibility. Because of rapid advances in the medical sciences and changes in government regulations and clearances, we recommend that independent verification of diagnoses, treatments, indications choice of drugs and drug dosages should be made. Discussions, views, and recommendations expressed may not be considered absolute and universal for every situation. Elsevier remains neutral with regard to jurisdictional claims in published maps and institutional affiliations. Elsevier will not be liable for the failure by any user of the Services, Content or Submission to use due care in the use and validation of results made available through the Services or included in the Content or any Submission, nor will Elsevier be liable for any medical

treatment provided by users to their patients, whether or not the Services, Content or Submission were used in connection with such treatment.

TO THE EXTENT PERMITTED UNDER APPLICABLE LAW, IN NO EVENT SHALL ELSEVIER OR ITS SUPPLIERS OR LICENSORS BE LIABLE FOR ANY DAMAGES (INCLUDING, WITHOUT LIMITATION, CONSEQUENTIAL, SPECIAL, INCIDENTAL, INDIRECT, OR SIMILAR DAMAGES, PERSONAL INJURY (INCLUDING DEATH), LOSS OF PROFITS, CORRUPTION OR LOSS OF DATA, BUSINESS INTERRUPTION OR ANY OTHER COMMERCIAL DAMAGES OR LOSSES) ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SERVICES OR THE CONTENT OR SUBMISSIONS, OR SHALL THE LIABILITY OF ELSEVIER OR ITS SUPPLIERS AND LICENSORS EXCEED A SUM EQUAL TO THE FEES PAID BY YOU HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Elsevier does not claim ownership, endorse or take responsibility for any third-party products, information, guidelines, materials or services that may be offered, advertised, provided or displayed on the Service or incorporated in the Content or any Submission contained on, accessible from or distributed through the Services.

Indemnification

You hereby agree to indemnify and hold Elsevier, its directors, officers, shareholders, predecessors, successors in interest, employees, agents, suppliers and licensors harmless from and against any and all third-party claims of liability, losses, damages and costs, including, without limitation, reasonable attorneys' or legal fees, arising out of or in connection with your violation of these Terms and Conditions, and your use of or inability to use any of the Services or the Content or Submissions.

Registered user grant of license

Some sites may require you to register. If registration is required, you agree to provide accurate and complete registration information. It is your responsibility to inform Elsevier of any changes to that information. Each registration is for a single individual only, unless specifically designated

otherwise on the registration page. Elsevier does not permit a) anyone other than you to use the sections requiring registration by using your name or password; or b) access through a single name being made available to multiple users on a network or otherwise. As a registered user of a Service, Elsevier grants to you a non-transferable, non-exclusive and revocable license to use the Service according to these Terms and Conditions. Except as expressly granted herein or in any other agreement you have with Elsevier, you acquire no right, title or license in the Service or any Content or Submission accessed from or incorporated in the Service.

Password use and security

If you use a password to access a Service, you must not reveal your password and must take reasonable steps to keep your password confidential and secure. You agree to immediately notify Elsevier if you become aware of or have reason to believe that there is any unauthorized use of your password or account or any other breach of security. Elsevier is in no way liable for any claims or losses related to the use or misuse of your password or account due to the activities of any third party outside of our control or due to your failure to maintain their confidentiality and security.

About these terms and conditions

Term and Termination: Any license granted to you to use any Service is effective until it expires, until Elsevier terminates it, or until you provide notice to Elsevier of your decision to terminate it. Your rights under the license will terminate automatically without notice to you if you fail to comply with any of the provisions of these Terms and Conditions. Elsevier reserves the right to suspend, discontinue or change a Service, or its availability to you, at any time without notice. Upon termination of the license to a Service, you shall cease all use of the Service.

Export Controls: Where the Service, its Content or any Submission consist of software that is subject to US, EU or any other Export Controls, you represent and warrant that (i) you are not located in a country that is subject to a U.S. Government, EU or other embargo, or that has been designated by the U.S. Government or the EU as a “terrorist supporting” country; and (ii) you are not listed on any U.S. Government, EU or any other list of prohibited or restricted parties.

No Assignment: You may not assign your rights or obligations under these Terms and Conditions to anyone.

No Waiver: Neither failure nor delay on the part of Elsevier to exercise or enforce any right, remedy, power or privilege hereunder nor course of dealing between the parties shall operate as a waiver thereof, or of the exercise of any other right, remedy, power or privilege. No part of these Terms and Conditions shall be deemed waived, and no breach consented to, unless such waiver or consent shall be in writing and signed by the party claimed to have waived or consented. No waiver of any rights or consent to any breaches shall constitute a waiver of any other rights or consent to any other breach.

Additional Terms: Additional or superceding terms and conditions may apply to purchases or supply of goods or services including intellectual property, to specific portions or features of a Service, and to subscriptions or licenses with institutions with which you may be employed or affiliated. If there is a conflict between these Terms and Conditions and the terms that are posted for or applicable to a specific portion of the Service, for any service offered on or through the Service, or set forth in an institutional subscription or license agreement, the latter terms shall control.

Compliance with Laws: You agree to comply with relevant laws and regulations that apply to your use of the Services, Content or any Submission.

Severability: If any provision in these Terms and Conditions is held invalid or unenforceable under applicable law, the remaining provisions shall continue in full force and effect.

Governing Laws and Venue: All matters relating to your access to or use of the Services, including all disputes, shall be governed by and construed in accordance with the laws of The Netherlands, without regard to its conflicts of law principles, except if you reside outside of the European Union, then the laws of the country of the Elsevier regional office in the region where you reside will apply. The exclusive jurisdiction and venue with respect to any action or suit arising out of or pertaining to the subject matter hereof shall be the courts of competent jurisdiction located in Amsterdam, The Netherlands, except if you reside outside of the European

Union, then the courts located in the country of the Elsevier regional office in the region where you reside will have exclusive jurisdiction. Any claim arising out of or in connection with your use of or inability to use the Service or the Content or any Submission must be brought within one (1) year after the event or such claim is barred.

Changes: Elsevier reserves the right to change, modify, add or remove portions of these Terms and Conditions at its sole discretion at any time and without prior notice. Please check these Terms and Conditions periodically for any modifications. Your continued use of any Service following the posting of any changes will mean that you have accepted and agreed to the changes.

# APPENDIX F

# COPYRIGHT DOCUMENTATION FOR CHAPTER 5

American Institute of Mathematical Sciences 2013.

References to “AIMS” mean the American Institute of Mathematical Sciences . References to “this website” are to the specific AIMS website from which the user is linking to this Copyright statement.

Please read the following Copyright statement carefully as your access to and use of this website will be deemed as acceptance of its terms. This Copyright statement was last updated on July 02, 2013.

Subject to subscriber status, reasonable amounts of AIMS copyright-protected material (other than logos) may be reproduced, distributed or communicated to others free of charge in any format or media for the purposes of research for a non commercial purpose or for private study or for internal circulation within an organization. This is subject to the material being reproduced accurately and not used in a misleading context. Where any AIMS copyright items on this website are being reproduced, distributed or communicated to others, the source of the material must be identified and the copyright status acknowledged. Systematic downloading of files is prohibited.

In particular, where a third party wishes to include AIMS copyright items on another website they must only use non-locked down content, link to the home page of this website rather than reproducing the items and fully credit this website and the copyright status of the item.

The permission to reproduce, distribute or communicate AIMS-protected material does not extend to anything which is identified as being the copyright of a third party. Authorization to reproduce, distribute or communicate such material must be obtained from the copyright owner concerned.

In addition, the names, images and logos identifying AIMS are proprietary marks of AIMS or its licensors. Other names, images and logos identifying third parties and their information, products and services are the proprietary marks of third parties.