Computational Modeling and Analysis of Cardiac Excitation

A Dissertation presented

by

Abhishek Murthy

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

August 2014

Stony Brook University

The Graduate School

Abhishek Murthy

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Radu Grosu - Dissertation Advisor
Research Professor, Department of Computer Science

Scott A. Smolka - Chairperson of Defense
Professor, Department of Computer Science

Allen R. Tannenbaum
Professor, Department of Computer Science

Flavio H. Fenton
Associate Professor, School of Physics, Georgia Institute of Technology

This dissertation is accepted by the Graduate School.

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Computational Modeling and Analysis of Cardiac Excitation

by

Abhishek Murthy

Doctor of Philosophy

in

Computer Science

Stony Brook University

2014

Modeling, analysis, and control of cardiac excitation, the biological process cardiac cells undergo on a periodic basis, is indispensable for understanding and countering life-threatening electrical disturbances of the heart, such as atrial and ventricular fibrillation. Modeling involves the derivation of a mathematical/computational representation of the dynamics of cardiac cells and their diffusivity. The key to effective modeling lies in striking a balance between i) *precision*, the ability of the model to replicate the underlying biological phenomena, and ii) *performance*, in terms of amenability to automated analysis techniques, such as simulation and formal verification.

The techniques of abstraction and compositionality have been instrumental in striking this balance. *Abstraction* is the process of removing unnecessary detail from a given model so that the resulting abstract version is both observationally equivalent to the original model and a conservative approximation of it, but better suited for the analysis of the properties of interest. Given a system consisting of a number of interacting components, also known as subsystems, *Compositionality* enables us to substitute a component by its equivalent abstraction, such that the overall system retains the properties of interest.

In the case of real-valued continuous-time dynamical systems, such as cardiac-cell models, a notion of *approximate equivalence* has been proposed, which can be used to show that an abstraction is approximately equivalent to the concrete model. Care must be taken, however, when a concrete model of a system component C

is replaced by an approximately equivalent abstraction C' within a *feedback loop*; in such situations, the approximation error between C and C' may get amplified. For feedback compositions, Antoine Girard has shown that Lyapunov-like *Bisimulation Functions* (BFs) satisfying a certain Small Gain Theorem can be used to establish substitutivity, which follows from i) *input-to-state stability* of the concrete and abstract subsystems, and ii) the *robustness* of the rest of the system to input deviations.

In this thesis, we first extend BFs to handle *input-to-output stability* and present two *Sum-of-Squares* formulations for automating the search for BFs. The automated proof technique, which has been implemented using MATLAB SOSTOOLS, enables *component-wise approximate model-order reduction of feedback-composed dynamical systems*. Using our techniques, we show that within a detailed 67-variable cardiac-cell model, the 13- and 10-variable Markovian subsystems for sodium and potassium channels can be safely (with bounded error) substituted by two-variable Hodgkin-Huxley-type models. The two-variable abstractions were identified using using a two-step curve fitting technique.

We then present the *Spiral Classification Algorithm* (SCA), a fast and accurate algorithm for computing the curvature of electrical waves and their associated breakup in cardiac tissues. Given a digitized frame of a propagating wave, SCA constructs a highly accurate representation of the front and the back of the wave, piecewise interpolates this representation with cubic splines, and subjects the result to an accurate curvature analysis. SCA has been applied to a number of representative types of spiral waves, and, for each type, a distinct curvature evolution in time (signature) has been identified.

Finally, we present explicit and online *Model-Predictive Controllers* for an excitable-cell simulator based on the nonlinear FitzHugh-Nagumo model. Despite the plant's nonlinearity, we formulate the control problem as an instance of quadratic programming, using a piecewise affine abstraction of the plant. The speed-versus-accuracy tradeoff for the explicit and online versions is analyzed on various reference trajectories.

This dissertation is dedicated to my parents, P.N. Murthy and Pushpa Murthy, for their love, countless sacrifices, endless support, and encouragement.

Contents

List of Figures

List of Tables

List of Abbreviations

CMACS	Computational Modeling and Analysis of Complex Systems
ODE	Ordinary Differential Equation
IMW	Iyer-Mazhari-Winslow
MM	Minimal Model
AP	Action Potential
HH	Hodgkin-Huxley
VCE	Voltage Clamp Experiment
CTMC	Continuous Time Markov Chain
PEFT	Parameter Estimation from Finite Traces
RFI	Rate Function Identification
CCM	Canonical Cell Model
BF	Bisimulation Function
IOS	Input-to-Output Stability
SOS	Sum-of-Squares
OD	(squared) Output Difference
GPU	Graphics Processing Unit
SCA	Spiral Classification Algorithm
CUDA	Compute Unified Device Architecture
PIRA	Parallel Isopotential Reconstruction Algorithm
PMS	Parallel Marching Squares
PIE	Parallel Isopotential Extraction
SOG	Selection and Output Generation
MPC	Model-Predictive Controller

FHN	FitzHugh-Nagumo
MIQP	Mixed-Integer Quadratic Programming
MPT	Multi-Parametric Toolbox
PWA	PieceWise Affine
MDR	Modified Dumas-Rondepierre

List of Symbols

$\kappa_j(t)$	Curvature of the j^{th} isopotential strip.
$\overline{g}_{Na}$	Maximum conductance of the sodium channel.
Σ_I	The 13-variable voltage-controlled CTMC for $I_N a$.
Σ_{HI}	The 2-variable HH-type abstraction for Σ_I.
Σ_{HI}^v	The constant-voltage version of Σ_{HI}, where $V = v$.
Σ_{HK}	The 2-variable HH-type abstraction for Σ_K.
Σ_{HK}^v	The constant-voltage version of Σ_{HK}, where $V = v$.
A_I	The 13×13 rate matrix of Σ_I.
I_v	Isopotential on a 2D simulation frame of a cardiac model, $I_{v,j}$ denotes the j^{th} strip.
$I_{Kv4.3}$	Fast recover calcium-independent transient outward Potassium current.
$I_{Kv4.3}$	calcium-independent transient outward Potassium current, modeled using the voltage-controlled CTMC Σ_K in the Iyer-Mazhari-Winslow (IMW) model.
I_{Na}	Sodium current, modeled using the voltage-controlled CTMC Σ_I in the IMW model.
$O_1 + O_2$	Conductance (output) of Σ_I
O_K	Conductance (output) of Σ_K.
$OVERLAP$	The SCA parameter that measures the overlap between consecutive strips of an isopotential.
SL	The strip-length parameter in the SCA.

V	Transmembrane potential of a cardiac myocyte.
V_K	Nernst potential for potassium.
V_{max}	The maximum potential reached by cardiac myocytes during an AP.
V_{Na}	Nernst potential for sodium.
V_{res}	The resting potential for cardiac myocytes.

Acknowledgments

Over the past five years, I have been privileged to work with a highly supportive and inspiring set of people. I have learned a lot from everyone of them, and would like to express my gratitude for their help and support.

I would like to begin by thanking my advisers Prof. Scott A. Smolka and Prof. Radu Grosu. Prof. Smolka's patience, rigor, and effective communication style, combined with his unsurpassed knowledge of formal verification have enabled me to mature as a researcher. He will always be an inspiration. Prof. Grosu's passion and enthusiasm for research was contagious and motivational for me, even during the tough times of my research. I could not be prouder of my academic roots, and will endeavor to pass on the research values and the dreams that they have given to me.

My advisers and I collaborated with Prof. Flavio H. Fenton on computational modeling of cardiac excitation. Along with Prof. Smolka and Prof. Grosu, Flavio patiently guided me through our interdisciplinary research. I am grateful for all his help. I would like to thank Prof. Allen R. Tannenbaum, for agreeing to serve on my dissertation committee. I have learned something new from each of our meetings, and I look forward to working with him.

I gratefully acknowledge the funding source that made my Ph.D. work possible. The research presented in this thesis was performed under the US National Science Foundation's Expedition in Computing, entitled "Computational Modeling and Analysis of Complex Systems", which is funded by the grant: NSF CCF-0926190.

During my initial years, Ezio Bartocci helped me with starting my my research. His diligence and dedication have continued to inspire me. I have always enjoyed our interactions and look forward to our sustained collaboration. I am immensely thankful to Md. Ariful Islam, my fellow-graduate student, for sticking out for me over these years. I will cherish the memories of us buckling down in the trenches for a very long time to come.

I am thankful to Prof. Elizabeth M. Cherry for her help with revising the journal version of the paper on curvature estimation. Prof. James Glimm provided invaluable insights into the spiral curvature analysis problem. We also collaborated with Dr. Richard A. Gray on this problem. I am grateful for his advise regarding the isopotential extraction subroutine. Prof. Antoine Girard helped us with the compositionality results for cardiac cell dynamics. I am very thankful for his insights on the sum-of-squares computation procedure for Bisimulation Functions. I am grateful to Prof. Scott D. Stoller for guiding me on the model-predictive control problem, and I look forward to working with him on other control-related projects. My seniors Mohammad T. Irfan and Tushar Deshpande have been very helpful. The support of my fellow-Ph.D. students Richard Defrancisco, Junxing Yang, and

Dung Phan has been instrumental during my Ph.D. They have helped me rehearse presentations, critiqued my papers, and given me vital research advice.

My graduate life was enriched by two internships: with the Los Alamos National Laboratory and Thermo Fisher Scientific Inc. Sami Ayyorgun, my manager at the Los Alamos National Laboratory, exposed me to research for the first time. The discipline and perseverance that I developed during my Los Alamos days has stayed with me. At Thermo Fisher Scientific Inc., under Vasisht R. Tadigotla's mentorship, I developed a Bayesian classifier to improve the homopolymer length estimation errors of the Ion Torrent DNA sequencing platform. His style of leadership and guidance made my internship successful and highly enjoyable.

I would also like to thank Prof. I.V. Ramakrishnan, the Graduate Program Director, and all the other professors of the department. I have learned much though our interactions. I would like to thank all the teachers and professors from my undergraduate and school days.

I am grateful to the department's academic and administrative staff: Betty Knittweis, Cynthia Scalzo and Kathy Germana. In the past five years, their cogent and efficient style of working has been critical for my success, and inspiring at the same time. Every semester, Betty filed the paperwork for my research assistantship and travel reimbursements. Cindy and Kathy ensured that my academic paperwork was always complete and updated. In addition to the secretaries, Brian Tria has been extremely helpful with the department's computing infrastructure. He has also been a very good friend over the years.

I made two close friends during my Ph.D.: Olga Kashpur and Gaurish Telang. I am thankful for all their support. In addition to working on our common courses, Gaurish also helped me move into my apartment. I met Olga when I was interning with Thermo Fisher Scientific Inc. Since then, I have enjoyed all our conversations and continue to be inspired by her dedication to research.

I would like to give special thanks to my uncle, Prof. Umesh Bellur and aunt, Sumana Srinivasan. They have been a constant source of encouragement and inspiration all my life. Lastly, I would like to thank my parents, sister Amrutha, brother-in-law Harshikesh, and my friends Vinay and Maheep for all their love and encouragement. You have always stood-by and supported me in all my pursuits. Thank you.

Abhishek Murthy
Stony Brook University
August 2014

Publications

1. Md. Ariful Islam, Abhishek Murthy, Tushar Deshpande, Scott D. Stoller, Scott A. Smolka, Ezio Bartocci, and Radu Grosu. Tracking Action Potentials of Nonlinear Excitable Cells Using Model Predictive Control. In *Proceedings of the 6th International Conference on Bioinformatics, Biocomputational Systems and Biotechnologies* (BIOTECHNO 2014), Chamonix, France, April 2014.

2. Md. Ariful Islam, Abhishek Murthy, Ezio Bartocci, Antoine Girard, Scott A. Smolka, and Radu Grosu. Compositionality Results for Cardiac Cell Dynamics. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control* (HSCC 2014), Berlin, Germany, April 2014.

3. Md. Ariful Islam, Abhishek Murthy, Ezio Bartocci, Elizabeth M. Cherry, Flavio H. Fenton, James Glimm, Scott A. Smolka, and Radu Grosu. Model-Order Reduction of Ion Channel Dynamics using Approximate Bisimulation. *Theoretical Computer Science.*

4. Abhishek Murthy, Md. Ariful Islam, Ezio Bartocci, Elizabeth M. Cherry, Flavio H. Fenton, James Glimm, Scott A. Smolka, and Radu Grosu. Approximate Bisimulations for Sodium Channel Dynamics. In *Proceedings of the 10th International Conference on Computational Methods in Systems Biology* (CMSB 2012), London, UK, October 2012.

5. Abhishek Murthy, Ezio Bartocci, Flavio H. Fenton, James Glimm, Richard A. Gray, Elizabeth M. Cherry, Scott A. Smolka, and Radu Grosu. Curvature Analysis of Cardiac Excitation Wavefronts. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (TCBB), 10(2): 323-336, 2013.

6. Abhishek Murthy, Ezio Bartocci, Flavio H. Fenton, James Glimm, Richard A. Gray, Scott A. Smolka, and Radu Grosu. Curvature Analysis of Cardiac Excitation Wavefronts. In *Proceedings of the 9th International Conference on Computational Methods in Systems Biology* (CMSB 2012), Paris, France, September 2011.

Chapter 1

Introduction

The *cardiovascular system*, which enables the circulation of blood and its constituent nutrients throughout the body, is composed of two circulatory loops: pulmonary circulation, the circuit through the lungs, where blood is oxygenated, and systemic circulation, which provides the rest of the body with oxygenated blood. The *heart* is the central organ of this system, and is responsible for pumping the blood through the pulmonary and the systemic circulation loops. Cardiovascular disorders, such as coronary heart disease and arrhythmias, affect the heart, the blood vessels and the cardiac tissue.

An estimated 81,000,000 American adults, more than one in three, have one or more types of cardiovascular disorders [51]. These disorders can be potentially fatal; about 600,000 people die of heart disease in the United States every year [60]. In addition to the grave health risks, heart diseases also pose a significant financial burden on the economy; coronary heart disease alone costs the United States $108.9 billion each year [64].

Arrhythmias, electrical disturbances of the heart, are an important subclass of cardiovascular diseases. Under nominal conditions, the cardiac tissue undergoes rhythmic electrical excitation, known as the *sinus rhythm*. This electrical excitation then induces the synchronized contraction of the cardiac tissue. Arrhythmias, anomalous patterns of electrical excitation, can originate in the *atria*, the upper chambers of the heart, or the *ventricles*, the lower chambers of the heart. *Tachycardia*, the abnormally fast electrical pacing of the heart, and *Bradycardia*, abnormally slow pacing, are common forms of arrhythmias. Tachycardia, in either the atria or ventricles, may lead to *fibrillation*, which is potentially fatal in the case of ventricles. Atrial fibrillation may lead to stroke and is an important precursor to ventricular arrhythmias.

Detailed understanding of cardiac electrical excitation is critical to controlling the pathological behaviors, such as tachycardias and arrhythmias. In this regard, *model-based approaches* have played a transformative role. Mathematical modeling of cardiac excitation is an active area of research [15, 16, 17]. Experimental data is used to calibrate models that range from the sub-cellular to the organ-level. With the ability to model and simulate complex arrhythmias on computers, many potential solutions, both drug-based and device-based, can be quickly evaluated, effectively reducing the the search space to focus on a few hypotheses that can then be experimentally validated. Iterative interaction between simulation-based approaches and experimentation has yielded invaluable insights, and promises to further refine our understanding and predictive ability of cardiac excitation.

To this end, the National Science Foundation's expedition in computing, entitled "*Computational Modeling and Analysis of Complex Systems (CMACS)*"[1] is aimed at gaining fundamental new insights into the emergent behavior of complex biological and embedded systems through the use of revolutionary, highly scalable and fully automated modeling and analysis techniques. The project entails developing the next generation of Formal Verification algorithms and tools. *Formal Verification* involves exhaustively exploring the behaviors (trajectories) of a model of the underlying system to check if it satisfies a given property, as opposed to purely simulation-based approaches, which involve a finite number of behaviors. *Model Checking* (MC) [12] and *Abstract Interpretation* (AI) [13] are two main techniques of formal verification. Four challenge problems have been identified as drivers for the development of the next generation of formal verification tools and techniques: i) Atrial Fibrillation, ii) Pancreatic Cancer, iii) Distributed Automotive Software, and iv) Avionics.

The *Atrial Fibrillation challenge is the focus of this dissertation*; the challenge entails the development and validation of personalized, adaptive strategies that counteract life-threatening electrical disturbances of the heart such as atrial fibrillation. Fig. 1.1 illustrates a typical workflow for formal verification-based computational modeling and analysis of cardiac excitation.

First, a mathematical model of cardiac excitation is identified using experimental data. These models usually take the form of a system of partial differential equations, which captures the excitation of the cardiac cells and the diffusion process of the tissue. The cell-level excitation is highly nonlinear in time, and can be captured at varying levels of detail. Physiologically detailed models reproduce the intracellular and transmembrane ion movements that contribute to cellular excitation. On the other hand, abstract models lack physiological details, and sketch a caricature of aggregate cell-level behavior. Due to their nonlinearity, mathematical models may not be directly amenable to formal verification-based analysis.

[1]See `http://cmacs.cs.cmu.edu` for more details.

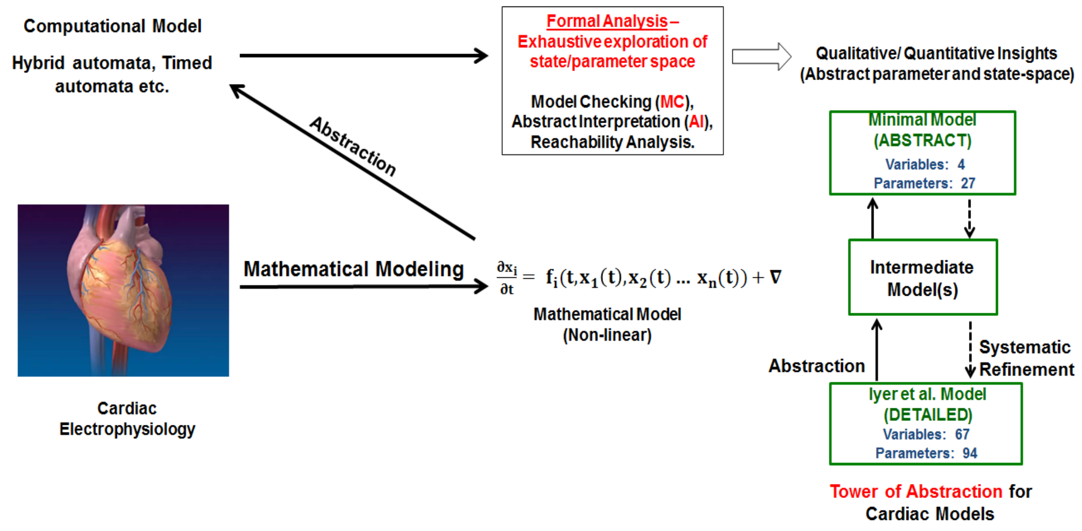

Computational Model
Hybrid automata, Timed automata etc.
Formal Analysis – Exhaustive exploration of state/parameter space
Model Checking (MC), Abstract Interpretation (AI), Reachability Analysis.
Qualitative/ Quantitative Insights (Abstract parameter and state-space)
Abstraction
Minimal Model (ABSTRACT) Variables: 4 Parameters: 27
Mathematical Modeling
$\frac{\partial x_i}{\partial t} = f_i(t, x_1(t), x_2(t) \dots x_n(t)) + \nabla$
Mathematical Model (Non-linear)
Intermediate Model(s)
Abstraction
Systematic Refinement
Iyer et al. Model (DETAILED) Variables: 67 Parameters: 94
Cardiac Electrophysiology
Tower of Abstraction for Cardiac Models

Figure 1.1: Computational modeling and analysis of cardiac excitation: a typical workflow.

The mathematical models are then transformed to *computational models*, such as hybrid automata and timed automata, which are better suited for formal verification. The conversion from mathematical models to computational ones might incur some loss in precision, but guarantees better performance in terms of verification techniques, such as MC, AI and reachability analysis.

The focus of this dissertation is to enable the Atrial Fibrillation Challenge workflow of Fig. 1.1. The results presented in the thesis fall into three broad categories: Modeling, Analysis and Control, see Fig. 1.2. We elaborate on these three aspects below.

Modeling: Computational models derived from detailed mathematical models, which intricately capture the electrophysiology of the cells, are not very well-suited for the workflow in Fig. 1.1. A detailed high-dimensional model of the underlying system often leads to the problem of *state-explosion* [32] when formal verification techniques are applied. The Iyer-Mazhari-Winslow model [36], which is introduced in Chapter 2, is an example of such a high-dimensional model; it uses 67 state-variables to capture the electrical behavior of a single cell. On the other hand, abstract models, such as the Minimal Model [8], use fewer variables to sketch a caricature of the cell-level behaviors. Such reduced models are better suited to the workflow in Fig. 1.1.

The downside of using abstract models for analysis is as follows. When an abstract model, such as the Minimal model, is used for analysis, the quantitative and qualitative insights provided by the verification techniques are in terms of the corresponding abstract state and parameter spaces. In other words, they lack physiological interpretation, which is essential for drug design and root-cause detection.

The concept of *towers of abstraction* has been proposed in [40, 22] to alleviate this problem. As shown in Fig. 1.1, a hierarchy of cardiac models is constructed. The hierarchy relates a detailed model, such as the Iyer-Mazhari-Winslow model, to abstract ones, such as the Minimal model. By formally establishing the equivalence of the Iyer-Mazhari-Winslow model and the Minimal model, the hierarchy allows the insights obtained from the later to be interpreted in terms of the physiological state-variables of the former model.

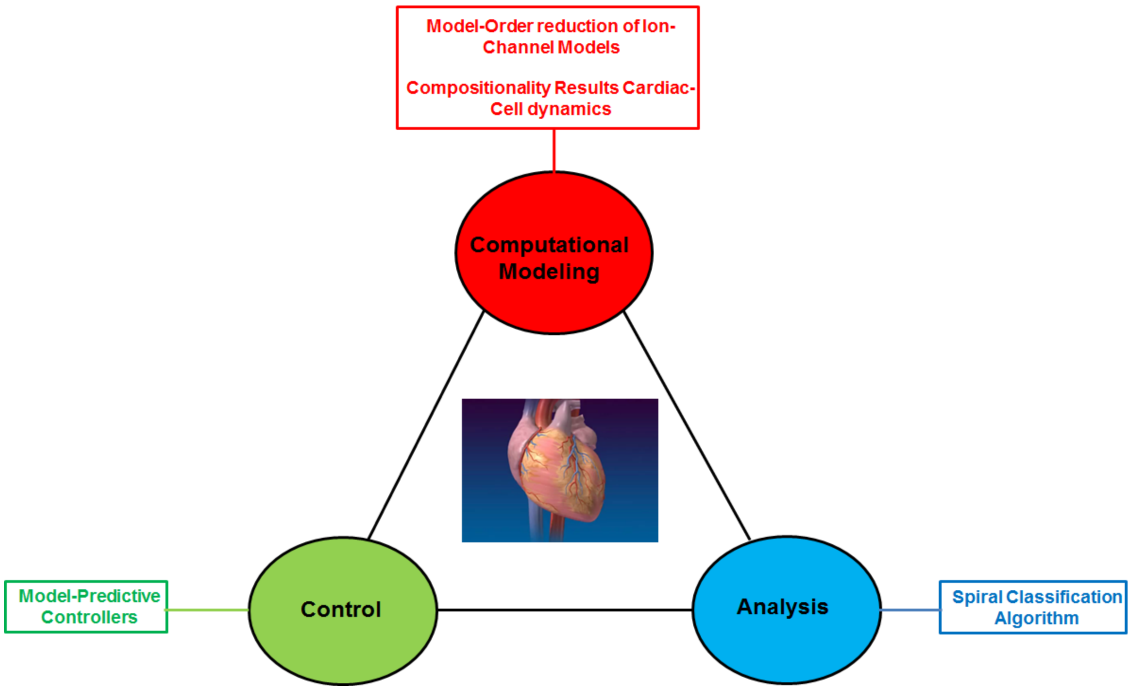

Model-Order reduction of Ion-Channel Models
Compositionality Results Cardiac-Cell dynamics
Computational Modeling
Model-Predictive Controllers
Control
Analysis
Spiral Classification Algorithm

Figure 1.2: Overview of the results in the dissertation.

In this dissertation, we propose a model-order reduction technique that can be used iteratively to construct a tower of abstraction for cardiac models, and thus enable the workflow depicted in Fig. 1.1. Components of detailed cardiac models are approximated using low-dimensional models. Specifically, we propose a curve fitting-based approximation technique to identify two-variable models corresponding to the ion-channel subsystems of the Iyer-Mazhari-Winslow model.

The two-variable models are approximately equivalent to the detailed ion-channel components. Care must be taken when the detailed component is substituted by its approximation within the whole-cell model. The feedback loop, within which the ion-channel components are composed in a cardiac-cell model, may amplify the approximation error between the detailed component and its two-variable approximation. We propose a control-theoretic proof technique to show that the substitution leads to bounded error.

Analysis: After identifying the cardiac-cell model at the requisite level of detail, its analysis becomes important. *Parameter range identification*, a pertinent analysis problem, involves estimating the parameters of a given cardiac-cell model that reproduce a particular arrhythmia. Arrhythmias are defined by tissue-level wave-like patterns of electrical excitation, and can be identified using the geometry of the waves. To this end, we propose a curvature estimation algorithm for cardiac waves: the Spiral Curvature Analysis algorithm. The algorithm can be used to automatically characterize and classify the arrhythmias, thus enabling the exploration of parameter spaces of a given cardiac model.

Control: In this part of the dissertation, we explore *model predictive control of excitable cells.* Model predictive control entails computing optimal inputs to the plant, the system being controlled, by solving an optimization problem, such that the plant follows a given reference trajectory. The optimization problem involves simulating a model of the plant over a finite time horizon. We investigated the case where the controller is equipped with a piecewise affine approximation of the nonlinear plant. Offline and online approaches to model predictive control are compared over several reference trajectories.

A Dynamical system, a recurring modeling formalism in the thesis, is defined using a 6-tuple $(\mathcal{X}, \mathcal{X}^0, \mathcal{U}, f, \mathcal{O}, g)$, where $\mathcal{X}$ is the *state space*, $\mathcal{X}^0 \subseteq \mathcal{X}$ is the set of *initial conditions*, $\mathcal{U}$ is the *input space*, $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the *vector field* defining the dynamics, $\mathcal{O}$ is the set of *outputs*, and $g : \mathcal{X} \to \mathcal{O}$ maps a state to its output.

The outline of the dissertation is as follows. In the next chapter, we introduce the Iyer-Mazhari-Winslow cardiac cell model, and identify two of its subsystems as candidates for component-wise model-order reduction. Chapters 3 and 4 are dedicated to model-order reduction. Chapter 3 presents a curve fitting-based approach to approximating the ion-channel subsystems of detailed cardiac models with two-variable abstractions. Chapter 4 presents a proof technique that allows us to show that the detailed ion-channel subsystem can be substituted, with bounded error, by its approximately equivalent two-variable abstraction within the whole-cell model. Chapter 5 presents the Spiral Curvature Analysis algorithm. Chapter 6 presents model-predictive controllers for excitable cells. Chapter 7 presents concluding remarks and directions for future research.

Chapter 2

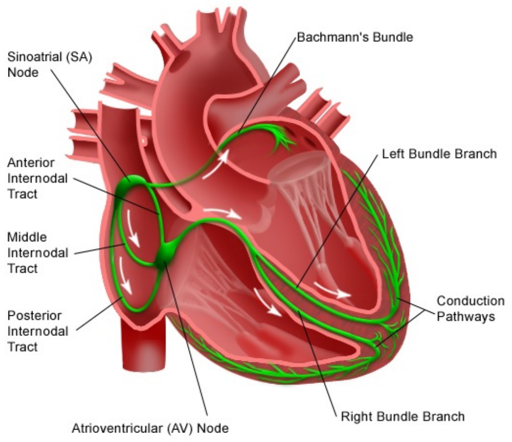
Background

This chapter begins with an overview of cardiac electrophysiology and introduces the detailed Iyer-Mazhari-Winslow ventricular-cell models. The Sodium and potassium-channel subsystems of the model are described in detail.

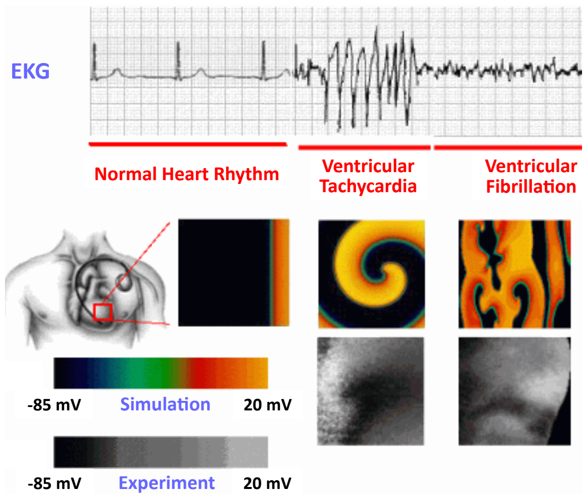
The heart is the central organ of the circulatory system and is responsible for pumping blood in the pulmonary and systemic circulation loops [19]. Pumping is achieved through electrically induced synchronized contraction and expansion of the cardiac tissue, which is composed of excitable myocytes. Thus, there are two aspects to the heart's operation: i) electrophysiology, which describes the electrical excitation of the cardiac cells and the tissue and ii) the mechanical properties which determine the rhythmic contractions. In this thesis, we will focus exclusively on cardiac electrophysiology.

Electrical excitation originates in the sino-atrial node, which consists of specialized impulse-generating cells, see Fig. 2.1 (a). The electrical impulses diffuse to the immediate neighboring myocytes and excite them. The cells then diffuse the charge to their neighbors, and this process continues resulting in sustained propagation of electrical energy across the cardiac tissue. Different regions of the heart, like the four atrial (upper) and ventricular (lower) chambers, the cardiac septum and the atrio-ventricular junction have different types of cardiac cells that respond to electrical stimulus in slightly different ways. Their electrical properties regulate the conduction of electrical impulses throughout the organ.

Electrical coupling between adjacent cardiac cell results in the diffusion of electrical impulses across the entire tissue. These propagating impulses manifest as characteristics patterns known as *electrical waves*, see Fig. 2.1 (b). Planar waves correspond nominal patterns of propagation, whereas anomalous patterns, such as a single spiral wave or multiple spiral wavelets, correspond to disorders, such as reentrant tachycardia and fibrillation respectively.



(a) Cardiac Electrophysiology.



(b) Measuring and modeling electrophysiological behaviors of the heart.

Figure 2.1: Cardiac electrophysiology: an emergent cyber-physical system with hybrid micro-level behavior.

Cardiac Electrophysiological modeling attempts to capture the cellular electrical excitation and coupled diffusion that jointly lead to the emergent tissue-level patterns that characterize nominal and anomalous cardiac behaviors. We begin this chapter with an overview of electrophysiological models for cardiac cells. The sodium- and potassium-channel subsystems of a detailed cardiac model are reviewed.

Cardiac myocytes belong to the class of *excitable cells*, which also includes neurons and skeletal muscle cells. Such cells have the following excitability property: supra-threshold electrical excitation, either external or from neighboring cells, results in a characteristic change in the transmembrane voltage known as the *Action Potential (AP)*. Either an external stimulus, or the diffusing charge from the neighboring cells can excite the myocyte, causing an AP to quickly depolarize the membrane from a negative resting potential of V_{res} mV to a maximum of V_{max} mV followed by gradual repolarization. See Fig. 2.2 for a typical AP.

The transmembrane potential V of the cell is determined by the differences in the concentrations of sodium (Na), potassium (K), and calcium (Ca) ions present in the extracellular and intracellular mediums. The resulting gradients of concentrations lead to the flow of ions across the membrane that can be measured as an *ionic current* for each type of flow. The ion flow is facilitated by *ion channels* present in the membrane of the cells. Ion channels are proteins that exhibit conformational changes on varying the transmembrane voltage, and are selectively permeable to either Na, K, or Ca ions. Some of the conformations allow ion flow, whereas others inhibit the respective current. The rate of change of voltage resulting from the ion flows is a function of the various ionic currents at any point in time. The Iyer-Mazhari-Winslow model captures the various ion-channel dynamics and the corresponding ionic currents of a ventricular myocyte. We describe this model next.

2.1 Iyer-Mazhari-Winslow Model for Ventricular Myocytes

The IMW model [36] is a detailed mechanistic model for ventricular myocytes. The membrane potential V is modeled as

$$\begin{aligned}-C\frac{dV}{dt} = {} & I_{Na}+I_{Nab}+I_{Ca}+I_{Cab}+I_{Kr}+I_{Ks}+I_{K1}+I_{to1}+I_{p(Ca)}+ \\ & I_{NaCa}+I_{NaK}+I_{CaK}+I_{st},\end{aligned} \tag{2.1}$$

where C is the membrane conductance. The terms on the right represent 12 ionic currents and the external stimulus I_{st}. Each ionic current results from the ion flow through the corresponding ionic channel. Each type of ion-channel channel contributes a subsystem (submodel) that is responsible for the corresponding ionic current. Each such submodel is a mean-field approximation of the collective stochastic behavior of all the ion channels of the corresponding type. In this report, we focus on the sodium-channel subsystem, which contributes I_{Na}, and the potassium channel subsystem, which contributes the $I_{Kv4.3}$ component of the calcium-independent transient outward Potassium current I_{to1}. The sodium current I_{Na} is primarily responsible for the upstroke phase, whereas I_{to1} influences the AP's notch in the early repolarization phase. See Fig. 2.2 for a schematic of the IMW model.

2.1.1 Sodium-Channel Subsystem of the IMW Model

The sodium current I_{Na} is modeled as:

$$I_{Na}(V,t) = \overline{g}_{Na}(O_1 + O_2)(V - V_{Na}), \tag{2.2}$$

where $\overline{g}_{Na}$ is the maximum conductance of the sodium channel, and V_{Na} is sodium's Nernst potential. The term $(O_1 + O_2)$ represents the conductance (output) of the sodium-channel subsystem, which is defined as follows.

Definition 2.1.1. The *sodium channel subsystem* Σ_I is given by $(X_I, X_I^0, \mathcal{V}, f_I, \mathcal{O}_I, g_I)$. A state $\mathbf{x}_I \in X_I \subseteq \mathbb{R}^{13}_{\geq 0}$ is the occupancy probability distribution over the 13 states of the voltage-controlled Continuous Time Markov Chain (CTMC) shown in Fig. 2.3 in

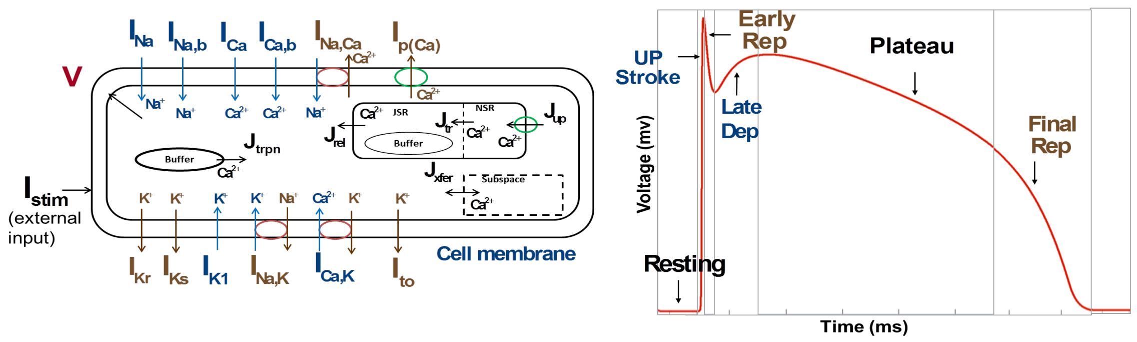


Figure 2.2: (Left) Currents in IMW: Blue and brown arrows show ionic currents flowing through channels. Blue circles and arrows correspond to ionic exchanger currents and green circles denote ionic pumps. Intra-cellular currents are shown in Magenta. (Right) A typical AP and its phases.

the following order of the state labels: $[C_0, C_1, C_2, C_3, C_4, O_1, O_2, CI_0, CI_1, CI_2, CI_3, CI_4, I]$. The dynamics f_I is given by

$$f_I : \dot{\mathbf{x}}_I = A_I(V)\ \mathbf{x}_I, \tag{2.3}$$

where $V \in \mathcal{V} \subseteq \mathbb{R}$, the transmembrane voltage, is the input to the system and $A_I(V)$ is the 13×13 voltage-controlled rate matrix. The off-diagonal entry $A_I(i,j), i \neq j$, is the transition rate from state $\mathbf{x}_{Ij}$ to state $\mathbf{x}_{Ii}$. For example, $A_I(5,6) = \delta(V)$, the transition rate from O_1 to C_4. The diagonal entry $A_I(i,i)$ is the sum of all the outgoing rates from state $\mathbf{x}_{Ii}$. The transition rates are exponential functions of V, and can be found in Table 2.1.

The set of outputs $\mathcal{O}_I \subseteq \mathbb{R}_{\geq 0}$ contains the conductance values for the states. Given a state $\mathbf{x}_I$, $g_I(\mathbf{x}_I) \triangleq \mathbf{x}_{I6} + \mathbf{x}_{I7}$ maps it to its conductance given by the sum of the occupancy probabilities of the states labeled O_1 and O_2. We use O_I to denote the output when the state can be inferred from the context. The system has a single initial condition $\mathbf{x}_{I0} \in X^0$, which is defined in Table 4 of [36].

2.1.2 Potassium Channel Subsystem of the IMW Model

The potassium current $I_{Kv4.3}$ is modeled as

$$I_{Kv4.3} = \overline{g}_{Kv4.3}\,(O_K(V))\,(V - V_K), \tag{2.4}$$

where $\overline{g}_K$ is the maximum conductance of the channel, V_K is the Nernst potential for potassium, $O_K(V) = O(V)$ is the conductance (output) of the potassium-channel subsystem, which is defined as follows.

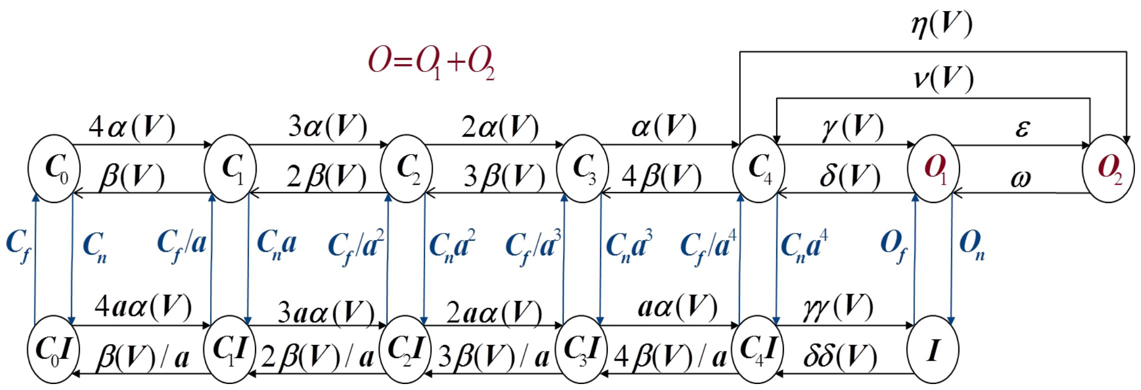


Figure 2.3: The sodium-channel subsystem Σ_I of the IMW model.

rate	function	rate	function	rate	function
$\alpha(V)$	$c.e^{-19.6759+0.0113V}$	$\delta\delta(V)$	$c.e^{-38.4839-0.1440V}$	ϵ	0.0227
$\beta(V)$	$c.e^{-26.2321-0.0901V}$	$\gamma\gamma(V)$	$c.e^{-21.9493+0.0301V}$	ω	1.0890
$\gamma(V)$	$c.e^{-16.5359+0.1097V}$	$\eta(V)$	$c.e^{-19.6729+0.0843V}$	c_n	0.7470
$\delta(V)$	$c.e^{-27.0926-0.0615V}$	$O_n(V)$	$c.e^{-20.6726+0.0114V}$	c_f	0.2261
$\nu(V)$	$c.e^{-26.3585-0.0678V}$	$O_f(V)$	$c.e^{-39.7449+0.0027V}$	a	1.4004

Table 2.1: Transfer rates of Σ_I, which is shown in Fig. 2.3. Values were instantiated from Table 6 of [36] at temperature T = 310K, and $c = 8.513 \times 10^9$.

Definition 2.1.2. The *potassium channel subsystem* Σ_K is given by $(X_K, X_K^0, \mathcal{V}, f_K, \mathcal{O}_K, g_K)$. A state $\mathbf{x}_K \in X_K \subseteq \mathbb{R}^{10}_{\geq 0}$ is the occupancy probability distribution over the 10 states of the voltage-controlled CTMC shown in Fig. 2.4 in the following order of the state labels: $[C_0, C_1, C_2, C_3, O, CI_0, CI_1, CI_2, CI_3, OI]$. The dynamics f_k is given by

$$f_K : \dot{\mathbf{x}}_K = A_K(V)\ \mathbf{x}_K, \tag{2.5}$$

where $V \in \mathcal{V} \subseteq \mathbb{R}$, the transmembrane voltage, is the input to the system and $A_K(V)$ is the 10×10 voltage-controlled rate matrix. The off-diagonal entry $A_K(i,j), i \neq j$, is the transition rate from state $\mathbf{x}_{Kj}$ to state $\mathbf{x}_{Ki}$. For example, $A_K(4,5) = 4\beta_a(V)$, the transition rate from O to C_3. The diagonal entry $A_K(i,i)$ is the sum of all the outgoing rates from state $\mathbf{x}_i$. The transition rates are exponential functions of V, and can be found in Table 2.2.

The set of outputs $\mathcal{O}_K \subseteq \mathbb{R}_{\geq 0}$ contains the conductance values for the states. Given a state $\mathbf{x}_K$, $g_K(\mathbf{x}_K) \triangleq \mathbf{x}_5$ maps it to its conductance given by the occupancy probability of the state labeled O. We use O_K to denote the output when the state can be inferred from the context. The system has a single initial condition

$\mathbf{x}_{K0} \in X^0$, which is defined in Table 4 of [36].

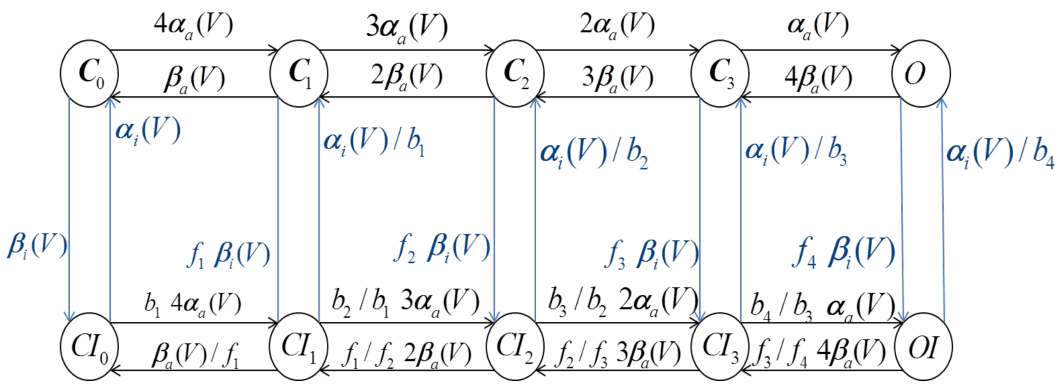


Figure 2.4: The potassium-channel subsystem Σ_K of the IMW model.

rate	function	rate	function	rate	function
$\alpha_a(V)$	$0.5437e^{0.029V}$	f_1	1.8936	b_1	6.7735
$\beta_a(V)$	$0.0802e^{-0.0468V}$	f_2	14.2246	b_2	15.6213
$\alpha_i(V)$	$0.0498e^{-0.0004V}$	f_3	158.5744	b_3	28.7533
$\beta_i(V)$	$0.0008e^{(5.374\times10^{-8}V)}$	f_4	142.9366	b_4	524.5762

Table 2.2: Transfer rates of Σ_K, shown in Fig. 2.4. $c = 8.513 \times 10^9$. Note that these values are different from the ones given in Table 9 of [36]. Corrections were made to the parameters to match the observed APs of the IMW model.

Chapter 3

Model-order Reduction of Ion-Channel Models

In this chapter, we present an abstraction technique based on [61, 34] for the ion-channel subsystems of the IMW model. We begin by motivating model-order reduction for ion-channel components, and then present the Hodgkin-Huxley formalism for ion-channels. Then, we present a curve fitting-based approach to identifying two-variable Hodgkin-Huxley-type abstractions for the sodium and potassium-channel subsystems introduced in Sections 2.1.1 and 2.1.2. We conclude with empirical results showing that the sodium and potassium channel subsystems can be substituted with their respective two-variable abstractions with bounded error.

3.1 Motivation

Improved data acquisition has led to the creation of increasingly sophisticated cardiac models. Their main purpose is to elucidate the biological laws governing the electric behavior of cardiac myocytes, i.e., their underlying ionic processes [20].

Inspired by the squid-neuron model [31], which will be introduced in the next section, Luo and Rudy devised one of the first myocyte models, for guinea pig ventricular cells [52]. Adapting this model to human myocytes led to the ten Tusscher-Noble^2-Panfilov model [72], which has 17 state variables and 44 parameters. Based on updated experimental data, IMW subsequently developed their model comprising of 67 state variables and 94 parameters [36].

From 17 to 67 variables, all such models capture myocytic behavior at a particular level of abstraction, and hence all of them play an important role in the modeling hierarchy. It is essential, however, to maintain focus on the purpose of a particular model; that is, of the particular cellular and ionic processes whose behavior the models is intended to capture. Disregarding this purpose may lead to the use of unnecessarily complex model, which may render not only analysis, but also simulation, intractable.

If the only entity of interest is the transmembrane voltage, Cherry and Fenton have experimentally shown that the Minimal model [8] consisting of only 4 variables and 27 parameters can accurately capture voltage propagation properties in 1D, 2D, and 3D networks of myocytes. The Minimal model has enabled dramatic simulation speedups [3], and its linear hybridization has even been used for formal symbolic analysis [29].

Since new technological advances are expected to lead to further insights into myocytic behavior, it is likely that the IMW model will be further refined by adding new variables. As in model checking and controller synthesis, one would therefore like to compute the smallest approximation that is observationally equivalent to the state-of-the-art cardiac model with respect to the property of interest, modulo some bounded approximation error. This, however, is not easily accomplished, as it implies the automatic approximation of very large system of nonlinear Ordinary Differential Equations (ODEs).

A first step toward the desired automation is to identify a set of approximation techniques that allow one to systematically remove unobservable variables from say, a detailed model such as IMW to end up with the Minimal model, assuming that the only observable variable is the voltage. A byproduct of this work is to establish a long-missing formal relation among the existing myocyte models, thus facilitating the transfer of properties established at one layer of abstraction to the other layers. Building such *towers of abstraction* is becoming increasingly prevalent in systems biology [40, 22].

In this chapter we focus on model-order reduction and abstraction of ion channel dynamics. The main question posed in this chapter is the following: *Assuming that the conductance of the ion channel is the only observable, is the behavior of a Hodgkin-Huxley-type channel equivalent to the behavior of the IMW channel, modulo a well-defined approximation error?* Specifically, we answer this question for the sodium and the calcium-independent potassium channels. See Fig. 3.1 for an overview of the chapter.

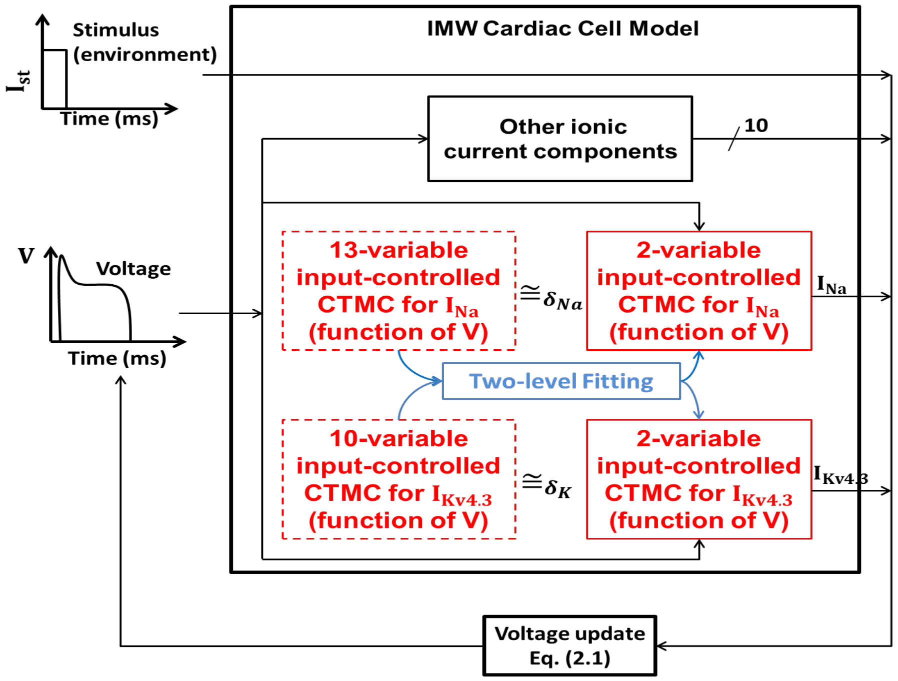

Stimulus
(environment)
I_{st}
Time (ms)
IMW Cardiac Cell Model
Other ionic current components
10
V
Voltage
Time (ms)
13-variable input-controlled CTMC for I_{Na} (function of V)
$\cong_{\delta_{Na}}$
2-variable input-controlled CTMC for I_{Na} (function of V)
I_{Na}
Two-level Fitting
10-variable input-controlled CTMC for $I_{Kv4.3}$ (function of V)
$\cong_{\delta_K}$
2-variable input-controlled CTMC for $I_{Kv4.3}$ (function of V)
$I_{Kv4.3}$
Voltage update Eq. (2.1)

Figure 3.1: A modular view of the IMW model, which was introduced in Section 2.1. It composes various concurrently evolving components corresponding to the different ionic currents. We replace the 13- and the 10-variable subsystems for I_{Na} and $I_{Kv4.3}$ respectively with corresponding 2-variable Hodgkin-Huxley-type abstractions. A two-level curve fitting process, described in Section 3.3, is used to identify the abstractions. Approximate equivalence of the detailed components and their corresponding abstractions allows us to substitute them for each other within the whole-cell model. The stimulus current affects the overall voltage update and is not an input to the ionic current components. The system outputs the 13 currents in Eq. (2.1).

3.2 Hodgkin-Huxley Formalism for Modeling Ion-Channels

Hodgkin and Huxley, in their seminal work of [31], modeled the squid neuron's sodium channel behavior using two independent processes: activation and inactivation. Starting from the resting potential, if the cell is depolarized to a constant voltage, *activation* causes a sudden increase in the channel's conductance. This is followed by *inactivation*, which gradually brings the conductance down, before reaching a steady state. The activation and inactivation processes are captured by the Hodgkin-Huxley (HH) model as follows.

Definition 3.2.1. The *HH model* Σ_H is given by $(Y, Y^0, \mathcal{V}, f_H, \mathcal{O}, g_H)$. Its state $\mathbf{y} \in Y \subseteq \mathbb{R}^2_{\geq 0}$ measures the degrees of activation, denoted by m, and inactivation of the channel, denoted by h. Component y_1 corresponds to m and y_2 corresponds to

h. The dynamics f_H is given by

$$f_H : \dot{\mathbf{y}} = A_H(V)\ \mathbf{y} + B_H(V), \tag{3.1}$$

where $A_H = \begin{bmatrix} -(\alpha_m(V) + \beta_m(V)) & 0 \\ 0 & -(\alpha_h(V) + \beta_h(V)) \end{bmatrix}$, $B_H = \begin{bmatrix} \alpha_m(V) \\ \alpha_h(V) \end{bmatrix}$, and $V \in \mathcal{V} \subseteq \mathbb{R}$, the transmembrane voltage, is the input. The rates $\alpha_i(V)$ and $\beta_i(V)$, $i \in \{m, h\}$, are nonlinear functions of V that will be identified during the two-step curve-fitting process.

The set of outputs $\mathcal{O} \subseteq \mathbb{R}_{\geq 0}$ contains the conductance values for the states. Given a state $\mathbf{y}$, $g_H(\mathbf{y}) \triangleq \mathbf{y}_1^\gamma \mathbf{y}_2$, maps it to its conductance, which corresponds to $m^\gamma h$. The parameter γ is called the *degree of activation*. We use O_H to denote the output when the state can be inferred from the context.

The system has a single initial condition that will be identified in the curve fitting process.

3.3 Model-Order Reduction of Ion Channel Dynamics

We construct two HH-type models, Σ_{HI} and Σ_{HK}, that can be substituted for Σ_I and Σ_K respectively within the IMW cardiac-cell model. We perform the following abstractions in the process:

- The abstractions Σ_{HI} and Σ_{HK} employ three and four activating subunits respectively. In other words, the degree of activation $\gamma = 3$ for Σ_{HI} and $\gamma = 4$ for Σ_{HK}. A single subunit is used to model inactivation.

- We abstract away the conditional dependence between activation and inactivation. This is done by abstracting away the scaling factors: a of Σ_I and $f_1 - f_4$, $b_1 - b_4$ of Σ_K, see Figs. 2.3 and 2.4.

After identifying Σ_{HI}, its conductance, m^3h, is substituted for Σ_I's conductance, $O_1 + O_2$, in the IMW model's Eq. (2.2). Similarly Σ_{HK}'s conductance, m^4h, replaces Σ_K's conductance, O_K in Eq. (2.4). Note that this leads to two levels of substitution. First, the conductance of the detailed ionic current components is substituted by the abstract model conductances. Then the modified current component replaces the original term in Eq. (2.1).

Our approach to identifying Σ_{HI} and Σ_{HK} from the detailed models, Σ_I and Σ_K respectively, is summarized in Fig. 3.2, and described as follows.

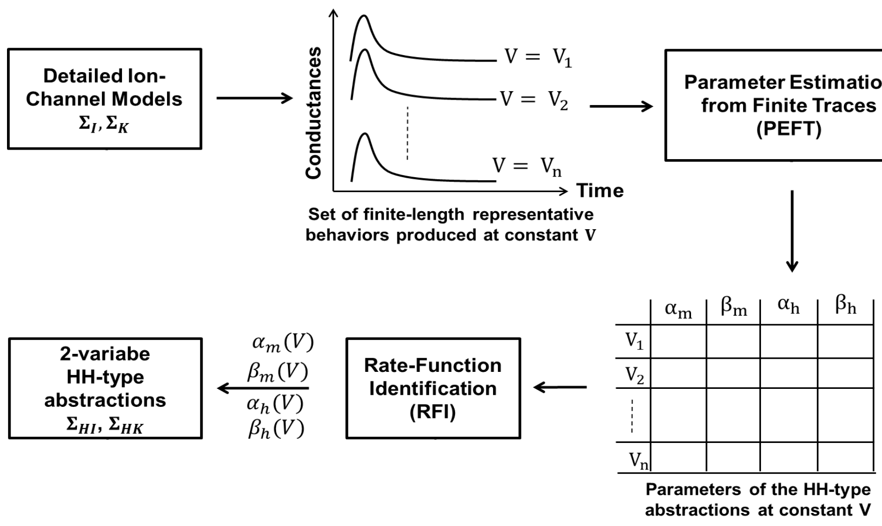

Detailed Ion-Channel Models Σ_I, Σ_K
Conductances
V = V1
V = V2
V = Vn
Time
Set of finite-length representative behaviors produced at constant V
Parameter Estimation from Finite Traces (PEFT)
αm βm αh βh
V1
V2
Vn
Parameters of the HH-type abstractions at constant V
Rate-Function Identification (RFI)
$\alpha_m(V)$ $\beta_m(V)$ $\alpha_h(V)$ $\beta_h(V)$
2-variabe HH-type abstractions Σ_{HI}, Σ_{HK}

Figure 3.2: Abstraction process for the ion-channel models. The voltage-controlled CTMC components are simulated at constant voltages (clamp potentials) using the steady state values corresponding to $V = V_{res}$ as the initial conditions. The conductance time courses are then fit as per Eq. (3.2) to obtain the parameters $\alpha_m, \beta_m, \alpha_h, \beta_h$ at the clamp potentials used for voltage clamp simulations. The four parameter values, along with the initial conditions determine the abstractions at constant voltage. The parameters are then fit in the RFI step to obtain parameter functions $\alpha_m(V), \beta_m(V), \alpha_h(V), \beta_h(V)$.

1. **Voltage clamp simulations**
   Voltage Clamp Experiments (VCEs), pioneered by Hodgkin and Huxley in their seminal work of [31], are intended to expose the activation and inactivation processes governing a channel's behavior. The experiments involve stimulating the channel by changing the membrane potential suddenly and then holding it constant, starting from appropriate initial conditions. As the ion channel reacts by opening, and then closing, the resulting ionic currents are recorded. The corresponding conductance time courses characterize the channel's response to varying the membrane potential.

   We simulated VCEs by simulating the detailed models, Σ_I and Σ_K, for various values of V. In other words, the systems of linear differential equations governing the evolution of Σ_I^v and Σ_K^v, were simulated for different values of $V_{res} \leq v \leq V_{max}$. We used 20,000 uniformly spaced voltage values for v. The models $\mathcal{M}_{Na}^v$ and $\mathcal{M}_K^v$ were simulated using MATLAB's ODE45 solver [57], starting from the initial conditions specified in Table 4 of [36]. The initial conditions correspond to the steady state of the models at $v = V_{res}$. This is exactly the initially conditions used by Hodgkin and Huxley in [31]. The resulting conductance time courses, $O_1(t)+O_2(t)$ for Σ_I^v and $O(t)$ for Σ_K^v, were recorded until steady state was reached. As per Theorem A.0.1 in Appendix A, for all $v \in [V_{res}, V_{max}]$, Σ_I^v and Σ_K^v have stable equilibria and therefore steady state is guaranteed. Simulating the models at constant voltage values corresponds to the clamp potentials to which the membrane was excited in [31] to uncover the activation and inactivation process.

2. **Parameter Estimation from Finite Traces (PEFT)**
   Parameter Estimation from Finite Traces (PEFT) is a procedure that identifies Σ_{HI}^v and Σ_{HK}^v models corresponding to Σ_I^v and Σ_K^v respectively. The parameters $\alpha_m^v, \beta_m^v, \alpha_h^v, \beta_h^v$ for both the models are estimated such that the resulting conductance time courses, produced by Σ_{HI}^v and Σ_{HK}^v, match the conductance time courses observed in the voltage clamp simulations for Σ_I^v and Σ_K^v respectively.

   In our implementation, we fit Σ_I^v's conductance time series, $O_1(t) + O_2(t)$, to Σ_{HI}^v's $m^3(t)h(t)$. Time series $O(t)$ observed from Σ_K^v was fit with $m^4(t)h(t)$ to identify Σ_{HK}^v. At constant voltage v, the trajectories $m(t)$ and $h(t)$ of Σ_{HI}^v and Σ_{HK}^v are given by

   $$z(t) = \frac{\alpha_z^v}{\alpha_z^v + \beta_z^v} + \left(z(0) - \frac{\alpha_z^v}{\alpha_z^v + \beta_z^v} \right) \exp\left(-\left(\alpha_z^v + \beta_z^v\right)t\right) \tag{3.2}$$

   where $z \in \{m, h\}$ and $x \in \{I, K\}$. The fitting was performed using MATLAB's curve fitting utility *cftool* [58] for each voltage value v used in the voltage clamp simulations. Two aspects of our implementation deserve further elaboration:

   - **Choosing $m(0)$ and $h(0)$** - The initial conditions were chosen such that for Σ_{HI}^v, $m(0)^3h(0)$ and for Σ_{HK}^v, $m(0)^4h(0)$ was approximately equal to the steady state conductances of $\Sigma_{HI}^{V_{res}}$ and $\Sigma_{HK}^{V_{res}}$ respectively. As per convention, we also ensured that $m^v(0) \approx 0$ and $h(0) \approx 1$ for both the models. We chose $m(0) = 0.0027, h(0) = 0.95$ for Σ_{HI}^v; $m(0) = 0.007, h(0) = 0.98$ for Σ_{HK}^v.
   - **Providing seed-values** - For each voltage-value v, *cftool* needs seed values of α_x^v, β_x^v, $x \in \{m, h\}$, to start optimizing over the parameter space. The parameters estimated for the i^{th} voltage v_i were used as seed-values for v_{i+1}. For $i = 0$, when $v = v_{res}$, the parameters were calculated by trial and error.

3. **Rate Function Identification (RFI)**
   Rate Function Identification (RFI) is a procedure that fits the parameter values, α_m^v, β_m^v, α_h^v and β_h^v, as functions of voltage to produce the parameter functions $\alpha_m(V)$, $\beta_m(V)$, $\alpha_h(V)$ and $\beta_h(V)$. We fit polynomial functions using *cftool* [58]. The parameter functions defining Σ_{HI} and Σ_{HK} are as follows.

Rate functions that define Σ_{HI}:

$$\alpha_m(V) = \begin{cases} 13.63 - \frac{14.3}{1+\exp(0.061V+1.72)} & V \le 19.98 \\ 20.76 - \frac{7.89}{1+\exp(464.1V-13920)} & V > 19.98 \end{cases} \tag{3.3}$$

$$\beta_m(V) = \begin{cases} 9.925 & V \le -65 \\ 4.7 - \frac{2.58}{1+\exp(0.61V+38.19)} & -65 < V \le 5.8 \\ 12.24 - \frac{7.77}{1+\exp(1877V-56314)} & V > 5.8 \end{cases} \tag{3.4}$$

$$\alpha_h(V) = \frac{0.1745}{1+\exp(269.8V+17720)} \tag{3.5}$$

$$\beta_h(V) = 10.1 - \frac{10}{1+\exp(0.0579V+0.71)} \tag{3.6}$$

Rate functions that define Σ_{HK}

$$\alpha_m(V) = \begin{cases} 0.45\exp(0.026V) & V \le 24.5 \\ 0.85 - \frac{0.048}{1+\exp(-0.2V+9.5)} & V > 24.5 \end{cases} \tag{3.7}$$

$$\beta_m(V) = \begin{cases} 0.029\exp(0.065V) & V \le 24.5 \\ 0.1839 - \frac{0.05}{1+\exp(0.19V-9.32)} & V > 24.5 \end{cases} \tag{3.8}$$

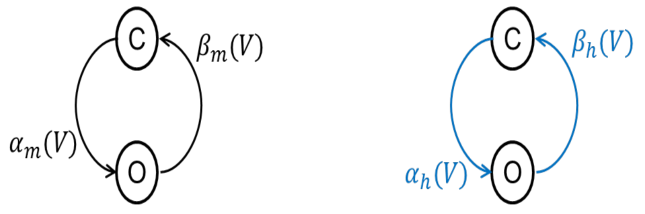
$$\alpha_h(V) = 0.0015 - \frac{0.0014}{1+\exp(0.027V-2.54)} \tag{3.9}$$

$$\beta_h(V) = \begin{cases} 0.12 - \frac{0.06}{1+\exp(-0.054V+2.62)} & V \le 24.5 \\ 0.109 + \frac{0.015}{1+\exp(-0.033V+10.83)} & V > 24.5 \end{cases} \tag{3.10}$$

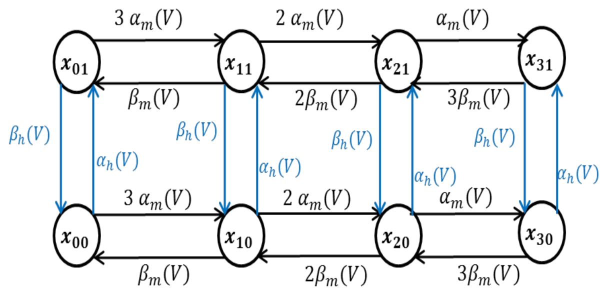
Adapting the Abstraction Process to Arbitrary Observable Functions of Σ_I and Σ_K:
The two-step abstraction process, consisting of PEFT and RFI, assumed that the conductance of the detailed model was the observable (ouput) function for them. In other words, $O_1(V,t) + O_2(V,t)$ for Σ_I, and $O(V,t)$ for Σ_K mapped a state of the corresponding model to its output. The abstraction methodology described above is not restricted by these observable functions and can be adapted to arbitrary functions that map a state to its output.

Suppose we are given a stochastic (detailed) ion-channel model Σ, with a function that maps the state occupancy probability vector to a real-valued output. The goal is to reduce it to an HH-type abstraction, Σ_H, that has a degree of activation λ and 1 as the degree of inactivation. We provide details for modifying PEFT, such that the resulting set of constant-voltage Σ_H^v systems are behaviorally equivalent to constant-voltage versions, Σ^v, of the detailed model.

The first step is to establish a mapping between the states of Σ and a $2(\lambda+1)$-state stochastic model corresponding to the HH-type model, denoted by Σ_H^{stoch}. We will label the states of Σ_H^{stoch} as x_{ij}, and denote the corresponding occupancy probability by p_{ij} where $i = 0, \ldots, \lambda$ and $j = 0, 1$. The model Σ_H^{stoch} interprets the degrees of activation and inactivation as the number of independent activating and inactivating subunits of the channel. In our case, the state x_{ij} corresponds to the conformation of the channel where i activating and j inactivating subunits are in an "open" state that allow ion flow. Fig. 3.3 shows the model Σ_H^{stoch} corresponding to an HH-type model with $\lambda = 3$ and a degree of inactivation of 1. In the model, the



(a) Activating (m-type) subunit **(b) Inactivating (h-type) subunit**



(c) Stochastic model for a channel 3 activating (λ = 3) and 1 inactivating subunits.

Figure 3.3: Invariant manifolds can be used to map the states of a $2(\lambda+1)$-state stochastic model to an HH-type model with a degree of activation λ and degree of inactivation 1 ($\lambda = 3$ in the example).

state x_{21} corresponds to the a conformation where the inactivating subunit and two of the activating subunits are open. The inactivating subunit can close at a rate of $\beta_h(V)$ and change the conformation to x_{20}. The remaining activating subunit can open at the rate of $\alpha_m(V)$ to change the state to x_{31}. In this conformation, all the three activating subunits and the inactivating subunit are open. Thus, this state corresponds to the conformation of the channel which allows ion flow. From the state x_{21}, any of the two independent activating subunits could close at a rate of $2\beta_m(V)$ to change the state to x_{11}. Once we have mapped the states of the given stochastic model to the states of Σ_H^{stoch}, PEFT can be modified to identify $\Sigma_H^{stoch,v}$ systems that can match the behavioral traces observed from Σ^v.

The two-state HH-type model forms an invariant manifold [47] of Σ_H^{stoch}. The occupancy probability of the state x_{ij} is given by $m^i h^j$. This correspondence helps us map a state vector of Σ_H^{stoch} to a state of the HH-type model (the vector $[m, h]^T$). Note that this mapping is exact and provides an output function that can be matched to the output function of Σ_I. Suppose the output function maps the states of Σ that correspond to the states x_{11} and x_{31} of Σ_H^{stoch}. Then the output function of Σ_H will take as arguments mh and m^3h. PEFT can be used to minimize divergence of this new observation function to identify $\Sigma_H^{stoch,v}$ systems.

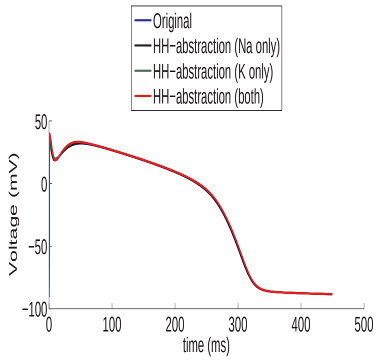
3.4 Empirical Evidence of Substitutivity

The subsystems Σ_I and Σ_K were substituted by their respective HH-type abstractions Σ_{HI} and Σ_{HK} within the IMW model. The substitutions were done in three combinations: 1) substitution of Σ_I only, 2) substitution of Σ_K only, and 3) substitution of both Σ_I and Σ_K. The modified IMW models were simulated in MATLAB in all three cases with an integration time step of 0.001 ms. Both supra- and sub-threshold stimuli, lasting 0.5 ms, were used to excite the cardiac cell. Suprathreshold stimuli used were: $S1 = -100$ pA/pF, and $S2 = -120$ pA/pF. Subthreshold stimuli employed were: $S3 = -10$ pA/pF, and $S4 = -20$ pA/pF.

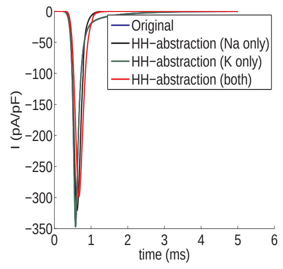
S	V (mV)			I_{Na} (pA/pF)			I_K (pA/pF)		
	Na Only	K Only	Both	Na Only	K Only	Both	Na Only	K only	Both
S1	7.73×10^{-4}	1.4×10^{-3}	1.6×10^{-3}	2.7×10^{-3}	8.3×10^{-6}	5.2×10^{-3}	6.2×10^{-5}	1.72×10^{-4}	1.1×10^{-3}
S2	7.3×10^{-4}	1.4×10^{-3}	1.5×10^{-3}	5.2×10^{-3}	5.4×10^{-5}	5.2×10^{-2}	4.67×10^{-5}	2.54×10^{-4}	2.5×10^{-4}
S3	1.61×10^{-5}	1.1×10^{-3}	1.1×10^{-3}	-	-	-	-	-	-
S4	1.39×10^{-4}	1.2×10^{-3}	1.3×10^{-3}	-	-	-	-	-	-

Table 3.1: Mean L2 errors incurred in the simulations after substituting Σ_I by Σ_{HI} and Σ_K by Σ_{HK} in the IMW model. The first column S stands for the stimulus used to excite the cell at the beginning of the simulation. Only the voltage errors were recorded for the sub-threshold stimuli S3 and S4 as the currents were negligible.

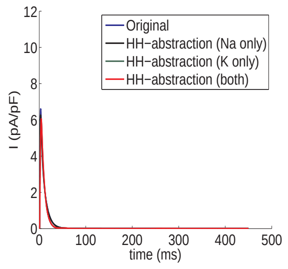
Fig. 3.4 provides empirical evidence of the modified whole cell models being behaviorally equivalent to the original models. The model retains both normal and



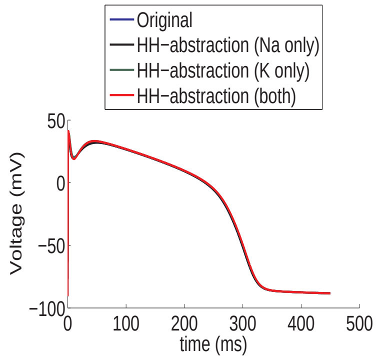
(a) Comparison of APs.



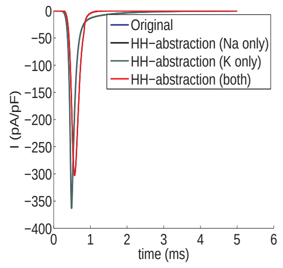
(b) Comparison of I_{Na}.



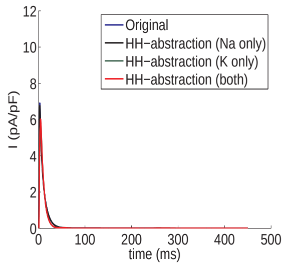
(c) Comparison of $I_{Kv4.3}$.



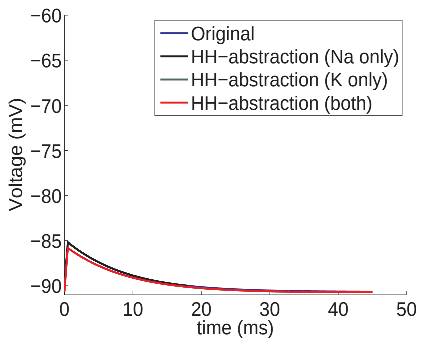
(d) Comparison of APs.



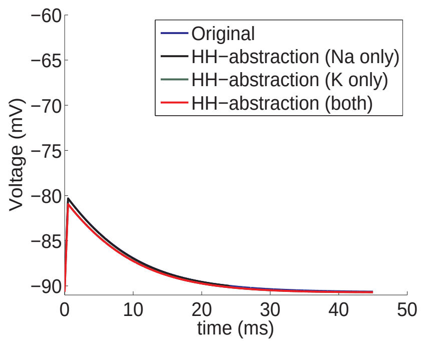
(e) Comparison of I_{Na}.



(f) Comparison of $I_{Kv4.3}$.



(g) Comparison of APs ($S3$).



(h) Comparison of APs ($S4$).

Figure 3.4: Comparison of the original and the modified IMW models when Σ_I and Σ_K are substituted by Σ_{HI} and Σ_{HK} respectively. Subfigures (a)-(c) are obtained for the S1 stimulus, (d)-(f) for the S2 stimulus, (g) for the S3 stimulus and (h) for the S4 stimulus. S1 and S2 are supra-threshold stimuli and lead to an AP, whereas S3 and S4 being subthreshold stimuli fail to produce the AP.

anomalous cell-level behaviors on replacing the 13-state sodium channel and/or 10-state potassium channel components with the corresponding 2-state abstraction(s).

In the following chapter, we will introduce a control theoretic proof technique that can be used to show the approximate equivalence of i) Σ_I and Σ_{HI}, ii) Σ_K and Σ_{HK}, and iii) the original IMW model and the modified version obtained by substituting Σ_I and Σ_K with Σ_{HI} and Σ_{HK} respectively.

Chapter 4

Compositionality Results for Cardiac Cell Dynamics

In the previous chapter, we presented the two-step procedure, based on PEFT and RFI, to identify the HH-type abstractions Σ_{HI} and Σ_{HK} corresponding to the detailed ion-channel models Σ_I and Σ_K respectively. We substituted the detailed models with their respective abstractions within the IMW model, and presented empirical results that show that the error remains bounded. In this chapter, we present a control theoretic proof technique that enables us to formally derive proofs of substitutivity. The proofs entail computing Lyapunov-like functions that characterize input-to-state stability and robustness of dynamical systems, using sum-of-squares optimization.

We begin with a brief introduction in the next section. Then, we present canonical cell models, and pose the pose the problem of substitutivity fir them. The proof technique is presented in the following section. We then present two algorithms for implementing the technique using sum-of-squares optimization. The two algorithms are then applied to show that Σ_I and Σ_K can be substituted by Σ_{HI} and Σ_{HK} respectively within canonical cell models. We conclude the chapter by presenting results and comparing the two algorithms.

4.1 Motivation

Effective modeling for insightful analysis of a given biological system has been the cornerstone of Systems Biology. Modeling involves coming up with a mathematical, and often a computational, representation of the *dynamics* of the *states* of the biolgical system. The key to effective modeling lies in striking the balance between

i) *precision*: the ability of the model to replicate the underlying biological intricacy, and ii) *performance*: being amenable to automated computational analysis techniques, such as simulations and formal verification. The techniques of abstraction and compositionality have been instrumental for striking this balance [40, 22]. *Abstraction* is the process of removing unnecessary detail from a given model such that the resulting abstract version is an equivalent conservative approximation, but is better suited for the analysis of the given properties-of-interest. Given a system consisting of several interacting components, also known as a subsystems, *Compositionality* enables us to substitute a component by its abstraction. Compositional reasoning is used to show that the overall system retains the properties-of-interest when the component is substituted by its equivalent abstraction.

Mechanistic cardiac models, such as the IMW model, capture the electrophysiology of the myocytes in great detail, but do scale with automated analysis techniques, such as Model Checking [12] and Reachability Analysis [14]. Even simulations become intractable. On the other hand, the Minimal model portrays only a caricature of the action potential, but is amenable to very fast simulations [3] and even formal analysis [29].

In the case of real-valued continuous-time dynamical systems, which are often used to model biological systems, a notion of approximate model-order reduction has been proposed, where the abstraction is approximately equivalent to the concrete model [26, 27, 28, 43]. PEFT and RFI, presented in Sections 3.3, constitute a curve fitting-based approach to component-wise model-order reduction of cardiac models. Detailed electrophysiological models, such as the IMW model can be systematically reduced to abstract models, such as the Minimal model, by performing PEFT and RFI-based reduction on each of the components of the IMW model.

Fig. 4.1 illustrates component-wise model-order reduction of the IMW model. The sodium-channel component Σ_I is reduced to Σ_{HI} using PEFT and RFI. Then, Σ_I is substituted by Σ_{HI} within the IMW model. In Section 3.4, we presented empirical evidence that the substitution does not lead to divergent systems, i.e. the error remains bounded.

The *feedback composition* $R \,||\, C$ of two dynamical systems R and C is obtained by feeding the output of R as the input to C and vice versa. Care must be taken, however, when a concrete model of a system component C is replaced by an approximately equivalent abstraction C' within a feedback loop. In a feedback loop, the approximation error between C and C' may get get amplified. The Σ_I component is present in a feedback loop within the IMW model, as shown in Fig. 4.1. Thus, it is important to ensure that the feedback loop does not amplify the error between Σ_I and Σ_{HI}. Similarly, Σ_K is in a feedback loop within the IMW model, and we must ensure that the error between Σ_K and Σ_{HK} does not get amplified.

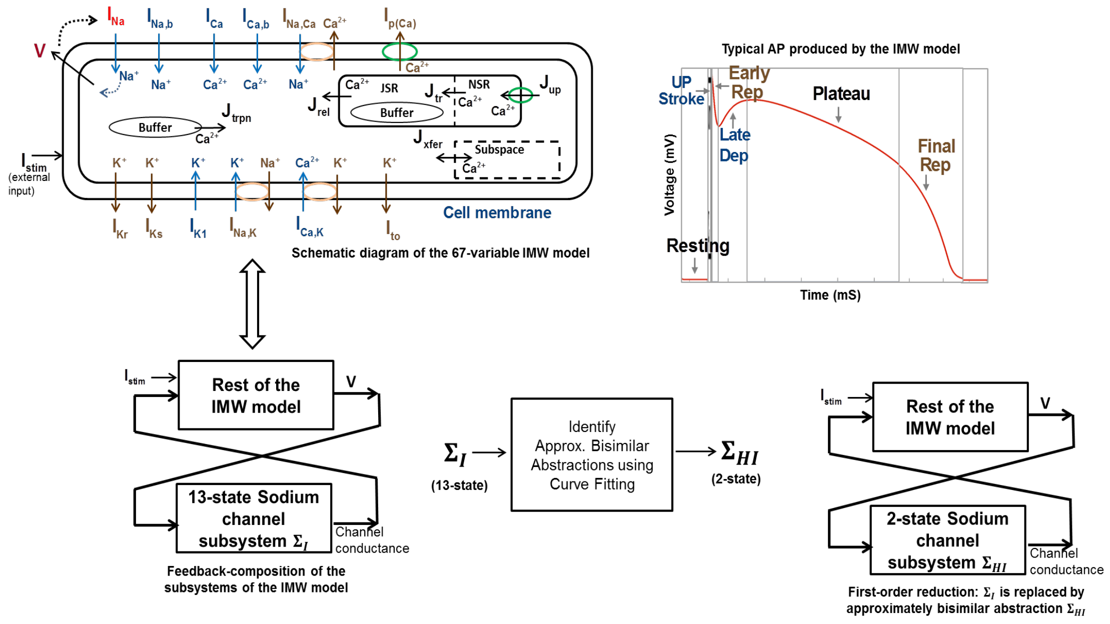

Schematic diagram of the 67-variable IMW model
Cell membrane
Typical AP produced by the IMW model
UP Stroke
Early Rep
Plateau
Late Dep
Final Rep
Resting
Voltage (mV)
Time (mS)
Rest of the IMW model
13-state Sodium channel subsystem Σ_I
Channel conductance
Feedback-composition of the subsystems of the IMW model
Σ_I (13-state)
Identify Approx. Bisimilar Abstractions using Curve Fitting
Σ_{HI} (2-state)
2-state Sodium channel subsystem Σ_{HI}
First-order reduction: Σ_I is replaced by approximately bisimilar abstraction Σ_{HI}

Figure 4.1: Component-wise model-order reduction of IMW model using abstraction and compositional reasoning. PEFT and RFI, see Section 3.3, present a curve fitting-based technique to identify HH-type abstractions for the components of the IMW model. The sodium-channel component Σ_I is replaced by Σ_{HI} within the feedback loop of the IMW model. Similarly, Σ_K is replaced by Σ_{HK}.

In this chapter, we present a proof technique that enables us to show that the feedback loop does not amplify the error when components are substituted by their approximate model-order reduced versions within feedback loops.

4.2 Canonical Cell Models

We begin this section by introducing the voltage subsystem Σ_C, which represents the cell membrane. Σ_C is feedback composed with the detailed ion-channel components Σ_I and Σ_H, and their respective HH-type abstractions Σ_{HI} and Σ_{HK} to obtain four *Canonical Cell Models (CCMs)*. We then state our compositionality results in terms of the CCMs.

Definition 4.2.1. The *voltage subsystem* Σ_C is a capacitor-like model given by $(\mathcal{V}, \mathcal{V}^0, \mathcal{O}, f_C, \mathcal{V}, g_C)$. State $V \in \mathcal{V} \subseteq \mathbb{R}$ is the voltage. Σ_C evolves as:

$$f_C : \dot{V} = -G(V - V_{eq})\ O, \tag{4.1}$$

where G and V_{eq} are the parameters of the model, and $O \in \mathcal{O} \subseteq \mathbb{R}_{\geq 0}$, the conductance of the ion channel, is Σ_C's input. The system outputs its state, i.e., for $V \in \mathcal{V}$, $g_C(V) = V$, and the initial condition is denoted by V_0.

As per Eq. (4.1), V_{eq} represents the equilibrium for a fixed-conduc-tance input. In the case of detailed cardiac cell models, such as the IMW model, ion-channel subsystems such as Σ_I and Σ_K take voltage as input from the rest of the model and provide the conductance of the channel as the output. The rest of the model takes the channel conductance as input and outputs the voltage, which is then fed back to the ion-channel subsystems.

Next, we define CCMs Σ_{CI}, Σ_{CK}, Σ_{CHI}, and Σ_{CHK} that reflect this feedback-based composition. The models are canonical in the sense that other ion-channel subsystems can be similarly added to obtain a complete cardiac-cell model.

Definition 4.2.2. Systems Σ_{CI}, Σ_{CK}, Σ_{CHI}, and Σ_{CHK} (see Fig. 4.2) are obtained by performing feedback-composition on the voltage subsystem Σ_C with Σ_I, Σ_K, Σ_{HI}, and Σ_{HK} respectively; i.e., $\Sigma_{CI} = \Sigma_C || \Sigma_I$, $\Sigma_{CH} = \Sigma_C || \Sigma_H$, $\Sigma_{CHI} = \Sigma_C || \Sigma_{HI}$, and $\Sigma_{CHK} = \Sigma_C || \Sigma_{HK}$. The state spaces, initial conditions, dynamics and outputs are inherited from the subsystems, as explained below. The composed systems are autonomous systems and do not receive any external inputs.

A state of Σ_{CI} is given by $[\mathbf{x}_I, V_I]^T$, where $\mathbf{x}_I$ is a state of Σ_I and V_I is a state of Σ_C. The subscript I in V_I is used to denote the copy of Σ_C composed with Σ_I. The system dynamics are given by Eqs. (2.3) and (4.1). The parameters for Σ_C's dynamics, which is given by Eq. (4.1), are $G = 5$, $V_{eq} = 30\ mV$, $V_0 = -30\ mV$. The output is given by $[g_I(\mathbf{x}_I), V_I]^T$. The initial condition is the pair of the initial conditions of Σ_I and Σ_C.

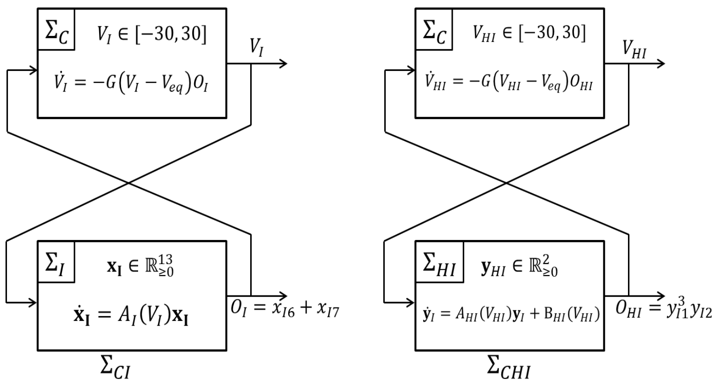
A state of Σ_{CK} is given by $[\mathbf{x}_K, V_K]^T$, where $\mathbf{x}_K$ is a state of Σ_K and V_K is a state of Σ_C. The subscript K in V_K is used to denote the copy of Σ_C composed with Σ_K. The system dynamics are given by Eqs. (2.5) and (4.1). The parameters for Σ_C's dynamics, which is given by Eq. (4.1), are $G = 4.5$, $V_{eq} = -30\ mV$, $V_0 = 30\ mV$. The output is given by $[g_K(\mathbf{x}_K), V_K]^T$. The initial condition is the pair of the initial conditions of Σ_K and Σ_C.

A state of Σ_{CHI} is given by $[\mathbf{y}_I, V_{HI}]^T$, where $\mathbf{y}_I$ denotes a state of Σ_{HI} and V_{HI} denotes a state of Σ_C. The subscript HI in V_{HI} is used to denote the copy of Σ_C composed with Σ_{HI}. The system dynamics are given by Eq. (3.1), where the rate functions are given by Eqs. (3.3) - (3.6), and Eq. (4.1). The parameters for Σ_C's dynamics are duplicated from Σ_{CI}: $G = 5$, $V_{eq} = 30\ mV$, $V_0 = -30\ mV$. The output is given by $[g_H(\mathbf{y}_I), V_{HI}]^T$. Note that $g_H(\mathbf{y}_I) = y_1^3 y_2$. The initial condition is the pair of the initial conditions of Σ_{HI} and Σ_C.

A state of Σ_{CHK} is given by $[\mathbf{y}_K, V_{HK}]^T$, where $\mathbf{y}_K$ denotes a state of Σ_{HK} and

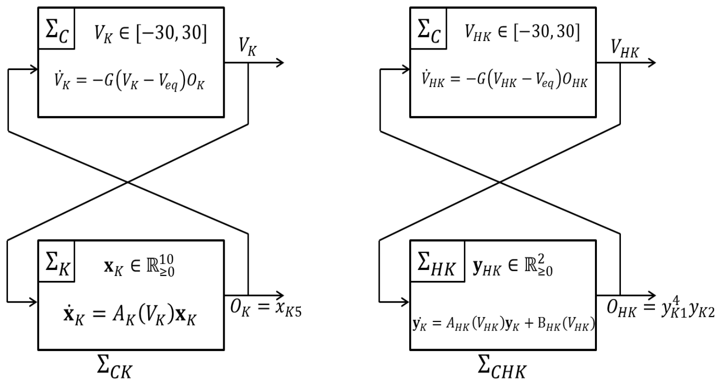
V_{HK} denotes a state of Σ_C. The subscript HK in V_{HK} is used to denote the copy of Σ_C composed with Σ_{HK}. The system dynamics are given by Eq. (3.1), where the rate functions are given by Eqs. (3.7) - (3.10), and Eq. (4.1). The parameters for Σ_C's dynamics are duplicated from Σ_{CK}: $G = 4.5$, $V_{eq} = -30\ mV$, $V_0 = 30\ mV$.

The output is given by $[g_H(\mathbf{y}_K), V_{HI}]^T$. Note that $g_H(\mathbf{y}_I) = y_1^4 y_2$. The initial condition is the pair of the initial conditions of Σ_{HK} and Σ_C.



Σ_C
$V_I \in [-30,30]$
$\dot{V}_I = -G(V_I - V_{eq})O_I$
V_I
Σ_I
$\mathbf{x_I} \in \mathbb{R}^{13}_{\geq 0}$
$\dot{\mathbf{x}}_\mathbf{I} = A_I(V_I)\mathbf{x_I}$
$O_I = x_{I6} + x_{I7}$
Σ_{CI}
Σ_C
$V_{HI} \in [-30,30]$
$\dot{V}_{HI} = -G(V_{HI} - V_{eq})O_{HI}$
V_{HI}
Σ_{HI}
$\mathbf{y}_{HI} \in \mathbb{R}^2_{\geq 0}$
$\dot{\mathbf{y}}_I = A_{HI}(V_{HI})\mathbf{y}_I + B_{HI}(V_{HI})$
$O_{HI} = y_{I1}^3 y_{I2}$
Σ_{CHI}

(a) Σ_{CI} and Σ_{CHI}.



Σ_C
$V_K \in [-30,30]$
$\dot{V}_K = -G(V_K - V_{eq})O_K$
V_K
Σ_K
$\mathbf{x}_K \in \mathbb{R}^{10}_{\geq 0}$
$\dot{\mathbf{x}}_K = A_K(V_K)\mathbf{x}_K$
$O_K = x_{K5}$
Σ_{CK}
Σ_C
$V_{HK} \in [-30,30]$
$\dot{V}_{HK} = -G(V_{HK} - V_{eq})O_{HK}$
V_{HK}
Σ_{HK}
$\mathbf{y}_{HK} \in \mathbb{R}^2_{\geq 0}$
$\dot{\mathbf{y}}_K = A_{HK}(V_{HK})\mathbf{y}_K + B_{HK}(V_{HK})$
$O_{HK} = y_{K1}^4 y_{K2}$
Σ_{CHK}

(b) Σ_{CK} and Σ_{CHK}.

Figure 4.2: CCMs Σ_{CI}, Σ_{CHI}, Σ_{CK}, and Σ_{CHK}: ion-channel subsystems Σ_I, Σ_{HI}, Σ_K, Σ_{HK} are feedback-composed with Σ_C, which represents the cell membrane. Σ_{CHI} is obtained by i) identifying the 2-variable abstraction Σ_{HI} of Σ_I using the curve-fitting procedure given in Section 3.3, and ii) substituting Σ_{HI} for the detailed model Σ_I in the composition Σ_{CI}. Similarly, Σ_{CHK} is obtained by i) identifying the 2-variable abstraction Σ_{HK} of Σ_K using the curve-fitting procedure given in Section 3.3, and ii) substituting Σ_{HK} for the detailed model Σ_K in the composition Σ_{CK}.

The four CCMs are illustrated in Fig. 4.2 below. The two pairs of CCMs, i)

Σ_{CI} and Σ_{CHI} and ii) Σ_{CK} and Σ_{CHK} are obtained by replacing the detailed ion channel models, Σ_I and Σ_K, by the abstract models, Σ_{HI} and Σ_{HK}, respectively within feedback loops. In the following sections, we present a proof technique based on the theory of Bisimulation functions to prove the equivalence of the pairs of the CCMs.

4.3 Bisimulation Functions

The concept of Input-to-Output Stability (IOS) is key to proving our compositionality results. IOS is formalized using contractive metrics, called Bisimulation Functions [25], that characterize the joint IOS of two dynamical systems. The following definition is adapted from [25] and uses $\parallel . \parallel$ to denote the squared L2 norm.

Definition 4.3.1. Let $\Sigma_i = (\mathcal{X}_i, \mathcal{X}_i^0, \mathcal{U}, f_i, \mathcal{Y}, g_i)$, $i = 1, 2$, be two dynamical systems such that $\mathcal{X}_i \subseteq \mathbb{R}^{n_i}, \mathcal{U} \subseteq \mathbb{R}^m$ and $\mathcal{Y} \subseteq \mathbb{R}^p$. A *Bisimulation Function (BF)* is a smooth function $S : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}_{\geq 0}$ such that for every $\mathbf{x}_1 \in \mathcal{X}_1$, $\mathbf{x}_2 \in \mathcal{X}_2$, $\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{U}$:

$$\parallel g_1(\mathbf{x}_1) - g_2(\mathbf{x}_2) \parallel \leq S(\mathbf{x}_1, \mathbf{x}_2), \tag{4.2}$$

$$\begin{aligned} \forall \mathbf{x}_1, \mathbf{x}_2, \mathbf{u}_1, \mathbf{u}_2, \exists \lambda > 0, \gamma \geq 0 : \frac{\partial S}{\partial \mathbf{x}_1} f_1(\mathbf{x}_1, \mathbf{u}_1) + \frac{\partial S}{\partial \mathbf{x}_2} f_2(\mathbf{x}_2, \mathbf{u}_2) \\ \leq -\lambda S(\mathbf{x}_1, \mathbf{x}_2) + \gamma \parallel \mathbf{u}_1 - \mathbf{u}_2 \parallel \end{aligned} \tag{4.3}$$

Next, we present a modified version of Theorem 1 of [25], which captures the joint IOS of two systems.

Theorem 4.3.2. Let S be a BF with parameters λ and γ between dynamical systems Σ_i, $i = 1, 2$, and let $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ be two trajectories of the systems. For all $t \geq 0$,

$$\begin{aligned} \parallel g_1(\mathbf{x}_1(t)) - g_2(\mathbf{x}_2(t)) \parallel &\leq S(\mathbf{x}_1(t), \mathbf{x}_2(t)) \\ &\leq e^{-\lambda t} S(\mathbf{x}_1(0), \mathbf{x}_2(0)) + \\ &\frac{\gamma}{\lambda} \parallel \mathbf{u}_1 - \mathbf{u}_2 \parallel_\infty \end{aligned}$$

where $\parallel \mathbf{u}_1 - \mathbf{u}_2 \parallel_\infty = sup_{t \geq 0} \parallel \mathbf{u}_1(t) - \mathbf{u}_2(t) \parallel$ denotes the maximum difference in the input signals being fed to the two systems.

Proof. See Appendix B for a proof. □

When subsystems are connected using feedback, their respective BFs can be composed subject to a small-gain condition. We formalize this idea by stating a result based on Theorem 2 of [25].

Theorem 4.3.3. *Let $\Sigma_i = (\mathcal{X}_i, \mathcal{X}_i^0, \mathcal{U}_i, f_i, \mathcal{O}_i, g_i)$, $i = 1, 2, A, B$, be dynamical systems such that $\mathcal{U}_1 = \mathcal{O}_A$, $\mathcal{U}_A = \mathcal{O}_1$, $\mathcal{U}_2 = \mathcal{O}_B$ and $\mathcal{U}_B = \mathcal{O}_2$. Let S_{12}, parameterized by λ_{12} and γ_{12}, be a BF between Σ_1 and Σ_2. Let S_{AB}, parameterized by λ_{AB} and γ_{AB}, be a BF between Σ_A and Σ_B.*

Let $\Sigma_{A1} = \Sigma_A || \Sigma_1$ and $\Sigma_{B2} = \Sigma_B || \Sigma_2$. If the small-gain condition *(SGC) $\frac{\gamma_{AB}\gamma_{12}}{\lambda_{AB}\lambda_{12}} < 1$ is met, then a BF S can be constructed between Σ_{A1} and Σ_{B2} by composing S_{AB} and S_{12} as follows:*

$$S(\mathbf{x}_{A1}, \mathbf{x}_{B2}) = \alpha_1 S_{AB}(\mathbf{x}_A, \mathbf{x}_B) + \alpha_2 S_{12}(\mathbf{x}_1, \mathbf{x}_2) \tag{4.4}$$

where $\mathbf{x}_{A1} = [\mathbf{x}_A, \mathbf{x}_1]^T$ and $\mathbf{x}_{B2} = [\mathbf{x}_B, \mathbf{x}_2]^T$ and the constants α_1 and α_2 are given by:

$$\begin{cases} \frac{\gamma_{12}}{\lambda_{AB}} < \alpha_1 < \frac{\lambda_{12}}{\gamma_{AB}} \quad \textit{and} \quad \alpha_2 = 1 \quad \textit{if} \quad \lambda_{AB} \leq \gamma_{12} \\ \alpha_1 = 1 \quad \textit{and} \quad \frac{\gamma_{AB}}{\lambda_{12}} < \alpha_2 < \frac{\lambda_{AB}}{\gamma_{12}} \quad \textit{if} \quad \lambda_{12} \leq \gamma_{AB} \\ \alpha_1 = 1 \quad \textit{and} \quad \alpha_2 = 1 \quad \textit{in other cases} \end{cases} \tag{4.5}$$

Proof. See Appendix B for a proof. □

4.4 Compositional Approach

BFs characterize IOS between dynamical systems and thus establish equivalence between them. We use BFs to establish the following compositionality results.

Compositionality Result 1: There exists a BF S_1 between Σ_{CI} and Σ_{CHI} that renders the two CCMs to be approximately equivalent in the sense characterized by Theorem 4.3.2.

Compositionality Result 2: There exists a BF S_2 between Σ_{CK} and Σ_{CHK} that renders the two CCMs to be approximately equivalent in the sense characterized by Theorem 4.3.2.

S_1 is computed compositionally as follows. First, the components Σ_I and Σ_{HI} are proved to be approximately equivalent by computing a BF S_{IH} between the two systems. Then, the context Σ_C is proved to be robust to input deviations by computing a BF S_C for it. The computation procedure ensures that the prerequisite SGC condition is satisfied by S_{IH} and S_C, thereby enabling the application of Theorem 4.3.3; this results in a BF S_1 between Σ_{CI} and Σ_{CHI}.

S_2 is computed compositionally as follows. First, the components Σ_K and Σ_{HK} are proved to be approximately equivalent by computing a BF S_{KH} between the two systems. S_C, as explained above, characterizes the robustness of Σ_C to input deviations. The computation procedure again ensures that the prerequisite SGC condition is satisfied by S_{KH} and S_C, thereby enabling the application of Theorem 4.3.3; this results in a BF S_2 between Σ_{CK} and Σ_{CHK}.

We describe the computation procedure for S_{IH}, S_C, and S_1 in the next section. An alternative procedure, see Section 4.7, is used to compute S_{KH}, S_C and S_2.

4.5 BF Computation Using Sum-of-Squares Optimization and Input-Space Sampling

In this section, we present the Sum-of-Squares (SOS) optimization-based algorithm from [35] for computing BFs, and comment on its input-space sampling approach.

A multivariate polynomial $p(x_1, x_2, \ldots, x_n) = p(\mathbf{x})$ is an *SOS polynomial* if there exist polynomials $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})$ such that $p(\mathbf{x}) = \sum_{i=1}^{m} f_i^2(\mathbf{x})$. For example, $p(x, y) = x^2 - 6xy + 12y^2$ is an SoS polynomial; it can be expressed as $(x - 3y)^2 + (\sqrt{3}y)^2$. We denote the set of all SoS polynomials by $\mathbb{S}$.

An *SoS optimization Problem* (SoSP), involves finding an $S \in \mathbb{S}$ such that a linear objective function, whose decision variables are the coefficients of S, is optimized. The constraints of the problem are linear in the decision variables. A formal definition of an SoSP can be found in the MATLAB SOSTOOLS user guide ([65], p. 7).

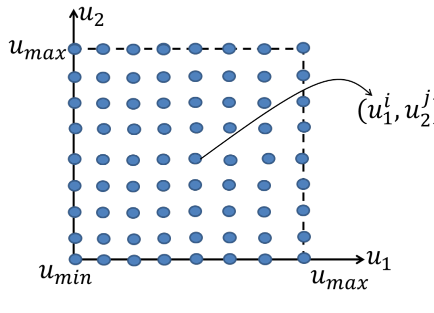


Figure 4.3: $\mathcal{U}$: a grid formed by uniformly sampling the input space. A solution of Eqs. (4.6)- (4.8) satisfies Eq. (4.3) only on the blue points denoting the samples on the grid.

Consider two dynamical systems $(X_i, \{\mathbf{x}_i^0\}, [u_{min}, u_{max}], f_i, O, g_i),\ i = 1, 2$. Firstly, note that the two dynamical systems have singleton initial conditions $\{\mathbf{x}_i^0\}, i =$

1, 2. Also, both the systems accept scalar inputs u_1 and u_2 respectively. Let $\mathcal{U}$ represent a discretized grid for u_1 and u_2 as shown in Fig. 4.3. The input space $[u_{min}, u_{max}]$ is divided into a finite number of uniformly spaced intervals, and (u_1^i, u_2^j) denotes the pair of inputs where u_1 takes the i^{th} value and u_2 takes the j^{th} value.

In [35], we presented the following SoSP for computing BFs using SoS optimization.

SOSP 1:

$$\text{Minimize } S(\mathbf{x}_1^0, \mathbf{x}_2^0) \tag{4.6}$$

subject to:

$$-S(\mathbf{x}_1, \mathbf{x}_2) + [g_1(\mathbf{x}_1) - g_2(\mathbf{x}_2)]^2 \in \mathbb{S}, \tag{4.7}$$

$$\exists \lambda > 0, \gamma \geq 0 \text{ such that } \forall u_1^i, u_2^j \in \mathcal{U} : \tag{4.8}$$

$$-\frac{\partial S}{\partial \mathbf{x}_1} f_1(\mathbf{x}_1, u_1^i) - \frac{\partial S}{\partial \mathbf{x}_2} f_2(\mathbf{x}_2, u_2^j) - \lambda S(\mathbf{x}_1, \mathbf{x}_2) + \gamma (u_1^i - u_2^j)^2 \in \mathbb{S}.$$

The BF S starts at its maximum value at the pair of initial conditions $(\mathbf{x}_1^0, \mathbf{x}_2^0)$, and then decays along various trajectories of the two systems. Thus, along any pair of trajectories, the gap between $S(\mathbf{x}_1(t), \mathbf{x}_2(t))$ and the output difference $||g_1(\mathbf{x}_1(t)) - g_2(\mathbf{x}_2(t))||$ is maximum at $t = 0$, i.e. at the initial states. To improve the bound on the output difference given by S, we minimized $S(\mathbf{x}_1(0), \mathbf{x}_2(0))$ as the objective function of the SoSP.

Note that Eq. (4.8) represents a family of constraints. In the implementation, the equation contributes one constraint for each pair $(u_1^i, u_2^j) \in \mathcal{U}$.

4.5.1 Computing S_{IH}, S_C, and S_1 in SOSTOOLS

Next, we show how to compute the BFs S_{IH} and S_C by implementing SOSP 1 in MATLAB SOSTOOLS. S_{IH} is a BF between Σ_I and Σ_{HI}, with parameters λ_{IH} and γ_{IH}, while S_C is a BF between Σ_C and itself, with parameters λ_C and γ_C. Values for these parameters must be determined such that the SGC condition of Theorem 4.3.3 is satisfied.

I. Rewriting the system definitions

The rate matrix $A_I(V_I)$ of Σ_I has a zero eigenvalue for all $V_I \in [-30, 30]$, the values the input can take when Σ_I is composed with Σ_C. The system dynamics can be rewritten to make the new rate matrix Hurwitz, which is a necessary and sufficient condition for exponential stability. As the system is a voltage-controlled CTMC, one of the variables is always redundant; i.e., the occupancy probability of one state can be expressed as 1 - (sum of all other occupancy probabilities).

Using this fact, we removed the redundant state occupancy probability corresponding to state I of Σ_I, see Fig. 2.3. The resulting dynamics are affine: $\dot{\mathbf{x}}'_I = A'_I(V_I).\mathbf{x}'_I + B_I(V_I)$, where the state vector $\mathbf{x}'_I$ now contains the first 12 probabilities from $\mathbf{x}_I$. The 12×12 matrix A'_I and the 12×1 affine term B_I can be found in [33]. For a fixed voltage input $V_I = v$, the equilibrium of the resulting linear system was shifted to the origin by solving $A'_I(v).\mathbf{x}'_I = -B_I$.

Σ_{HI} has an output function of degree 4, which would require S_{IH} to be an 8^{th}-order polynomial. Higher-order polynomials lead to possibly intractable instances of SoS optimization in SOSTOOLS. To resolve this issue, the 2-state system was converted to an equivalent 8-state stochastic model that has a linear output function. In [47], Σ_{HI} is shown to be the exact invariant manifold of the 8-state voltage-controlled CTMC shown in Fig. 4.4. S_{IH} is then defined as a quadratic function over 19 (12 + 7) variables, which ensures that the corresponding SOS problem can be solved despite the increase in the number of undetermined coefficients.

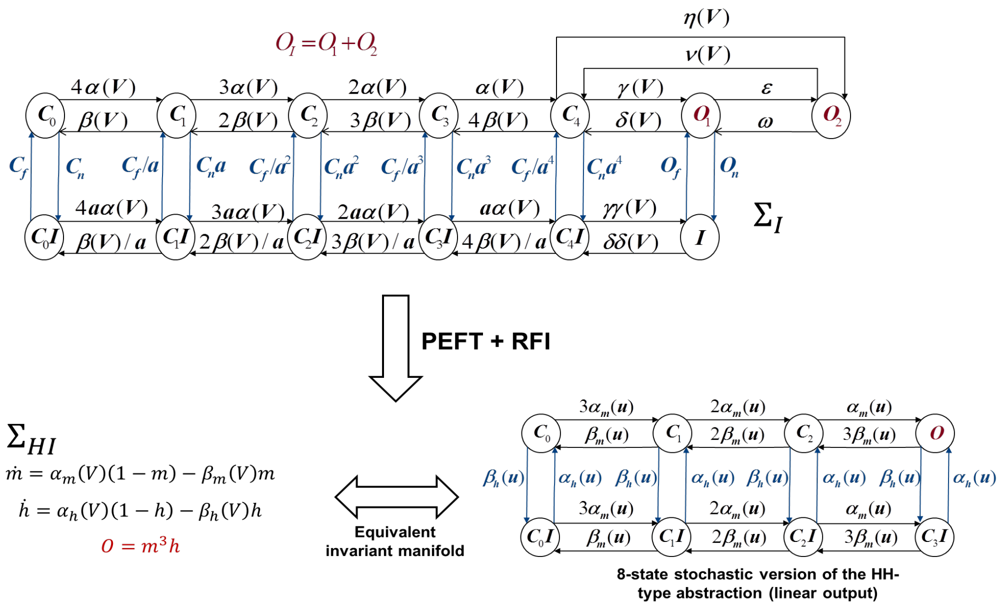

$O_I = O_1 + O_2$
Σ_I
PEFT + RFI
Σ_{HI}
$\dot{m} = \alpha_m(V)(1-m) - \beta_m(V)m$
$\dot{h} = \alpha_h(V)(1-h) - \beta_h(V)h$
$O = m^3 h$
Equivalent invariant manifold
8-state stochastic version of the HH-type abstraction (linear output)

Figure 4.4: Sodium channel models: Σ_I, Σ_{HI}, and its 8-variable stochastic version.

The output of Σ_{HI}, m^3h, represents the probability that three *activation (m-type)* gates and one *inactivation (h-type)* gate are open. As they are all independent, we obtain the net probability of the event as m^3h. This event corresponds to the occupancy probability of the state labeled O in voltage-controlled CTMC of Fig. 4.4, which is the output of this 8-state model. Similarly, the occupancy probability of each of the other 7 states corresponds to a certain number of m-type (maximum 3) and the h-type (maximum 1) gates being open. For example, the state labeled C_1 corresponds to one m-type gate and the h-type gate being open, resulting in a net probability of mh. Similarly, the state labeled C_1I corresponds to one m-type gate being open and the h-type gate being closed, resulting in a net probability of $m(1-h)$. Also see the discussion titled "Adapting the Abstraction Process to Arbitrary Observable Functions of Σ_I and Σ_K" in Section 3.3.

The 8-state stochastic version has dynamics of the form $\dot{\mathbf{y}}_I = A_H(V_{HI})$, where $\mathbf{y} \in \mathbb{R}^8$ is the vector representing the occupancy probability distribution among the eight states in the order $[C_0, C_1, C_2, O,$
$C_0I, C_1I, C_2I, C_3I]$. $A_H(V_{HI})$ is an 8×8 matrix similar to $A_I(V_{HI})$ in Definition 2.1.1. The linear system was converted to its affine form to remove the zero eigenvalue, as was done for Σ_I: $\dot{\mathbf{y}}'_I = A'_H(V_{HI}).\mathbf{y}'_I + B'_H(V_{HI})$, by eliminating the state labeled C_3I. The new state vector $\mathbf{y}'_I$ denotes the occupancy probabilities of the first 7 states from $\mathbf{y}_I$. The 7×7 rate matrix A'_H and the 7×1 affine term B_H can be found [33]. The output of this system is $\mathbf{y}_4$, corresponding to the occupancy probability of the state labeled O.

For a fixed voltage input $V_{HI} = v$, the equilibrium of the resulting linear system was shifted to the origin by solving $A'_H(v).\mathbf{y}'_I = -B'_H$. Similarly, the equilibrium of Σ_C, for a fixed conductance input, was shifted to the origin using an offset of V_{eq}.

II. SoS Optimization
We constructed two SoSPs: P_{IH} and P_C, to compute $S_{IH}(\mathbf{x}'_I, \mathbf{y}'_I)$ and $S_C(V_I, V_{HI})$, respectively. The two problems were then solved using SOSTOOLS. Next, we explain the construction of P_{IH} and P_C.

III. Choosing the form of the SoS BFs
Defining an instance of SoS optimization begins with declaring the form of the desired polynomial. We chose ellipsoidal forms for the BFs using the `sossosvar` function provided by SOSTOOLS: $S_{IH}(\mathbf{x}'_I, \mathbf{y}'_I) = [\mathbf{x}'_I, \mathbf{y}'_I].Q_{IH}.[\mathbf{x}'_I, \mathbf{y}'_I]^T$ and $S_C(V_I, V_{HI}) = [V_I, V_{HI}].Q_C.[V_I, V_{HI}]^T$. Variables $\mathbf{x}'_I, \mathbf{y}'_I, V_I$, and V_{HI} are declared using the `pvar` polynomial variable toolbox. The coefficients of the BFs, which form the decision variables of the SoS optimization problems, are contained in the positive semidefinite matrices Q_{IH} (19×19) and Q_C (2×2).

Eq. (4.2), the first constraint defining a BF, was implemented as:

$$P_{IH} : S_{IH} - ((\mathbf{x}'_{I6} + \mathbf{x}'_{I7}) - (\mathbf{y}'_{I4}))^2 \in \mathbb{S}, \text{ and}$$

$$P_C : S_C - (V_I - V_{HI})^2 \in \mathbb{S}.$$

IV. Input-space quantization
We sampled pairs of inputs to the two subsystems: Σ_I and Σ_{HI} for P_{IH}, and the

two copies of Σ_C for P_C. Eq. (4.3) was thus implemented as follows:

$$\begin{aligned} P_{IH} : - &\left[\frac{\partial S_{IH}}{\partial \mathbf{x}'_I} \left(A'_I(v_i).\mathbf{x}'_I \right) + \frac{\partial S_{IH}}{\partial \mathbf{y}'_I} \left(A'_H(v_j).\mathbf{y}'_I \right) \right] \\ &- \lambda_{IH} S_{IH}(\mathbf{x}'_I, \mathbf{y}'_I) + \gamma_{IH} |v_i - v_j| \in \mathbb{S}, \end{aligned} \tag{4.9}$$

$$\begin{aligned} P_C : - &\left[\frac{\partial S_C}{\partial V_I} \left(-G\ o_i\ V_I \right) + \frac{\partial S_C}{\partial V_{HI}} \left(-G\ o_j\ V_{HI} \right) \right] \\ &- \lambda_C S_C(V_I, V_{HI}) + \gamma_C |o_i - o_j| \in \mathbb{S}. \end{aligned} \tag{4.10}$$

For P_{IH}, the input pairs are $(v_i, v_j) \in \mathcal{V} \times \mathcal{V}$, where $\mathcal{V} = \{-60, -30, 0, 20, 30\}$. For P_C, the input pairs are $(o_i, o_j) \in \mathcal{O} \times \mathcal{O}$, where $\mathcal{O} = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. We use the dynamics of the system, with the origin shifted to the equilibrium corresponding to the input pair (v_i, v_j). Hence, there are no affine terms in the multipliers of $\partial S_{IH}/\partial \mathbf{x}'$, $\partial S_{IH}/\partial \mathbf{y}'$, $\partial S_C/\partial V_I$ and $\partial S_C/\partial V_{HI}$ terms. Note that each input pair contributes one constraint to the corresponding problem. Thus, Eq. (4.3) was implemented as 25 constraints in P_{IH} and 100 constraints in P_C.

The use of input-space sampling in computing BFs can be justified as follows. As per Theorem 4.3.2, BF S_{IH} bounds the difference in the outputs of Σ_I and Σ_{HI} when the maximum difference in the voltage (input) signals is $\parallel V_I - V_{HI} \parallel_\infty$. Voltage signals $V_I(t)$ and $V_{HI}(t)$ can be approximated by quantizing them using the set $\mathcal{V}$. At any point in time, the voltage signals would be rounded off to the nearest member of $\mathcal{V}$. The error in the outputs due to the quantization error in the inputs can be bound using sensitivity analysis. Then, the bound on the output difference given by S_{IH} would have to take into account the output error resulting from the quantization.

A similar analysis can be performed for S_C, where the conductance signals would now be quantized using $\mathcal{O}$. Providing revised bounds that reflect input-space quantization is part of our future work.

V. Handling the parameters

Eqs. (4.9) and (4.10) were implemented in SOSTOOLS with fixed values for parameters λ_{IH}, γ_{IH} of P_{IH}, and λ_C, γ_C of P_C. The parameter values we used are given in Table 4.1. We were unable to make λ_{IH} and λ_C decision variables of P_{IH} and P_C,

Problem	BF	λ	γ
P_{IH}	S_{IH}	$\lambda_{IH} = 0.1$	$\gamma_{IH} = 0.001$
P_C	S_C	$\lambda_C = 0.01$	$\gamma_C = 0.0001$

Table 4.1: SOSP1: parameter values used for the BFs.

respectively, as Eqs. (4.9) and (4.10) would have become nonlinear in the decision variables. Parameters γ_{IH} and γ_C, however, could have been declared as decision variables of P_{IH} and P_C, respectively. We tried this approach and also defined an objective function to minimize the amplification factor γ/λ of Theorem 4.3.2. The resulting BF was inferior to our current approach of defining the objective function (see below), and thus the parameters were fixed to the values provided in Table 4.1.

VI. Optimizing the BFs

Theorem 4.3.2 implies that $\forall t \geq 0$, the value of the BF bounds the difference in outputs observed from the two systems. Ideally, we want this bound to be as tight as possible. Also, the value of the BF at $t = 0$ is the highest value that it can assume along a pair of the trajectories (as it decays $\forall t > 0$).

To obtain a BF that provides tight bounds on the output difference, we implemented an objective function that minimizes the BF at the initial states of the two subsystems:

$$P_{IH} : \text{Minimize } S_{IH}(\mathbf{x}'_I(0), \mathbf{y}'_I(0)) \text{ and}$$
$$P_C : \text{Minimize } S_C(-30, -30),$$

where $\mathbf{x}'_I(0)$ and $\mathbf{y}'_I(0)$ represent the initial conditions of Σ_I as per Definition 2.1.1 and as per Section 3.3 for Σ_{HI}.

P_{IH} and P_C were solved in SOSTOOLS. In the process, SOSTOOLS outputs *feasratio*, *pinf*, *dinf* and *numerr*, which reflect the accuracy and reliability of the solutions. Both P_{IH} and P_C were solved by error-free executions with *feasratio* = 1 and *pinf*, *dinf*, *numerr* = 0, resulting in reliable and accurate BFs. SOSTOOLS took 956.17 seconds to solve P_{IH} and 2.43 seconds to solve P_C on an Intel i5 2.5 GHz CPU-based PC with 6 GB of memory.

The computed value of S_{IH}, as defined by the 19×19 matrix Q_{IH}, can be found in the supplementary document [33]. S_C, the BF between Σ_C and itself, was computed as:

$$S_C(V_I, V_{HI}) = 1.27V_I^2 - 1.4599V_I.V_{HI} + 1.27V_{HI}^2.$$

VII. Composing S_{IH} and S_C

The parameters of S_{IH} and S_C, given in Table 4.1, satisfy the SGC condition of Theorem 4.3.3, as $\frac{\gamma_{IH}\gamma_C}{\lambda_{IH}\lambda_C} = 0.0001 < 1$. Applying Theorem 4.3.3, we linearly composed S_{IH} and S_C to obtain $S_1 = \alpha_1 S_{IH} + \alpha_2 S_C$, where $\alpha_1, \alpha_2 = 1$. S_1 is a BF between the composite systems Σ_{CI} and Σ_{CHI}. As per Theorem 2 of [25], the parameter λ of S_1 can be calculated as

$$\lambda = min\left(\frac{\alpha_1\lambda_{IH} - \alpha_2\gamma_C}{\alpha_1}, \frac{\alpha_2\lambda_C - \alpha_1\gamma_{IH}}{\alpha_2}\right) = 0.009.$$

4.6 Visualizing BFs S_{IH}, S_C, and S_1

In this section, we experimentally validate BFs S_{IH}, S_C and S_1, which were obtained as described in the previous section. As per Theorem 4.3.2, BFs decay along a pair of trajectories of the two systems under consideration. To this end, we simulated Σ_I and Σ_{HI} (for S_{IH}), Σ_C (for S_C), and Σ_{CI} and Σ_{CHI} (for S_1) using different inputs and initial conditions, and evaluated the BFs along the resulting trajectories.

Empirical validation of the BFs is first provided by plotting them in 2D along the time axis. As the time proceeds in the same manner in both systems, the corresponding BF is plotted for the pair of states occurring at the same time along the trajectories of the systems. The squared difference in outputs observed for the pair of states is also plotted in the same graph. The resulting plots show that the BFs bound the (squared) Output Difference (OD) and decay in time along the pairs of trajectories, as per Theorem 4.3.2.

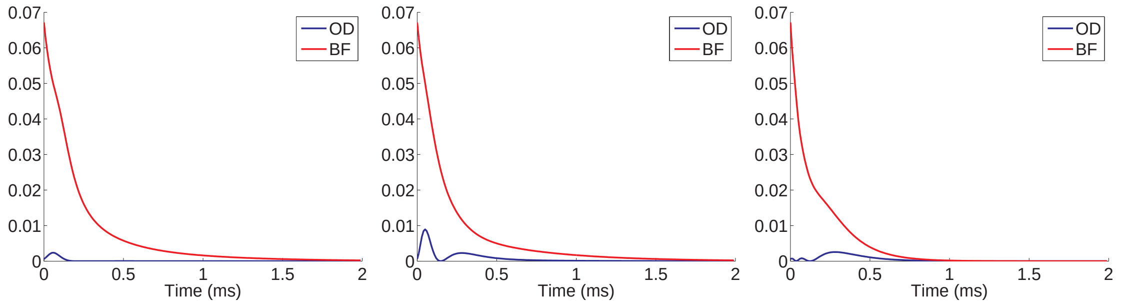
We also provide 3D plots, where the x- and y-axes measure time, and the BF along with the OD are plotted on the z-axis. This form of plotting allows us to visualize the fact that the BF upper bounds the difference in the outputs for all possible pairs of states. These plots also show that the BF decays along pairs of trajectories, even when there is a delay between the systems (off-diagonal states).

Fig. 4.5 shows S_{IH} plotted along three pairs of trajectories of Σ_I and Σ_{HI}. Each pair was generated by supplying a pair of constant voltage signals $(V_I(t), V_{HI}(t))$ as inputs to Σ_I and Σ_H, respectively. The two subsystems were initialized as per Defs. 2.1.1 and 3.2.1, and simulated using MATLAB's *ODE45* solver [57]. S_{IH} was then evaluated along the resulting pair of trajectories after shifting the origin to the equilibrium defined by $(V_I(t), V_{HI}(t))$.

Fig. 4.5(a) plots, in blue, the OD along two trajectories that receive the same input of -30 mV. S_{IH} is plotted in red and exhibits the decaying behavior predicted by Theorem 4.3.2. Fig. 4.5(b) plots the OD along two trajectories with the maximum possible difference in inputs: $V_I(t) = -30\ mV$ and $V_{HI}(t) = 30\ mV$. This results in a relatively large OD observed for the two subsystems. S_{IH} is shown to upper bound this difference and decay along the pair of trajectories. Fig. 4.5(c) inverts the inputs with $V_I(t) = 30\ mV$ and $V_{HI}(t) = -30\ mV$.

S_C characterizes the ability of Σ_C to tolerate small changes in the input conductance signals. In the composite systems Σ_{CI} and Σ_{CHI}, these signals are provided by subsystems Σ_I and Σ_{HI}, and thus vary slightly due to the fitting errors incurred by the abstraction process of Section 3.3.

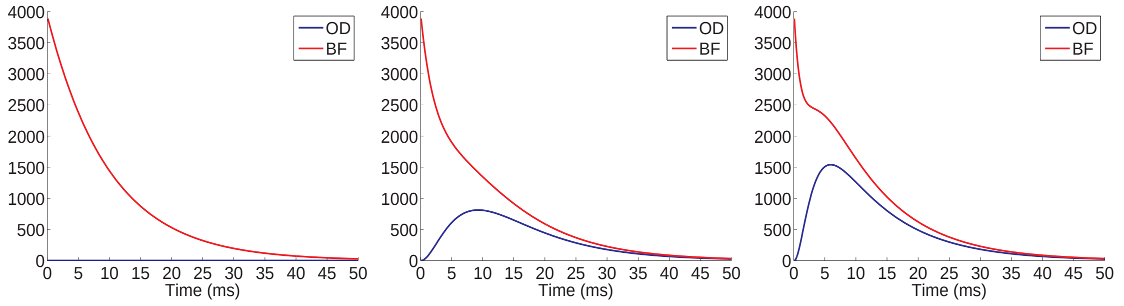
As in Fig. 4.5, S_C is plotted in Fig. 4.6 along three pairs of trajectories of Σ_C. Each pair of trajectories was generated by supplying constant conductance (input)



(a) $V_I(t) = V_{HI}(t) = -30\ mV$. (b) $V_I(t) = -30\ mV$, $V_{HI}(t) = 30\ mV$. (c) $V_I(t) = 30\ mV$, $V_{HI}(t) = -30\ mV$.

Figure 4.5: BF S_{IH} and the OD plotted along three pairs of trajectories of Σ_I and Σ_{HI} generated using constant voltage (input) signals. S_{IH} upper bounds the OD and decays along the trajectories.

signals ($O_I(t)$, $O_{HI}(t)$). Σ_C was initialized at $-30\ mV$ and simulated using the Euler method. S_C was evaluated along the resulting trajectories after shifting the origin to the equilibrium, 30 mV (V_{eq}). Fig. 4.6(a) shows the case when both $O_I(t)$ and $O_{HI}(t)$ were 0.01, resulting in equal voltage traces with an OD of 0. We have not scaled the systems nor the BFs; thus the scale of Fig. 4.6 is very different from that of Fig. 4.5. We can see that even when the input conductances vary by a factor of 8, in Fig. 4.6(c), S_C bounds the OD and decays along the trajectories.

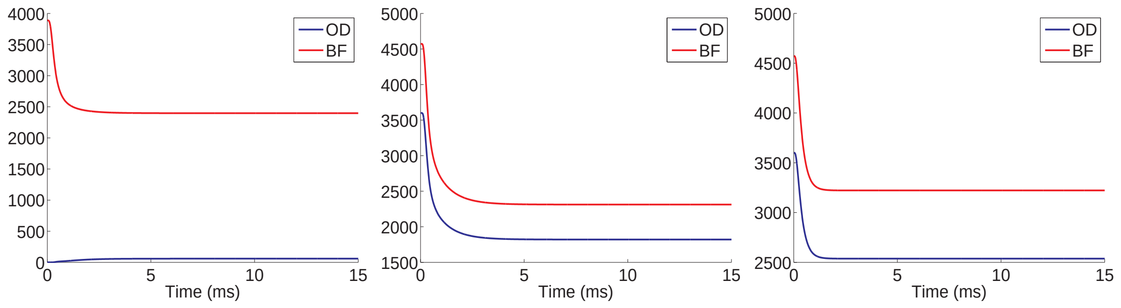


(a) $O_I(t) = 0.01$, $O_{HI}(t) = 0.01$. (b) $O_I(t) = 0.04$, $O_{HI}(t) = 0.01$. (c) $O_I(t) = 0.08$, $O_{HI}(t) = 0.01$.

Figure 4.6: BF S_C and the OD plotted along three pairs of trajectories of Σ_C generated using constant conductance (input) signals. S_C bounds OD even when input signals vary by a factor of 8 (subfig. c).

CCMs Σ_{CI} and Σ_{CHI} are autonomous dynamical systems and do not receive any external inputs. To visualize the composite BF S_1, we simulated Σ_{CI} and Σ_{CHI} using the Euler method. Fig. 4.8 plots the trajectories obtained from these simulations. The corresponding conductance traces of Fig. 4.8(a) and the voltage traces of Fig. 4.8(b) empirically validate that the composed models are approximately equivalent as predicted by Theorem 4.3.3. BF S_1 along this pair of trajectories is plotted in Fig. 4.7(a). The value of S_1 is dominated by the value of S_C, as it bounds the squared difference of voltages and is much larger than S_{IH}, which bounds differences in probabilities. This is reasonable as voltage is the primary entity of interest when analyzing excitable cells. One could scale subsystem Σ_C such that its output lies in $[0,1]$ and is thus comparable to the outputs of Σ_I and Σ_{HI}.

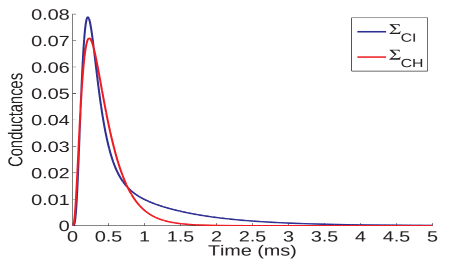
To test extreme cases, we simulated Σ_{CI} and Σ_{CHI} by initializing Σ_C at different values. Fig. 4.7(b) shows the OD plotted along pairs of trajectories, where Σ_C in Σ_{CI} starts at $V_I(0) = -30\ mV$ and in Σ_{CHI} at $V_{HI}(0) = 30\ mV$. Fig. 4.7(c) plots the other extreme, where the copy of Σ_C in Σ_{CI} starts at 30 mV and the copy in Σ_{CHI} is initialized to $-30\ mV$. S_1 bounds the OD in all these cases and decays along the trajectories.



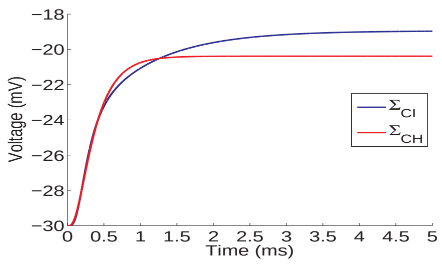
(a) $V_I(0) = -30\ mV$, $V_{HI}(0) = -30\ mV$. (b) $V_I(0) = -30\ mV$, $V_{HI}(0) = 30\ mV$. (c) $V_I(0) = 30\ mV$ $V_{HI}(0) = -30\ mV$.

Figure 4.7: S_1 and OD along pairs of trajectories of Σ_{CI} and Σ_{CHI}. Subfig. (a) plots OD and S_1 along trajectories shown in Fig. 4.8. Subfigs. (b) and (c) plot OD and S_1 along trajectories where Σ_C is initialized at different voltages. Value of OD is dominated by difference of outputs of the two copies of Σ_C, which is in mV. Similarly, value of S_1 is dominated by S_C, which bounds differences in voltages as opposed to S_{IH}, which bounds differences in probabilities.

Fig. 4.9 provides 3D views of S_{IH}, S_C, and S_1. Fig. 4.9(a) was generated as follows. Σ_I an Σ_{HI} were simulated using constant voltage input signals of $V_I(t) = -30$ mV and $V_{HI}(t) = 30$ mV. The two models were simulated using *ODE45*, starting from the nominal initial conditions specified in Defs. 2.1.1 and 3.2.1 until steady state was reached. Let T denote the time steps $[t_1, t_2, ..., t_n]$ of the two discrete-time simulations, and $O_I(t)$ and $O_{HI}(t)$ denote the resulting conductance (output) time series, with their origins shifted to the respective equilibria. Fig. 4.9(a) plots in red the squared output difference $(O_I(t_i) - O_{HI}(t_j))^2$ for all $(t_i, t_j) \in T \times T$.



(a) $O_I(t)$ and $O_{HI}(t)$ of Σ_{CI} and Σ_{CHI} respectively.



(b) $V_I(t)$ and $V_{HI}(t)$ of Σ_{CI} and Σ_{CHI} respectively.

Figure 4.8: Simulations of Σ_{CI} and Σ_{CHI}: when Σ_I is replaced by Σ_{HI}, feedback composition tends to accumulate the error incurred due to the abstract component. S_1 proves that these errors remain bounded due to the approximate equivalence of Σ_I and Σ_{HI}, established by BF S_{IH}, and the ability of Σ_C to tolerate deviations in the conductance inputs, established by BF S_C. The mean L1 errors: $O_{Na} : 9 \times 10^{-3}$, V: 1.42 mV.

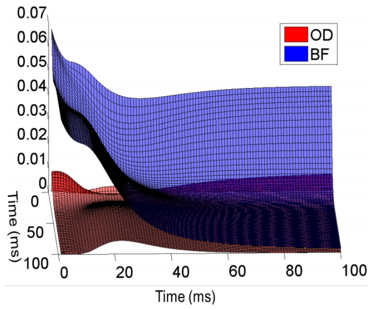
S_{IH} is plotted at the pair of states $(\mathbf{x}'_I(t_i), \mathbf{y}'_I(t_j))$ obtained in the simulation for all $(t_i, t_j) \in T \times T$, where $\mathbf{x}'_I$ and $\mathbf{y}'_I$ denote the origin-shifted versions of $\mathbf{x}_I$ and $\mathbf{y}_I$, respectively. The 3D view of S_{IH} shows that the BF provides an upper bound for the OD at all pairs of states of the two systems. It also shows that S_{IH} decays along the trajectories even when the two systems have a delay between them.

Fig. 4.9(b) was obtained by simulating Σ_C with two constant conductance signals of 0.1 and 0.01. The resulting voltage output signals differ significantly due to a 10-fold difference in the input signals. Fig. 4.9(c), on the other hand, was obtained by simulating Σ_{CI} and Σ_{CHI} using different initial conditions. In both the cases, the respective BFs S_C and S_1 bound the output difference and decay along the trajectories (even when the two systems have a delay between them).

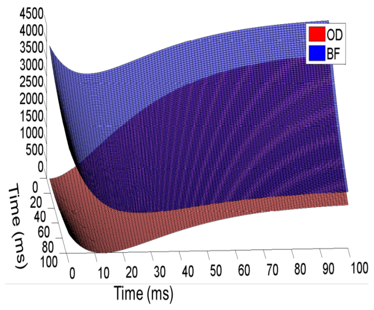
Eq. (4.8) of SOSP1 enforces Eq. (4.3) on a sampled subset of the entire input space. In the next section, we address this restriction, and present a revised BF computation procedure that enforces Eq. (4.3) on the entire input space, as opposed to the input-quantization-based approach of SOSP 1.

4.7 BF Computation: Version 2.0

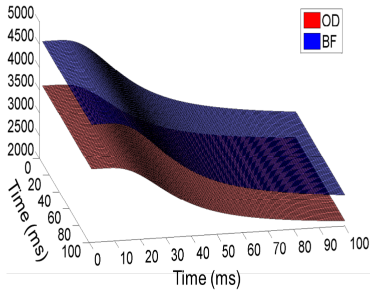
In this section, we describe a revised computation procedure for BFs. In the previous section, we presented an algorithm, based on [35], for computing BFs using SoS optimization, which used input-space quantization. In this section, we modify the SoSP formulation of [35] to exhaustively cover the input-space. Then, we show that the solutions of our SoSP are indeed BFs.



(a) $V_I(t) = -30$ mV, $V_{HI} = 30$ mV.



(b) $O_I(t) = 0.1$, $O_{HI} = 0.01$.



(c) $V_I(0) = -30$ mV, $V_{HI}(0) = 30$ mV.

Figure 4.9: 3D visualization of BFs (in blue): S_{IH} in subfig. (a), S_C in subfig. (b) and S_1 in subfig. (c) are plotted for pairs of states obtained at time points $(t_i, t_j) \in T \times T$, where $T = [t_1, t_2, ...]$ are the time steps of discrete-time simulations of the corresponding pairs of systems. The BFs upper bound the OD, plotted in red for all pairs of states, and decay along the trajectories even when there is delay between the two systems. The captions of subfigs. (a)-(b) specify the input signals used for simulation and the caption of subfig. (c) provides the initial conditions used for Σ_C while simulating Σ_{CI} and Σ_{CHI}.

We assume that the input spaces are described using sets, such as $\mathcal{U} = \{u \in \mathbb{R} : \rho(u) \geq 0\}$, where $\rho(u)$ is called a *descriptor function*. For example, $\rho(u) = (u - u_{min})(u_{max} - u)$ describes the input-space $\mathcal{U} = [u_{min}, u_{max}]$.

Definition 4.7.1. Consider two dynamical systems $\Sigma_i = (X_i, \mathbf{x}_i^0, [u_{min}, u_{max}], f_i, \mathcal{O}, g_i)$, $i = 1, 2$. The SoSP formulation for a BF, $S \in \mathbb{S}$, between the systems is given by:

SOSP 2:

$$\text{Minimize } S(\mathbf{x}_1^0, \mathbf{x}_2^0) \tag{4.11}$$

subject to:

$$-S(\mathbf{x}_1, \mathbf{x}_2) + [g_1(\mathbf{x}_1) - g_2(\mathbf{x}_2)]^2 \in \mathbb{S}, \tag{4.12}$$

$$\forall u_1 \in [u_{min}, u_{max}], u_2 \in [u_{min}, u_{max}], \exists \lambda > 0, \gamma \geq 0, \sigma_1(\mathbf{x}_1, u_1) \in \mathbb{S}, \sigma_2(\mathbf{x}_2, u_2) \in \mathbb{S} \text{ such that } : \tag{4.13}$$

$$-\frac{\partial S}{\partial \mathbf{x}_1} f_1(\mathbf{x}_1, u_1) - \frac{\partial S}{\partial \mathbf{x}_2} f_2(\mathbf{x}_2, u_2) - \lambda S(\mathbf{x}_1, \mathbf{x}_2) + \gamma(u_1 - u_2)^2 - \sigma_1(\mathbf{x}_1, u_1)\rho(u_1) - \sigma_2(\mathbf{x}_2, u_2)\rho(u_2) \in \mathbb{S}.$$

Next, we show that the feasible solutions of the SoSP in Definition 4.7.1 are indeed BFs for the two systems.

Proposition 4.7.2. *Consider a feasible solution, $(S, \sigma_1, \sigma_2, \lambda, \gamma)$, of the SoSP in Definition 4.7.1. S satisfies Eqs. (4.2) and (4.3), and thus is a BF between Σ_1 and Σ_2.*

Proof. A feasible solution $(S, \sigma_1, \sigma_2, \lambda, \gamma)$ satisfies Eq. (4.12), i.e.

$$\forall \mathbf{x}_1, \mathbf{x}_2 : -S(\mathbf{x}_1, \mathbf{x}_2) + [g_1(\mathbf{x}_1) - g_2(\mathbf{x}_2)]^2 \in \mathbb{S}.$$

As an SoS polynomial is always non-negative, we get

$$-S(\mathbf{x}_1, \mathbf{x}_2) + [g_1(\mathbf{x}_1) - g_2(\mathbf{x}_2)]^2 \geq 0,$$

which implies

$$[g_1(\mathbf{x}_1) - g_2(\mathbf{x}_2)]^2 \leq -S(\mathbf{x}_1, \mathbf{x}_2),$$

i.e. S satisfies Eq. (4.2).

The feasibility of $(S, \sigma_1, \sigma_2, \lambda, \gamma)$ implies that Eq. (4.13) is satisfied, i.e.

$$\begin{aligned}&-\frac{\partial S}{\partial \mathbf{x}_1} f_1(\mathbf{x}_1, u_1) - \frac{\partial S}{\partial \mathbf{x}_2} f_2(\mathbf{x}_2, u_2) - \lambda S(\mathbf{x}_1, \mathbf{x}_2) + \gamma(u_1 - u_2)^2 \dots \\ &- \sigma_1(\mathbf{x}_1, u_1)\rho(u_1) - \sigma_2(\mathbf{x}_2, u_2)\rho(u_2) \in \mathbb{S}.\end{aligned}$$

Non-negativity of SoS polynomials results in

$$\begin{aligned}&-\frac{\partial S}{\partial \mathbf{x}_1} f_1(\mathbf{x}_1, u_1) - \frac{\partial S}{\partial \mathbf{x}_2} f_2(\mathbf{x}_2, u_2) - \lambda S(\mathbf{x}_1, \mathbf{x}_2) + \gamma(u_1 - u_2)^2 \dots \\ &- \sigma_1(\mathbf{x}_1, u_1)\rho(u_1) - \sigma_2(\mathbf{x}_2, u_2)\rho(u_2) \geq 0,\end{aligned}$$

which implies

$$\begin{aligned}-\frac{\partial S}{\partial \mathbf{x}_1} f_1(\mathbf{x}_1, u_1) - \frac{\partial S}{\partial \mathbf{x}_2} f_2(\mathbf{x}_2, u_2) - \sigma_1(\mathbf{x}_1, u_1)\rho(u_1) - \sigma_2(\mathbf{x}_2, u_2)\rho(u_2) \\ \geq \lambda S(\mathbf{x}_1, \mathbf{x}_2) - \gamma(u_1 - u_2)^2.\end{aligned}$$

Multiplying both sides by -1 and then reversing the inequality, we get

$$\begin{aligned}\frac{\partial S}{\partial \mathbf{x}_1} f_1(\mathbf{x}_1, u_1) + \frac{\partial S}{\partial \mathbf{x}_2} f_2(\mathbf{x}_2, u_2) + \sigma_1(\mathbf{x}_1, u_1)\rho(u_1) + \sigma_2(\mathbf{x}_2, u_2)\rho(u_2) \\ \leq -\lambda S(\mathbf{x}_1, \mathbf{x}_2) + \gamma(u_1 - u_2)^2.\end{aligned}$$

As $\sigma_1(\mathbf{x}_1, u_1)\rho(u_1) + \sigma_2(\mathbf{x}_2, u_2)\rho(u_2)$ is always non-negative, we can eliminate it and still retain the inequality to get

$$\frac{\partial S}{\partial \mathbf{x}_1} f_1(\mathbf{x}_1, u_1) + \frac{\partial S}{\partial \mathbf{x}_2} f_2(\mathbf{x}_2, u_2) \leq -\lambda S(\mathbf{x}_1, \mathbf{x}_2) + \gamma(u_1 - u_2)^2.$$

□

4.7.1 Polynomialization of Ion-Channel Rate Matrices

In this section, we focus on implementing the SoSP of Defn. 4.7.1 in automated solvers such as MATLAB SOSTOOLS [65]. Specifically, we focus on the restriction that only polynomial vector fields, denoted by $f_i(\mathbf{x}_i, u_i)$, $i = 1, 2$ in Eq. (4.13), can be specified in SOSTOOLS. In other words, f_i must be a polynomial function of $\mathbf{x}_i$ and u_i. In the case of cardiac-cell models, the ion-channel subsystems do not satisfy this requirement. Their dynamics usually take the form $\dot{\mathbf{x}} = A(V).\mathbf{x}$, where $\mathbf{x}$ is the occupancy-probability vector, V denotes the input membrane potential and A is a rate matrix, whose entries are *exponential functions of* V. Thus, the dynamics do not have a polynomial form. We present a workaround for our ion channel models.

We transformed the rate matrix $A_K(V)$ to an approximately equivalent matrix $A^p(V)$ by fitting the entries of A_K with polynomial functions, see Fig. 4.10.

$$\dot{\mathbf{x}} = A(V)\,\mathbf{x} \qquad\qquad \dot{\mathbf{x}} = A^p(V)\,\mathbf{x}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} a_{11}(V) & \cdots & a_{1n}(V) \\ \vdots & \ddots & \vdots \\ a_{n1}(V) & \cdots & a_{nn}(V) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \cong \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} a^p_{11}(V) & \cdots & a^p_{1n}(V) \\ \vdots & \ddots & \vdots \\ a^p_{n1}(V) & \cdots & a^p_{nn}(V) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$a_{ij}(V) = b_{ij}e^{c_{ij}V+d_{ij}} \xrightarrow[\text{approximation}]{\text{Polynomial}} a^p_{ij}(V) = q^{ij}_0 + q^{ij}_1 V + q^{ij}_2 V^2 + \ldots + q^{ij}_m V^m$$

Figure 4.10: Polynomialization of vector fields of ion-channel models. Each of the rate-functions of A are fit with polynomial functions of varying degrees.

We used MATLAB's curve-fitting tool *cftool* [58] to obtain the polynomial fits of the functions. The original rate functions and their polynomial approximations for Σ_K are shown in Table 4.2.

Rate	Original function	Polynomial Approximation
$\alpha_a(V)$	$0.5437e^{0.029V}$	$(1.654 \times 10^{-8})V^4 + (2.301 \times 10^{-6})V^3 + 0.0002282\ V^2 + 0.01574\ V + 0.5437$
$\beta_a(V)$	$0.0802e^{-0.0468V}$	$(1.759\times10^{-8})V^4-(1.532\times10^{-6})V^3+(8.752\times 10^{-5})V^2 - 0.003725V + 0.0802$
$\alpha_i(V)$	$0.0498e^{-0.0004V}$	$(-1.859 \times 10^{-5})V + 0.04984$
$\beta_i(V)$	$0.0008e^{(5.374\times10^{-8}V)}$	$(4.404 \times 10^{-11})V + 0.0008195$

Table 4.2: Transfer rates of Σ_K, which is illustrated in Fig. 2.4, and defined in Def. 2.1.2. The polynomial approximations were obtained by fitting the the original exponential functions, which are given in Table 2.2, by polynomials for $V \in [-30, 30]$.

The rate functions $\alpha_m(V)$, $\beta_m(V)$, $\alpha_h(V)$, and $\beta_h(V)$, which were identified using PEFT and RFI in Section 3.3, see Eqs. (3.7) - (3.10), were also fit using polynomial functions. Table 4.3 contains the polynomial versions of the rate functions.

Rate	Polynomial Approximation
$\alpha_m(V)$	$(2.195 \times 10^{-6})V^3 + (0.0002444)V^2 + (0.01572)V + 0.5385$
$\beta_m(V)$	$(-1.884 \times 10^{-6})V^3 + (0.0001333)V^2 - 0.004262V + 0.04788$
$\alpha_h(V)$	$1.022 \times 10^{-10}V^4 - (6.407 \times 10^{-9})V^3 + (1.678 \times 10^{-7})V^2 - (2.938 \times 10^{-6})V + 0.0001217$
$\beta_h(V)$	$(2.849 \times 10^{-8})V^4 + (3.179 \times 10^{-7})V^3 - (6.913 \times 10^{-5})V.^2 + (0.001316)V + 0.1115$

Table 4.3: Transfer rates of Σ_{HK} and its 10-variable stochastic version, which is illustrated in Fig. 4.11. The polynomial approximations were obtained by fitting the the original exponential functions, which were identified in Section 3.3, by polynomials for $V \in [-30, 30]$.

Based on the polynomialized versions of Σ_K and Σ_{KH}, we proceed to computing S_{KH}, S_C, and S_2 using SOSP 2.

4.7.2 Computing S_{KH}, S_C, and S_2 in SOSTOOLS

Next, we show how to compute the BFs S_{KH}, S_C, and S_2 by implementing SOSP 2 in MATLAB SOSTOOLS. S_{KH} is a BF between Σ_K and Σ_{HK}, with parameters λ_{KH} and γ_{KH}, while S_C is a BF between Σ_C and itself, with parameters λ_C and γ_C. Values for these parameters must be determined such that the SGC condition of Theorem 4.3.3 is satisfied. Then, we can linearly compose S_{KH} and S_C to obtain S_2, a BF between the composed systems Σ_{CK} and Σ_{CKH}.

I. Rewriting the system definitions
The first step is to rewrite the system definitions, as was done for computing S_{IH} and S_C. The rate matrix $A_K(V_K)$ of Σ_K has a zero eigenvalue for all $V_K \in [-30, 30]$, the values the input can take when Σ_K is composed with Σ_C. The system dynamics can be rewritten to make the new rate matrix Hurwitz, which is a necessary and sufficient condition for exponential stability. As the system is a voltage-controlled CTMC, one of the variables is always redundant; i.e., the occupancy probability of one state can be expressed as 1 - (sum of all other occupancy probabilities).

Using this fact, we removed the redundant state occupancy probability corresponding to state OI of Σ_K, see Fig. 2.4. The resulting dynamics are affine:

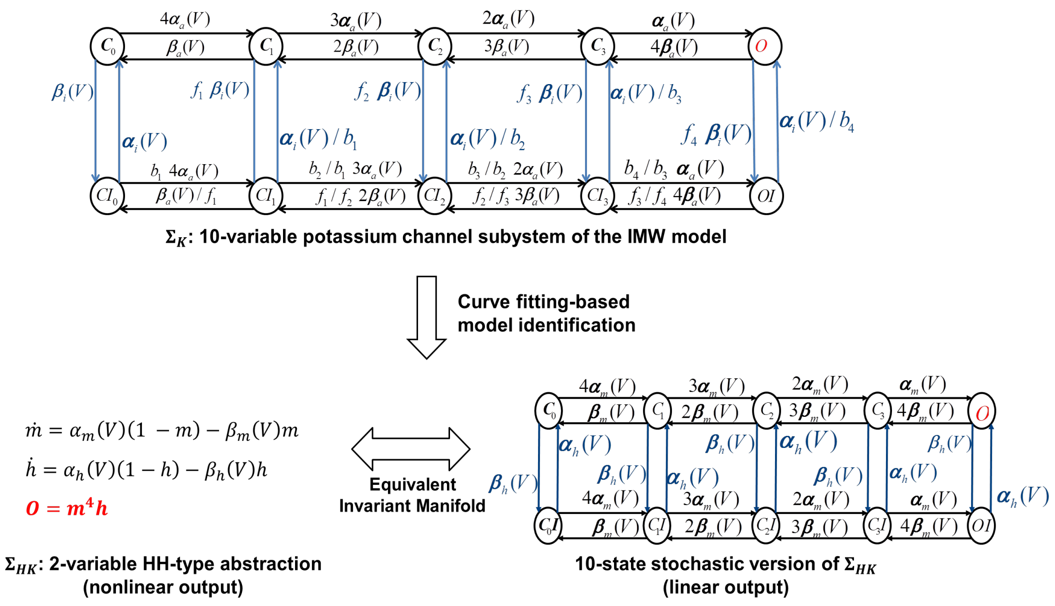

Σ_K: 10-variable potassium channel subystem of the IMW model
Curve fitting-based model identification
$\dot{m} = \alpha_m(V)(1-m) - \beta_m(V)m$
$\dot{h} = \alpha_h(V)(1-h) - \beta_h(V)h$
$0 = m^4h$
Equivalent Invariant Manifold
Σ_{HK}: 2-variable HH-type abstraction (nonlinear output)
10-state stochastic version of Σ_{HK} (linear output)

Figure 4.11: Potassium channel models: Σ_K, Σ_{HK}, and its 10-variable stochastic version.

$\dot{\mathbf{x}}'_K = A'_K(V_K).\mathbf{x}'_K + B_K(V_K)$, where the state vector $\mathbf{x}'_K$ now contains the first 9 probabilities from $\mathbf{x}_K$. The 9×9 matrix A'_K and the 9×1 affine term B_K can be found in [33]. For a fixed voltage input $V_I = v$, the equilibrium of the resulting linear system was shifted to the origin by solving $A'_I(v).\mathbf{x}'_I = -B_I$.

Σ_{HK} has an output function of degree 5, which would require S_{IH} to be an 10^{th}-order polynomial. Higher-order polynomials lead to possibly intractable instances of SoS optimization in SOSTOOLS. To resolve this issue, the 2-state system was converted to an equivalent 10-state stochastic model that has a linear output function. In [47], Σ_{HK} is shown to be the exact invariant manifold of the 10-state voltage-controlled CTMC shown in Fig. 4.11. Then, the redundant variable corresponding to the state OI was removed as was done for Σ_K. S_{KH} is then defined as a quadratic function over 18 (9 + 9) variables, which ensures that the corresponding SOS problem can be solved despite the increase in the number of undetermined coefficients.

Similarly, the equilibrium of Σ_C, for a fixed conductance input, was shifted to the origin using an offset of V_{eq}.

II. SoS Formulation

We constructed two SoSPs: P_{KH} and P_C, to compute $S_{KH}(\mathbf{x}'_K, \mathbf{y}'_K)$ and $S_C(V_K, V_{HK})$, respectively. The two problems were then solved using SOSTOOLS. Next, we explain the construction of P_{KH} and P_C.

III. Choosing the form of the SoS BFs
We chose ellipsoidal forms for the BFs using the `sossosvar` function provided by SOSTOOLS: $S_{KH}(\mathbf{x}'_K, \mathbf{y}'_K) = [\mathbf{x}'_K, \mathbf{y}'_K].Q_{KH}.[\mathbf{x}'_K, \mathbf{y}'_K]^T$ and $S_C(V_K, V_{HK})$ $= [V_K, V_{HK}].Q_C.[V_K, V_{HK}]^T$. Variables $\mathbf{x}'_K, \mathbf{y}'_K, V_K$, and V_{HK} are declared using the `pvar` polynomial variable toolbox. The coefficients of the BFs, which form the decision variables of the SOS optimization problems, are contained in the positive semidefinite matrices Q_{KH} (18×18) and Q_C (2×2).

Eq. (4.2), the first constraint defining a BF, was implemented as:

$$P_{KH} : S_{KH} - (\mathbf{x}'_{K5} - \mathbf{y}'_{K5})^2 \in \mathbb{S}, \text{ and}$$

$$P_C : S_C - (V_K - V_{HK})^2 \in \mathbb{S}.$$

IV. Implementing SOSP 2
In addition to the BF S and the parameters λ and γ, SOSP 2 also has two other unknowns: $\sigma_1(\mathbf{x}_1, \mathbf{u}_1)$ and $\sigma_2(\mathbf{x}_2, \mathbf{u}_2)$. In Eq. (4.13), these functions strengthen the decay condition of Eq. (4.3). In our SOSP problems, P_{KH} and P_C, we declare the ellipsoidal functions $\sigma_1^{KH}(\mathbf{x}'_K, V_K)$, $\sigma_2^{KH}(\mathbf{y}'_K, V_{KH})$, for P_{KH}, and $\sigma_1^C(V_K, O_K)$, $\sigma_2^K(V_{KH}, O_{KH})$, for P_C.

Eq. (4.13) is implemented in P_{KH} as

$$\begin{aligned}&-\frac{\partial S_{KH}}{\partial \mathbf{x}'_K} A'_K(V_K)\mathbf{x}'_K - \frac{\partial S_{KH}}{\partial \mathbf{y}'_K} A'_{KH}(V_{KH})\mathbf{y}'_K - \lambda_{KH} S_{KH}(\mathbf{x}'_K, \mathbf{y}'_K) + \\ &\gamma_{KH}(V_K - V_{KH})^2 - \sigma_1^{KH}(\mathbf{x}'_K, V_K)\rho_V(V_K) - \sigma_2^{KH}(\mathbf{y}'_K, V_{KH})\rho_V(V_{KH}) \in \mathbb{S},\end{aligned}$$

where $\rho_V(v) = (30 - v)(v + 30)$, which is always positive when $v \in [-30, 30]$. Eq. (4.13) is implemented in P_C as

$$\begin{aligned}&-\frac{\partial S_C}{\partial V_K}(-G\ V_K\ O_K) - \frac{\partial S_C}{\partial V_{KH}}(-G\ V_{KH}\ O_{KH}) - \lambda_C S_C(V_K, V_{KH}) + \\ &\gamma_C(O_K - O_{KH})^2 - \sigma_1^C(V_K, O_K)\rho_O(O_K) - \sigma_2^C(V_{KH}, O_{KH})\rho_O(O_{KH}) \in \mathbb{S},\end{aligned}$$

where $\rho_O(o) = (1-o)(o)$, which is always positive when the conductance $o \in [o_{min}, 1]$. The lower bound $o_{min} = 10^{-4}$, as making it 0 leads to infeasibility in SOSTOOLS.

Another important implementation aspect is the use of the *sparse* option. SOSTOOLS offers the option of using a sparse representation of the inequality constraints. This option constructs a memory-efficient representation of the constraints using the default *convhull* function. P_{KH} becomes infeasible when convhull is used, due to the large 18×18 symbolic matrix Q_{KH}. To resolve this, we used the CDD convex hull package, as suggested Section 3.4.3 of the user guide [65].

V. Handling the parameters
The parameter values we used are given in Table 4.4.

Problem	BF	λ	γ
P_{KH}	S_{KH}	$\lambda_{KH} = 0.01$	$\gamma_{KH} = 0.001$
P_C	S_C	$\lambda_C = 0.00001$	$\gamma_C = 0.00001$

Table 4.4: SOSP2: parameter values used for the BFs.

VI. Optimizing the BFs
To obtain a BF that provides tight bounds on the output difference, we implemented an objective function that minimizes the BF at the initial states of the two subsystems:

$$\begin{aligned} P_{KH} &: \text{Minimize } S_{KH}(\mathbf{x}'_K(0), \mathbf{y}'_K(0)) \text{ and} \\ P_C &: \text{Minimize } S_C(30, 30), \end{aligned}$$

where $\mathbf{x}'_K(0)$ and $\mathbf{y}'_K(0)$ represent the initial conditions of Σ_K as per Definition 2.1.2 and as per Section 3.3 for Σ_{HK}.

The computed value of S_{KH}, as defined by the 18×18 matrix Q_{KH}, can be found in the supplementary document [33]. S_C, the BF between Σ_C and itself, was computed as:

$$S_C(V_K, V_{HK}) = 1.9666V_K^2 - 0.066847V_K.V_{HK} + 1.9666V_{HK}^2.$$

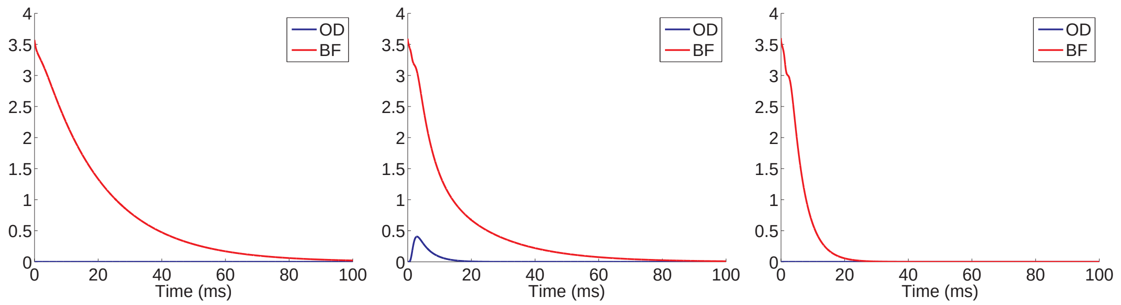
VII. Composing S_{KH} and S_C
The parameters of S_{KH} and S_C, given in Table 4.4, satisfy the SGC condition of Theorem 4.3.3, as $\frac{\gamma_{KH}\gamma_C}{\lambda_{KH}\lambda_C} = 0.0001 < 1$. Applying Theorem 4.3.3, we linearly composed S_{KH} and S_C to obtain $S_2 = \alpha_1 S_{KH} + \alpha_2 S_C$, where $\alpha_1 = 1, \alpha_2 = 100$. S_1 is a BF between the composite systems Σ_{CK} and Σ_{CHK}. As per Theorem 2 of [25], the parameter λ of S_1 can be calculated as

$$\lambda = min\left(\frac{\alpha_1\lambda_{KH} - \alpha_2\gamma_C}{\alpha_1}, \frac{\alpha_2\lambda_C - \alpha_1\gamma_{KH}}{\alpha_2}\right) = 0.001.$$

4.8 Visualizing BFs S_{KH}, S_C, and S_2

In this section, we plot the BFs S_{KH}, S_C, and S_2 along the trajectories of the corresponding pairs of systems, and show that they satisfy Eqs. (4.2) and Eqs. (4.3).

Fig. 4.12 shows S_{KH} plotted along three pairs of trajectories of Σ_K and Σ_{HK}. Each pair was generated by supplying a pair of constant voltage signals $(V_K(t), V_{HK}(t))$ as inputs to Σ_K and Σ_{HK}, respectively. The two subsystems were initialized as per Defs. 2.1.2 and 3.2.1, and simulated using MATLAB's *ODE45* solver [57]. S_{KH} was then evaluated along the resulting pair of trajectories after shifting the origin to the equilibrium defined by $(V_K(t), V_{HK}(t))$.



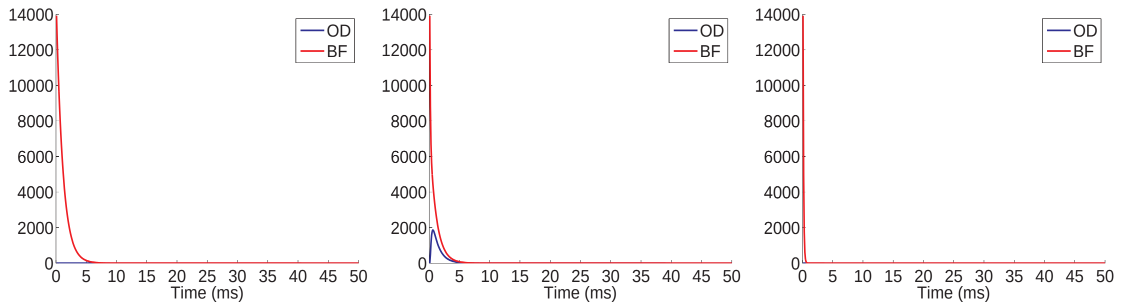
(a) $V_K(t) = V_{HK}(t) = -30\ mV$. (b) $V_K(t) = -30\ mV$, $V_{HK}(t) = 30\ mV$. (c) $V_K(t) = 30\ mV$, $V_{HK}(t) = 30\ mV$.

Figure 4.12: BF S_{KH} and the OD plotted along three pairs of trajectories of Σ_K and Σ_H generated using constant voltage (input) signals. S_{KH} upper bounds the OD and decays along the trajectories.

Fig. 4.12(a) plots, in blue, the OD along two trajectories that receive the same input of -30 mV. S_{KH} is plotted in red and exhibits the decaying behavior predicted by Theorem 4.3.2. Fig. 4.12(b) plots the OD along two trajectories with the maximum possible difference in inputs: $V_K(t) = -30\ mV$ and $V_{HK}(t) = 30\ mV$. This results in a relatively large OD observed for the two subsystems. S_{KH} is shown to upper bound this difference and decay along the pair of trajectories. Fig. 4.12(c) inverts the inputs with $V_K(t) = 30\ mV$ and $V_{HK}(t) = -30\ mV$.

S_C characterizes the ability of Σ_C to tolerate small changes in the input conductance signals. In the composite systems Σ_{CI} and Σ_{CHI}, these signals are provided by subsystems Σ_I and Σ_{HI}, and thus vary slightly due to the fitting errors incurred by the abstraction process of Section 3.3.

As in Fig. 4.12, S_C is plotted in Fig. 4.13 along three pairs of trajectories of Σ_C. Each pair of trajectories was generated by supplying constant conductance (input) signals ($O_K(t)$, $O_{HK}(t)$). Σ_C was initialized at 30 mV and simulated using the Euler method. S_C was evaluated along the resulting trajectories after shifting the origin to the equilibrium, $-30\ mV$ (V_{eq}). Fig. 4.13(a) shows the case when both $O_K(t)$ and $O_{HK}(t)$ were 0.1, resulting in equal voltage traces with an OD of 0. We have not scaled the systems nor the BFs; thus the scale of Fig. 4.13 is very different from that of Fig. 4.12. We can see that even when the input conductances vary by



(a) $O_K(t) = O_{HK}(t) = 0.1$. (b) $O_K(t) = 0.1, O_{HK}(t) = 0.9$. (c) $O_K(t) = 0.9, O_{HK}(t) = 0.9$.

Figure 4.13: BF S_{KH} and the OD plotted along three pairs of trajectories of Σ_K and Σ_H generated using constant voltage (input) signals. S_{KH} upper bounds the OD and decays along the trajectories.

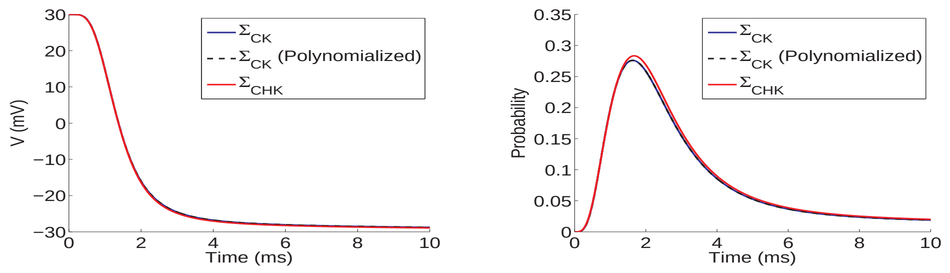
a factor of 9, in Fig. 4.13(b), S_C bounds the OD and decays along the trajectories.

CCMs Σ_{CK} and Σ_{CHK} are autonomous dynamical systems and do not receive any external inputs. To visualize the composite BF S_2, we simulated Σ_{CK} and Σ_{CHK} using the Euler method. Fig. 4.14 plots the trajectories obtained from these simulations. The corresponding conductance traces of Fig. 4.14(a) and the voltage traces of Fig. 4.14(b) empirically validate that the composed models are approximately equivalent as predicted by Theorem 4.3.3. BF S_2 along this pair of trajectories is plotted in Fig. 4.15(a). The value of S_2 is dominated by the value of S_C, as it bounds the squared difference of voltages and is much larger than S_{KH}, which bounds differences in probabilities. This is reasonable as voltage is the primary entity of interest when analyzing excitable cells. One could scale subsystem Σ_C such that its output lies in $[0, 1]$ and is thus comparable to the outputs of Σ_K and Σ_{HK}.

To test extreme cases, we simulated Σ_{CK} and Σ_{CHK} by initializing Σ_C at different values. Fig. 4.15(b) shows the OD plotted along pairs of trajectories, where Σ_C in Σ_{CK} starts at $V_K(0) = -25\ mV$ and in Σ_{CHK} at $V_{HK}(0) = 25\ mV$. Fig. 4.15(c) plots the other extreme, where the copy of Σ_C in Σ_{CK} starts at 25 mV and the copy in Σ_{CHK} is initialized to $-25\ mV$. S_2 bounds the OD in all these cases and decays along the trajectories.

4.9 Discussion

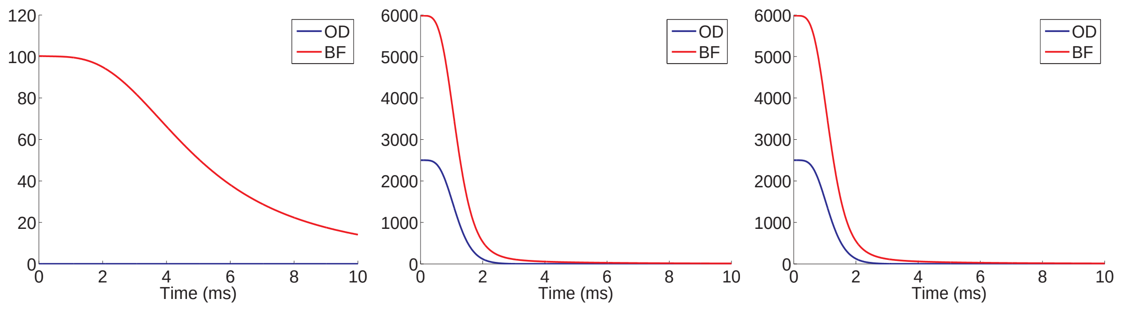
In Sections 4.5 and 4.7, we presented two SOS formulations for computing BFs. These BFs were then used to derive compositionality results for the Σ_I and σ_K components of the IMW model. In this section, we compare and contrast the two



(a) $O_K(t)$ and $O_{HK}(t)$ of Σ_{CK} and Σ_{CHK} respectively.

(b) $V_K(t)$ and $V_{HK}(t)$ of Σ_{CK} and Σ_{CHK} respectively.

Figure 4.14: Simulations of Σ_{CK} and Σ_{CHK}: when Σ_K is replaced by Σ_{HK}, feedback composition tends to accumulate error incurred due to the abstract component. S_2 proves that these errors remain bounded due to the approximate equivalence of Σ_K and Σ_{HK}, established by BF S_{KH}, and the ability of Σ_C to tolerate deviations in the conductance inputs, established by BF S_C.



(a) $V_K(0) = -25\ mV$, $V_{HK}(0) = -25\ mV$.

(b) $V_K(0) = -25\ mV$, $V_{HK}(0) = 25\ mV$.

(c) $V_K(0) = 25\ mV$ $V_{HK}(0) = -25\ mV$.

Figure 4.15: S_2 and OD along pairs of trajectories of Σ_{CK} and Σ_{CHK}. Subfig. (a) plots OD and S_2 along trajectories where the Σ_C component is initialized to -25 mV. Subfigs. (b) and (c) plot OD and S_2 along trajectories where Σ_C is initialized at different voltages. Value of OD is dominated by difference of outputs of the two copies of Σ_C, which is in mV. Similarly, value of S_2 is dominated by S_C, which bounds differences in voltages as opposed to S_{KH}, which bounds differences in probabilities.

SOS formulations and their implementation issues in SOSTOOLS. Table 4.5 presents a summary of our discussion. We elaborate on each of the points below.

SOSP 1, which is defined in Eqs.(4.6) - (4.8), computes the BF, the decay rate λ and the input-gain parameter γ as per Defn. 4.3.1. On the other hand, SOSP 2 involves two additional unknowns: $\sigma_1(,)$ and $\sigma_2(,)$.

Polynomialization of vector fields is not needed for SOSP 1. The rate matrices

	SOSP 1	**SOSP 2**
Definitions	Eqs.(4.6)-(4.8).	Eqs.(4.11)-(4.13).
Unknowns	S, λ, and γ.	S, λ, γ, σ_1, and σ_2.
Polynomialization	Not needed, input-space discretization makes the rate-matrix constant.	Required, exponential rate functions must be converted to polynomial forms before implementing in SOSTOOLS.
Use of the *sparse* option in SOSTOOLS	MATLAB's default convex hull program *convhulln* is sufficient.	The CDD package has to be used as per Section 3.4.3 of the SOSTOOLS user guide [65].
Input space	The BF is valid only on the quantized input space given by $\mathcal{U}$.	The BF is valid over the entire input space.
Error bounds	The BFs provide much tighter bounds on the OD.	The BFs are more conservative and provide relatively weaker bounds on the OD.

Table 4.5: A comparison of SOSP1 and SOSP 2.

$A_I(V_I)$, $A_K(V_K)$, $A_{IH}(V_{IH})$, $A_{KH}(V_{KH})$ become constant matrices due to input-space quantization. On the other hand, SOSP 2 requires the entries of the matrices to be polynomial functions of the respective inputs. Therefore the rate functions, which constitute the entries of the matrices, need to be approximated using polynomial functions as per Section 4.7.2.

Scalability is an important issue for computing BFs. Specifically, declaring the inequalities in SOSTOOLS is a memory-intensive operation. SOSTOOLS provides the *sparse* option to the *sosineq* function. When this option is used, SOSTOOLS constructs a sparse representation of the inequality using convex hull operations. MATLAB's convex hull function, *convhulln*, is called internally by default. SOSP 1, which works for a discretized input space, can exploit this feature to compute the BFs. SOSP 2, which involves a higher number of symbolic variables, does not scale well with the default *convhulln*-based *sparse* option. As a workaround, a specialized convex hull program, CDD, must be used, as per Section 3.4.3 of the SOSTOOLS user guide [65].

Finally, we observed that SOSP 2 results in BFs that provide a relatively more conservative bounds on the OD, as compared to the BFs provided by SOSP 1. Figs. show that the BF S_{IH}, which was computed using SOSP 1, and the OD have the same order of magnitude. On the other hand, Figs show that the BF

S_{KH}, which was computed using SOSP 2, is approximately two orders of magnitude greater than OD.

Despite its relatively weaker bound, SOSP 2 satisfies Eq. 4.3 exhaustively over the input space. On the other hand, the BFs computed using SOSP 1 satisfy Eq. 4.3 only on the discretized input space.

In Chapter 7, as a potential direction for future research, we propose an algorithm that combines SOSP 1 and SOSP 2 to compute BFs that provide meaningful bounds on the OD in addition to a proof of stability.

Chapter 5

Curvature Estimation of Cardiac Excitation Waves

Determining the physiological conditions underlying a cardiac arrhythmia is a grand challenge, whose resolution may lead to innovative treatment strategies. An important component of this quest is the mathematical modeling, analysis and simulation of cardiac-cell networks as seen in the previous chapters, also see [11, 3, 29]. In the context of reaction-diffusion-based cardiac models, the above challenge can be reformulated as follows: *For what parameter ranges does a DEM network accurately reproduce the arrhythmia?* In this chapter, we will present a simulation-based approach to this problem.

We present the *Spiral Classification Algorithm* (SCA) [61, 34], a fast and accurate algorithm for classifying electrical spiral waves and their associated breakup in cardiac tissues. The classification performed by SCA is an essential component of the detection and analysis of various cardiac arrhythmic disorders, including ventricular tachycardia and fibrillation. Given a digitized frame of a propagating wave, the SCA constructs a highly accurate representation of the front and the back of the wave, piecewise interpolates this representation with cubic splines, and subjects the result to an accurate curvature analysis. This analysis is more comprehensive than methods based on spiral-tip tracking, as it considers the entire wave front and back. To increase the smoothness of the resulting symbolic representation, the SCA uses weighted overlapping of adjacent segments which increases the smoothness at join points.

SCA has been applied to a number of representative types of spiral waves, and, for each type, a distinct curvature evolution in time (signature) has been identified. Distinct signatures have also been identified for spiral breakup. These results represent a significant first step in automatically determining parameter ranges for which a computational cardiac-cell network accurately reproduces a particular kind of cardiac arrhythmia, such as ventricular fibrillation.

5.1 Motivation

The past two decades have witnessed the development of increasingly sophisticated DEMs [20], ranging from 4 to 87 state variables [8, 52, 66, 72, 36, 23]. The increase in the number of variables reflects the technological advances in capturing the intrinsic ionic mechanisms more accurately. Unfortunately, the increase in the number of state variables inevitably leads to an increase in simulation time. In particular, simulation of the 67-variable DEM is so slow that its authors only simulated it in a single cell and even provided initial conditions corresponding to the steady state at different pacing rates to reduce the computation time for assessing the dynamics associated with those rates. In [3], the authors present CUDA-based GPU implementations for many of the above-cited DEMs, on both Tesla and Fermi cards, with a dramatic reduction in simulation times. This allows us to perform, for the first time on a desktop computer, a 2D (surface) simulation of the 67-variable DEM.

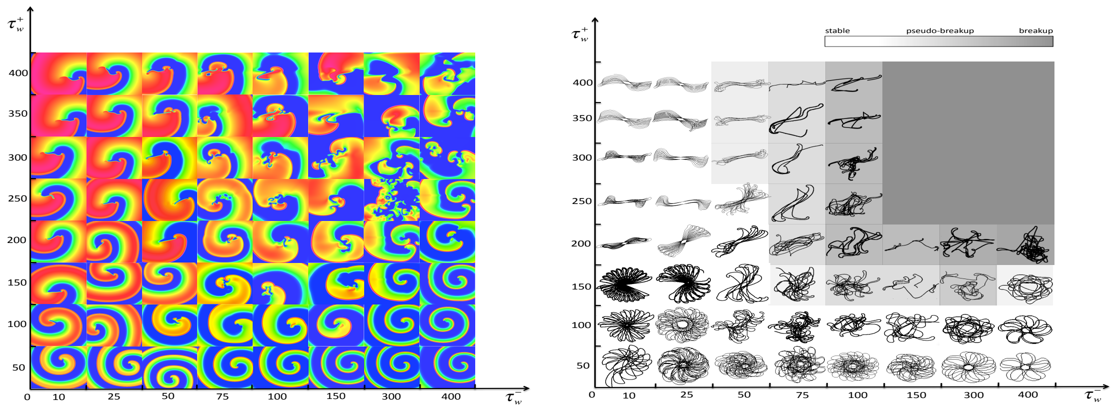


Figure 5.1: Simulation-based estimation of parameters τ_w^-/τ_w^+: (a) Wave forms. (b) Tip movement and regions of fibrillation.

One of the smallest cardiac models that can accurately replicate, in simulations, the macroscopic behavior of cardiac cells is the 4-variable Minimal Model (MM) of Fenton and Cherry [8]. The Graphics Processing Unit (GPU) implementation of the MM is so fast, that it allows the real-time simulation of a 500×500 grid of cells: one second of MM simulation time is approximately equal to one second of real-time.

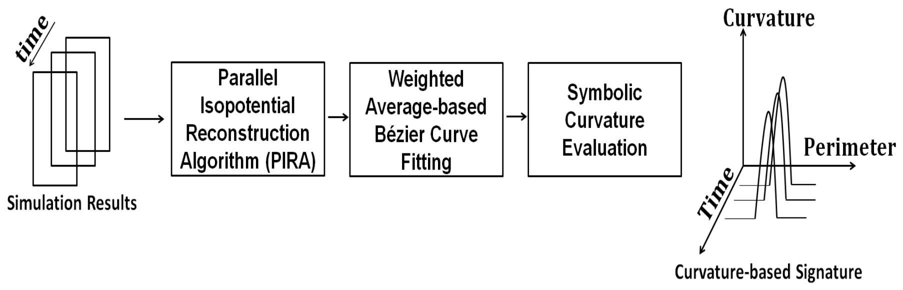

time
Simulation Results
Parallel Isopotential Reconstruction Algorithm (PIRA)
Weighted Average-based Bézier Curve Fitting
Symbolic Curvature Evaluation
Curvature
Perimeter
Time
Curvature-based Signature

Figure 5.2: SCA: Block diagram for curvature estimation.

This increase in the simulation speed enables systematic exploration [70] of its associated parameter space. For example, in a student workshop on Computational Modeling and Analysis of Complex Systems [4], undergraduate students were asked to repeatedly simulate the MM so that the ranges of two parameters, τ_w^- and τ_w^+, leading to fibrillation could be determined, see Fig. 5.1. Specifically, given a fixed spiral-initiation protocol, the students were asked to simulate the MM on each node of a 2D grid of parameter values, capture the generated waves, and track the tip movement of the spirals.

This "Crowd Sourcing" approach provided very encouraging initial results. Crowd sourcing, however, is unlikely to scale up to the exploration of large parameter spaces. Such exploration minimally requires: 1) A principled way of partitioning the parameter space; and 2) a fast and accurate algorithm for classifying spiral waves and their breakup. The Spiral Classification Algorithm (SCA) serves the later purpose by automatically classifying spiral waves and their associated breakup. See Fig. 5.2 for a block diagram.

5.2 Isopotential Reconstruction

The simulation data obtained by executing an $N \times N$ grid of DEMs, or the experimental data obtained by optically mapping a cardiac tissue with resolution $N \times N$, is a sequence of *digital frames* F_t of dimension $N \times N$. Each frame F_t is the snapshot at time t, with resolution $N \times N$, of the transmembrane *electrical potential* V_t, of the cardiac tissue.

The *wave-fronts (wave-backs)* of V_t are defined as the regions of V_t where each cell is in the activation (recovery) phase; that is, their electrical potential $V_t(x, y)$ equals -30mV, and their derivative $dV(x, y)/dt$ is positive (negative). The two fronts define together the *isopotentials* I_{-30} of V_t. These are curves of constant voltage in V_t, that never intersect each other. For simplicity, we will drop the subscript t

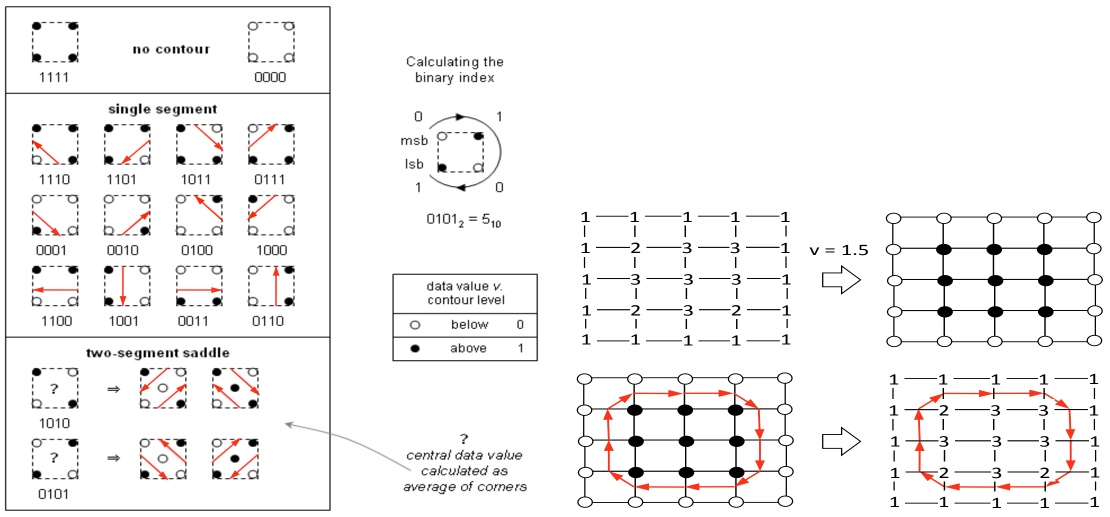


Figure 5.3: Marching squares.

occurring in F_t and V_t.

For any potential value v, the isopotentials I_v of V are smooth curves, whereas the isopotentials I_v of F are not. This is a consequence of the $N \times N$ resolution. The *isopotential reconstruction problem* is therefore defined as follows: *Given a frame F, and a voltage value v, reconstruct the smooth I_v, isopotentials of V*. As we will see in the following sections, smoothness is essential for an accurate *curvature analysis* of the wave-fronts and wave-backs.

The *Isopotentials Reconstruction Algorithm* is a key component of the SCA, as it consumes most of the memory and time resources of the SCA. Hence, IRA has to be designed very carefully, and run on an appropriate hardware platform, to make the SCA a success. Today, the the NVIDIA GPUs Fermi or Tesla video cards allow one to run a computationally intensive tasks on desktop machines. We take advantage of these cards to implement the Compute Unified Device Architecture (CUDA)-based *Parallel Isopotential Reconstruction Algorithm (PIRA)*.

PIRA belongs to a new CUDA-based class of algorithms that minimize the amount of synchronization; they require reconstruction of global information from a frame F. To achieve this goal, PIRA divides its work into: Parallel Marching Squares (PMS), a fully parallel, local-information-computation procedure; Parallel Isopotential Extraction (PIE), a hierarchically parallel, global-information-computation procedure; and Selection and Output Generation (SOG), an optional, sequential to random-access, global-information-computation procedure.

Parallel marching squares (PMS)

The fully parallel, local-information-computation procedure PMS uses an adaptation of the *marching squares* algorithm. Given a frame F, PMS considers in parallel, that is in a different thread of a acCUDA block, each 2×2 square s, of an adjacency-based $(N-1)\times(N-1)$ partition of F. Each thread locally and accurately computes the intersection points of zero, one or two lines with s. The number of lines and their crossing pattern with s depends on the values in the corners of s (type of s).

To determine the type of each square s of F, frame F is first subjected, in parallel, to the test $F_{i,j} \geq v$, for each index (i, j). The result of the test is stored in a boolean matrix B. The boolean value of the corners of s in B, traversed say in clockwise order, determine 16 possible intersection cases. These cases are shown in Figure 5.3. In summary: 1) If all corners of s in B have the same value, there is no line crossing at all. 2) Otherwise, if diagonal corners have the same value, there are two ambiguous line crossings. Ambiguity is removed by averaging the value in all corners in s and comparing it with v. 3) Otherwise precisely one line is crossing s. Our adaptation also associates each with line, a direction by requiring that the 0 corners of square s only occur to its left.

An isoline intersects s only between two corners that have opposite boolean value in B. In other words, one corner has a value less than v and the other has a value which is greater than v. The intersection points, which also represent the starting and the ending points of a directed, one-segment-long polyline, are computed via linear interpolation. For example, suppose that the corners of s at position (i, j) in B, result in bitvector 0111. Then, the line segment crossing s has the points $((x_0, y_0), (x_1, y_1))$ defined as follows:

$$\begin{array}{ll} x_0 = j, & y_0 = i + (v - F_{i,j}) \;/\; (F_{i+1,j} - F_{i,j}) \\ y_1 = i, & x_1 = j + (v - F_{i,j}) \;/\; (F_{i,j+1} - F_{i,j}) \end{array}$$

This information is stored in the corresponding one segment-polyline at position (i,j), in an $N \times N$ matrix S of squares. The starting and ending polylines of the open polylines list are initialized to the location of this polyline. If there are two polylines, the first one is the first in the list and the second one is the second. The list of closed polylines is initialized to null. The number of polyline segments and the number of open/closed polylines is also appropriately initialized.

```
struct Point    // typed points
{
   int    type; // 0 = row crossing, 1 column crossing
   float  x,y;  // x and y coordinates
   float  y;    // y coordinate
}

struct PolyLine                    // directed polylines
{
  int            type;             // 17 types
  int          number;          // number of segments
  Point        start, end;      // start and end points
  PolyLine *   next;            // ptr to next polyline
}
```

To speedup the interpolation process, the case analysis of the square type is efficiently pre-stored in a lookup table T, allocated in the constant memory of the GPU card.

```
struct Square                   // Data Structure for Squares
{
  int          num_of_opl;      // number of open polylines
  int          num_of_cpl;      // number of closed polylines
  PolyLine *   opl_start;       // pointer to first open polyline
  PolyLine *   opl_end;         // ptr to the last open polyline
  PolyLine *   cpl_start;       // ptr to the first closed polyline
  PolyLine *   cpl_end;         // ptr to the last closed polyline
  PolyLine     poly_lines[2];   // storage allocated on leaf level
}
```

For each square type t and for each one-segment polyline p, the table consists of the following entries: Two line coefficients and four coordinate displacements. For the above example, where $t = 7$, $p = 0$, table $T_{t,p}$ contains:

$$\begin{aligned} T_{t,p} = \{ & \{a = 0, b = 0, i_1 = 0, j_1 = 0, i_0 = 0, j_0 = 0\}, \\ & \{a = 1, b = 0, i_1 = 1, j_1 = 0, i_0 = 0, j_0 = 0\}, \\ & \{a = 1, b = 0, i_1 = 0, j_1 = 1, i_0 = 0, j_0 = 0\}, \\ & \{a = 0\ b = 0, i_1 = 0, j_1 = 0, i_0 = 0, j_0 = 0\}\} \end{aligned}$$

Denote $T_{t,0}$ and $T_{t,1}$ with the corresponding field names, indexed by subscripts 0 and 1. Then the one-segment-polyline starting coordinate can be computed uniformly, as follows:

$$\begin{aligned} x_0 &= j + b_0 + a_0 (v - F_{i+i_{00},j+j_{00}}) / \ , (F_{i+i_{01},j+j_{01}} - F_{i+i_{00},j+j_{00}}) \\ y_0 &= i + b_1 + a_1 (v - F_{i+i_{10},j+j_{10}}) / \ , (F_{i+i_{11},j+j_{11}} - F_{i+i_{10},j+j_{10}}) \end{aligned}$$

Parallel isoline extraction (PIE)

Using PMS one can readily plot the isolines of V. However, there is no global information available yet, despite the fact that we know the isoline segments, their orientation, and their linking. However, we do not know: 1) Which segments to the particular isolines? 2) How many isolines are there in V? 3) How many segments do they contain? 4) What are their starting and ending points.

The public *contour* function of Octave uses a sequential, recursive algorithm to obtain global information. This works as follows: 1) Traverse a matrix of interpolated segments, row by row and column by column, and pick the first unmarked

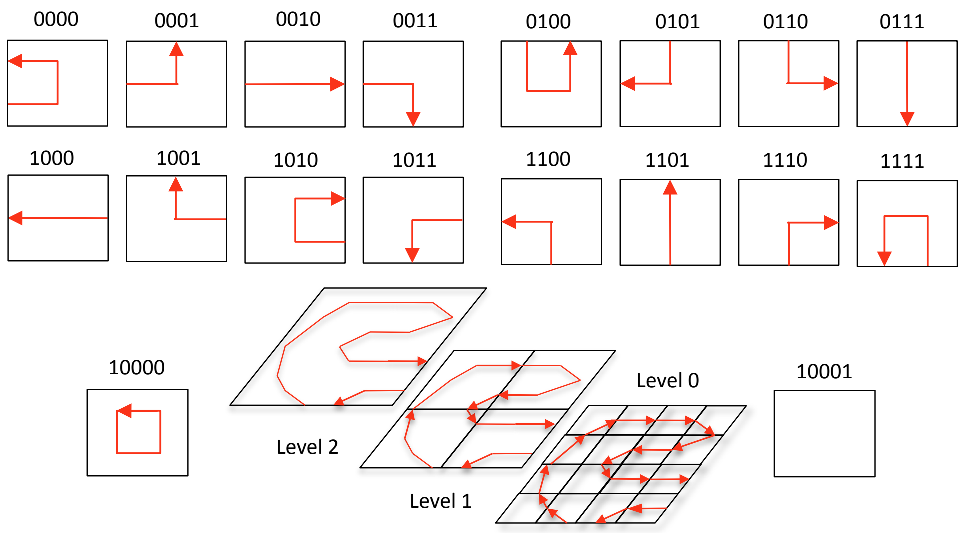


Figure 5.4: Isopotential extraction.

one. 2) Then recursively follow matching segments in a meaningful way, until the border or the starting point is reached again (no direction is readily available as in our PMS). During this process, dump all points traversed in a dynamic array, and mark all the segments as visited. 3) Once an entire isoline is found, search for the next unmarked segment, until the last row and column is reached.

Recursion and the sequential traversal of the entire matrix of segments is however prohibitive for a fast curvature analysis. The *isoline* function available at [54] is sequential, but not recursive. It: uses the marching squares algorithm to compute all (disoriented) segments and dump them, in no particular order, in a dynamically allocated array. Then, it extracts the first unmarked segment, and repeatedly searches the entire array for an unmarked, matching continuation. Its time complexity is therefore comparable to the recursive algorithm.

To obtain a fast algorithm, we take advantage of the GPU cards. However this imposes several important restrictions on PIE: 1) It cannot be recursive. 2) It cannot dynamically allocate memory. 3) It can only contain for-loops with known upper bounds. As shown in Figure 5.4, the main idea of PIE is to organize the squares S passed by PMS in a quad-tree fashion, such that, at every level in the tree hierarchy, sibling nodes (and their immediate children) are processed in parallel, by a different CUDA thread. Hence, global information is computed sequentially in $log_4 N$ steps.

During this process, a parent either collects or pastes together the polylines of its children, if they have matching start and end points. A particular problem during this process is dynamic memory allocation. In the CUDA core, such allocation is not possible. Moreover, there is no way to predict in advance, without gross overestimation, the amount of memory needed in each node, to store the list of polylines it has identified so far, and the polylines themselves.

Our solution is to reuse the memory already allocated in the leaf squares for their two one-segment-long polylines during the upward sweep of the quad-tree. Each time, when information is collected from the 4 children, the result is stored in child 0. This information is later on passed upward in the hierarchy. Therefore, child 0's information gets lost.

Whenever the parent process matches the ending point of polyline p of child c with the starting point of polyline q of child d, it updates the starting and ending points of polyline pq in p, and removes q from the polyline list of child d. It then continues from a copy of q to find the next match. The total number of matches, is bound by the number of polylines in all children, and this is used in a find-next-match for-loop. This loop is the equivalent of the recursive match-next-segment search in *contour* or *isoline*, but it is limited to the *polylines* of the parent's children. This dramatically decreases the time complexity of the recursive search.

Whenever, the parent process collects the polyline lists of the children, it appends these lists to the lists of child 0. For this purpose, the Square structure has the start/end fields of the list of open/closed polylines, and each polyline has a next polyline field. The number of polylines, and the number of segments, are also updated accordingly.

In order to efficiently match the end point of a polyline p of a child c with the starting point of a polyline q, it is important to know what sibling of c might have such a q. For this purpose, we classify the polylines according to their starting and ending faces in c, as shown in Figure 5.4. This leads to a classification of polylines that is similar to the one for squares. In contrast to squares however, polylines may start and end on the same face or they may be closed (that is, they are completely contained in c). The second case does not require further processing. Similarly, if a polyline of square c starts and ends on a face that is not adjacent to any of the c's siblings, no processing is required at this level either. To enable this kind of analysis, a type field is added to the polyline data structure, too.

The downside of reusing the squares while collecting and propagating information is that the linking information of one-segment-long polylines is lost. Consequently, one has to store this information in a different place. For simplicity and speedup reasons, we allocate two hash tables X and Y of size $N \times N$: the first is indexed by the integer value of horizontal intersection points, the second by the integer value of vertical intersection points. Each entry of X and Y stores the destination point and a bit classifying it as a horizontal or vertical intersection. This allows us to determine whether to choose the X or the Y hash table next. In general, unless the isolines are fractals, the number of segments is orders of magnitude smaller than $N \times N$. Hence, more information acquired about the expected kind of isolines may improve the size allocation, by choosing $M << N$ for the tables.

Selection and Output Generation (SOG) procedure
The optional SOG procedure selects the isolines of interest according to some given criterion, for example, the longest isoline, and stores the points of these isolines in an array, sorted by their traversal order. The size of the isolines their starting point, and their ending points are available at the root of the quad-tree. This information is used to dynamically allocate the corresponding arrays outside of the CUDA core. The X and Y hash tables are then used to traverse the associated isoline and dump the points traversed in the array. This process transforms the local sequential-linking information, into a global, random-access information, where entry $i+1$ is known to be the successor point of entry i.

We reconstructed the isolines of 10 000 frames, both with PIRA and the *contour* function of MATLAB. The first took 1.65 seconds while the second took 720 seconds. Hence, PIRA had a 444.44-fold speedup compared to *contour*. The platform was equipped with Intel Core i7-930 2.8GHz LGA 1366 130W Quad-Core Desktop Processor with OCZ Gold 12GB (6 x 2GB) 240-Pin DDR3 SDRAM DDR3 1600 (PC3 12800) Low Voltage Desktop Memory and a GPU Tesla C1060 with 240 SP cores divided equally among 30 SMs, and 4GB of DRAM. The SP core clock speed was 1.29 GHz and the maximum bandwidth of memory access was 102 GB/sec. The rest of the SCA has not been parallelized yet. However, this can easily accomplished, as described in the following sections.

5.3 Curvature Estimation

Propagation patterns of cardiac waves (isopotentials produced by SOG in the previous section) characterize cardiac arrhythmia. Obstacles or deformities affect the nominal linear motion of the waves resulting in anomalous propagation. Reentry is one such anomalous pattern in which cardiac waves assume spiral shapes, many of which are precursors to more dangerous arrhythmias. When these spiral waves break up, fibrillation, a possibly fatal arrhythmia follows. In [49], authors provide experimental evidence of atrial fibrillation being characterized by reentrant cardiac waves breaking-up into chaotic patterns.

Local geometric features like curvature can be critical in affecting wave propagation and breakups. In [18], the authors establish the relation between the curvature of the isopotential and its propagation velocity. If θ_0 is the steady state velocity of a linear isopotential, then the velocity of a curved isopotential is given by $\theta = \theta_0 - \frac{D}{r}$, where r is the local radius of curvature and D is a coefficient determined by the passive properties of the cardiac medium. Thus, any part of the isopotential that has high positive curvature slows down and eventually breaks off. Also, the existing arrhythmia can be classified accurately based on the curvature of the underlying cardiac waves. Thus, an efficient method of measuring the evolving curvature would enable expressing emergent tissue-level behaviors. This would further help in predicting and identifying arrhythmias.

In reality, the cardiac tissue is strongly anisotropic, with waves propagating approximately three times faster along the direction of the heart muscle fibers (longitudinal) than in the transverse direction. This anisotropy can lead to problems in measuring the true wave curvature, as noted in [75]. In practice, however, an anisotropic medium can be rescaled according to the known ratio of diffusion coefficients along and across the fibers to recover an isotropic medium. This rescaling can be used to extend our algorithm to anisotropic tissue behaviors. Section 4 of [75] discusses other methods that can be used to adjust the measurements so that they can be compared with isotropic theory. We will therefore restrict our focus to isotropic media in the following discussion.

Any method that estimates the curvature of cardiac isopotentials must satisfy the following requirements: not only must it be highly accurate but the method must provide curvature values continuously along the perimeter of the isopotential. In other words, the method must be independent of the spatial resolution at which the isopotential was estimated or the grid on which the cardiac model was numerically integrated.

The workflow of our curvature estimation technique is shown in Fig. 5.2. The isopotential obtained in the previous section is a series of points on $\mathbb{R}^2$. Starting from this, our method of obtaining a smooth curvature estimate involves the following steps:

1. **Step 0 - Preprocessing:** Divides the isopotential obtained by SOG into overlapping strips of constant length.
2. **Step 1 - Bézier curve fitting:** Fits each strip with a third degree Bézier curve, thus obtaining a piecewise degree-3 polynomial fit of overlapping Bézier curves.
3. **Step 2 - Estimate curvature using symbolic analysis:** Curvature is calculated along the Bézier curves using symbolic calculations.
4. **Step 3 - Obtain overall curvature estimate:** Obtains a smooth estimate of curvature along the entire isopotential.

First, we divide the isopotential I_v, obtained at potential v, into overlapping segments which we refer to as *strips*. The length of each strip is controlled by the parameter STRIP_LENGTH (SL). The percentage overlap among them is controlled by the parameter $OVERLAP$. This is the initial pre-processing done before the polynomial fitting. The following sections describe each of the remaining steps in detail.

5.4 Piecewise Bézier Fitting

Most of the formalisms of curvature in $\mathbb{R}^2$ involve second-order derivative terms. This motivates a degree-3 polynomial approximation to the isopotential. We fit the isopotential with overlapping cubic Bézier curves in a piecewise manner, thus satisfying the above-mentioned requirements of curvature estimation.

Let the j^{th} strip of the isopotential I_v be denoted as $I_{v,j}$. We obtain one Bézier curve that approximates this strip up to a certain degree of L2 error. A Bézier curve approximation for $I_{v,j}$ would have the following parametric form, where $t \in [0, 1]$:

$$X_j(t) = (1-t)^3 P_j^0 + 3t(1-t)^2 P_j^1 + 3t^2(1-t)P_j^2 + t^3 P_j^3 \tag{5.1}$$

$$Y_j(t) = (1-t)^3 Q_j^0 + 3t(1-t)^2 Q_j^1 + 3t^2(1-t)Q_j^2 + t^3 Q_j^3 \tag{5.2}$$

Here P_j^0 - P_j^3 and Q_j^0 - Q_j^3 are the control points of the Bézier curves, which approximate the isopotential strip along x- and y-axes respectively. To sample from this curve, we evaluate the above-mentioned expressions over the interval $t \in [0, 1]$. *Terminal* control points P_j^0, P_j^3, Q_j^0 and Q_j^3 coincide with the start and end points of the strip respectively. The fitting procedure, adapted from [48] and [68], optimizes the intermediate control points P_j^1, P_j^2, Q_j^1 and Q_j^2 to minimize the least squared error. We assume uniform parametrization of t in $[0, 1]$ for each segment. The error functions for fitting $I_{v,j}$ are

$$E_x = \sum_{i=1}^{SL} [x_i - X_j(t_i)]^2, E_y = \sum_{i=1}^{SL} [y_i - Y_j(t_i)]^2,$$

where (x_i, y_i) denotes the i^{th} point on the isopotential strip. On replacing Eq. (1) and (2) in the above error functions respectively, we obtain:

$$\sum_{i=1}^{SL} \quad [x_i - (1-t_i)^3 P_j^0 - 3t(1-t_i)^2 P_j^1 - 3t_i^2(1-t_i)P_j^2 - t_i^3 P_j^3]^2,$$

$$\sum_{i=1}^{SL} \quad [y_i - (1-t_i)^3 Q_j^0 - 3t_i(1-t_i)^2 Q_j^1 - 3t_i^2(1-t_i)Q_j^2 - t_i^3 Q_j^3]^2$$

as expressions for E_x and E_y, respectively. We show the calculations only for E_x. For E_y, the expressions follow from those for E_x. Control points P_j^1 and P_j^2 can be obtained at the minimum value of E_x by:

$$\frac{\partial E_x}{\partial P_j^1} = 0, \frac{\partial E_x}{\partial P_j^2} = 0.$$

Solving the above two equations we obtain the following expressions for P_j^1 and P_j^2:

$$P_j^1 = \frac{\alpha_2^j\beta_1^j - \alpha_3^j\beta_2^j}{\alpha_1^j\alpha_2^j - {\alpha_3^j}^2}, P_j^2 = \frac{\alpha_1^j\beta_2^j - \alpha_3^j\beta_1^j}{\alpha_1^j\alpha_2^j - \alpha_3^{2j}},$$

where, $\alpha_1, \alpha_2, \alpha_3, \beta_1$ and β_2 for the j^{th} segment are given by:

$$\alpha_1 = 9\sum_{i=1}^{SL}[t_i^2(1-t_i)^4],$$

$$\alpha_2 = 9\sum_{i=1}^{SL}[t_i^4(1-t_i)^2], \alpha_3 = 9\sum_{i=1}^{SL}[t_i^3(1-t_i)^3],$$

$$\beta_1 = 3\sum_{i=1}^{SL}[t_i(x_i - (1-t_i)^3P_0 - t_i^3P_3)(1-t_i)^2],$$

$$\beta_2 = 3\sum_{i=1}^{SL}[t_i^2(x_i - (1-t_i)^3P_0 - t_i^3P_3)(1-t_i)].$$

The fitting procedure described above is sequential in nature. However, for all the strips, one can run this procedure in parallel, by using a different CUDA-thread on the GPU cards. Although we have not done this yet, this is a simple adaptation of this algorithm. To make the curvature estimates smooth, we fit the curves on overlapping segments, as explained in the next section.

5.5 Weighted Average-based Improvement

After each strip is fit with cubic Bézier curves as explained above, we improve the smoothness of the overall fit. A pure piecewise fitting approach would create discontinuities in the derivative of the isopotential-approximation at the points where two curves meet. We ensure that derivatives up to second order are well defined everywhere on the isoline. To do this, fitting is performed on overlapping strips. Parameter OVERLAP determines the percentage overlap between adjacent strips of the isopotential.

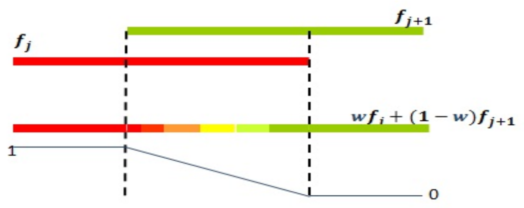


Figure 5.5: Weighted average-based fit smoothing and curvature estimation.

Consider a part of the isopotential that has two adjacent overlapping strips with indices j and $j+1$. To define the Bézier curve fit for the part of the isoline covered by the two strips, we use a weighted average-based method. Suppose the curves describing the two strips are f_j and f_{j+1}. Then the fit for the region covered by the two strips is given by $w_j f_j(t_j) + w_{j+1} f_{j+1}(t_{j+1})$, where $w_{j+1} + w_j = 1$. In essence, the influence of the adjacent curves on the fit is gradually varied in the region of overlap. In the region where there is no overlap, the fit is completely described by the only Bézier curve corresponding to that strip. As one enters the region of overlap, the fit is a weighted average of the two Bézier curves corresponding to adjacent strips. The weights are varied linearly in our scheme, as shown in Fig. 5.5. The same weighted-average smoothing is performed for the derivatives and the curvature values.

Each Bézier curve is a third degree polynomial. In the overlapping region, the fit is a weighted average of two cubic polynomial functions. Thus even in the overlapping region the fit is C^2 smooth, rendering it amenable to curvature estimation (described in the next section). The smoothing described above can also be performed in parallel. The weighted-average based scheme is different from [46], which uses cubic splines to fit the isopotentials.

5.6 Curvature Estimation using MATLAB Symbolic Computation

We use symbolic computations in MATLAB to evaluate curvature along the isopotential. This constitutes step 2 of our method listed above. The fitting procedure described above provides a set of smooth Bézier curves that describe the isopotentials. As these are closed form expressions, they can be manipulated symbolically to calculate the magnitude of the curvature along the isopotentials. MATLAB's symbolic toolbox provides the facility to declare symbolic variables, construct functions out of them and operate on those functions. Once the operations yield the expressions-of-interest, they can be evaluated at arbitrary spatial resolution by suitably specifying the interval for the symbolic variables.

In our case, we obtain the functions $X_j(t)$ and $Y_j(t)$ for each strip. The absolute curvature of the strip of the isopotential is derived using elementary calculus:

$$\kappa_j(t) = \frac{|r'_j(t) \times r''_j(t)|}{|r'_j(t)|^3} \tag{5.3}$$

where $r_j(t) = [X_j(t), Y_j(t)]$ is the position vector described by the Bézier curve. The grid size for all the cardiac simulations is in the order of microns, thus, the unit of the curvature measured is $(\mu m)^{-1}$. The important point to note is that $X_j(t)$, $Y_j(t)$ and thus $r_j(t)$ are managed as symbolic expressions which are functions of the symbolic variable t. Thus $\kappa_j(t)$ is obtained in closed form as an expression in t. Symbolic operations on $r_j(t)$ are performed using MATLAB's symbolic math toolbox [55].

After obtaining closed form expressions for curvature, their continuity ensures that we can evaluate them at any resolution of the parameter t. This translates to obtaining a continuous estimate of curvature along the perimeter of the isopotential.

Step 3 of our method evaluates these curvature functions along the isopotential using the weighted approach described in the previous section. Currently we maintain uniform resolution for t along all the strips. Adapting this to the shape of the isoline is part of our future work. In particular, the information stored in the quad-tree of PIE, for example the filling factor of the area associated to its nodes might facilitate a fast and accurate breakup of the isoline in isoline-strips, improving on the idea in [46].

An alternative approach to curvature estimation involves pre-computing the generic form of the curvature function for a Bézier curve. Consider a curve $r(t) = [X(t), Y(t)]$ where $X(t)$ and $Y(t)$ are described by Bézier curves in Eq. (5.1) and (5.2) respectively. Also let $a_x = -P_j^0 + 3P_j^1 - 3P_j^2 + P_j^3$, $b_x = 3P_j^0 - 6P_j^1 + 3P_j^2$, $c_x = -3P_j^0 + 3P_j^1$, $d_x = P_j^0$ and a_y, b_y, c_y, d_y be defined symmetrically. Expressions $a_x \dots d_x$ and $a_y \dots d_y$ define the curves $X(t)$ and $Y(t)$ in the nominal polynomial form, for example, $X(t) = a_x t^3 + b_x t^2 + c_x t + d_x$. Now

$$\begin{aligned}|r'(t) \times r''(t)| &= X'(t).Y''(t) - Y'(t).X''(t) \\ &= [12(b_x a_y - a_x b_y) + 6(a_x b_y - b_x a_y)]t^2 \\ &+ [6(c_x a_y - a_x c_y)]t + 2[b_y c_x - b_x c_y]\end{aligned}$$

In the same way, we can calculate

$$\begin{aligned}|r'(t)| &= [r'(t).r'(t)]^{1/2} \\ &= [(3a_x t^2 + 2b_x t + c_x)^2 + (3a_y t^2 + 2b_y t + c_y)]^{1/2}\end{aligned}$$

Using the above expressions, the curvature of each strip can be calculated. This method is also amenable to parallel computation as it does not need the symbolic computational toolbox of MATLAB. We have implemented both techniques. The curvature for the whole isoline is calculated using the weighted average-based method as explained above.

The runtimes for the curve fitting and curvature calculation routines depend on the length of the isopotential. The extraction process checks each grid square for a possible point of the isopotential. The number of edges on a $n \times n$ grid can be calculated by solving the following recurrence:
$E(n) = E(n-2) + 4n + 4(n-3) + 8$
$E(1) = 4$, $E(2) = 8$
The solution is given by

$$E(n) = -2(-1)^n + 2n(n+1) - 2$$

Hence the maximum length of the isoline is $O(n^2)$, which bounds the number of operations in the fitting and curvature routines to $O(n^2)$.

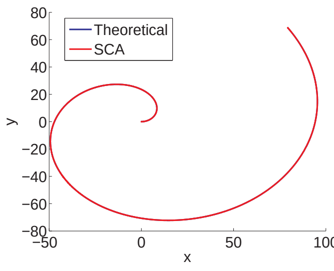
5.7 Curvature Estimation Accuracy

The curvature estimation technique developed in Sections 5.3-5.6 was implemented on data generated by plotting standard curves, for which curvature can be calculated in closed-form. A summary is given in Table 5.1.

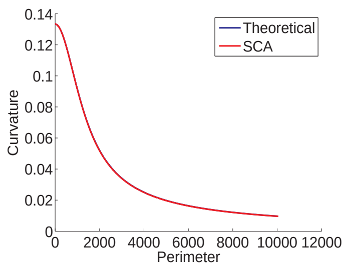
Curve in polar coordinates (r,t)	Curvature $\kappa(t)$	Average fit (L2) error	Average percentage L2 error: κ
Archimedes' spiral: $r = at$	$\frac{2+t^2}{a(1+t^2)^{3/2}}$	0.0515	1.62
Hyperbolic spiral: $r = a/t$	$\frac{t^4}{a(1+t^2)^{3/2}}$	0.0011	1.88
Circle: $r = a$	$1/a$	0.0073	0.76
Parabola: $r = \frac{-2}{1+cost}$	$\frac{1}{2(1+t^2)^{3/2}}$	0.0025	8.1

Table 5.1: Curvature estimation for standard curves, $a = 15$, total number of points on each curve = 4000, $SL = 150$, $OVERLAP = 70$.

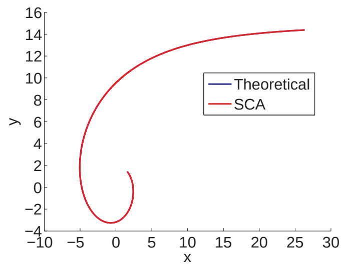
The standard curves were generated by plotting them in MATLAB. Then their fits were obtained using the weighted average-based Bézier curve fitting, and curvature was computed along the perimeter of the curve. The error depends on the granularity of the data; if more points are sampled from the theoretical curve and then subjected to curvature estimation, the accuracy increases. Fig. 5.6 (a)-(h) show the results of these experiments.



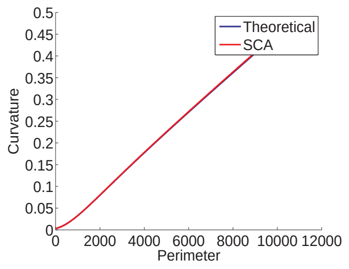
(a) Archimedes' spiral fits.



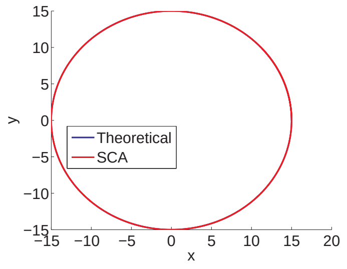
(b) Archimedes' spiral curvature.



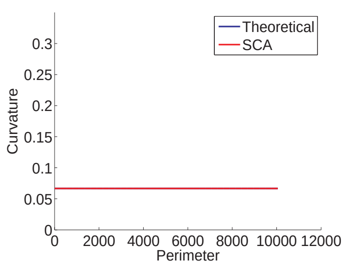
(c) Hyperbolic spiral fits.



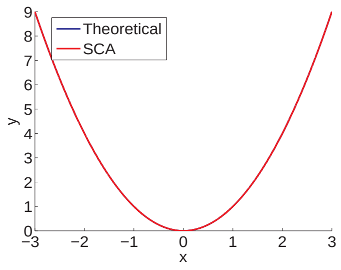
(d) Hyperbolic spiral curvature.



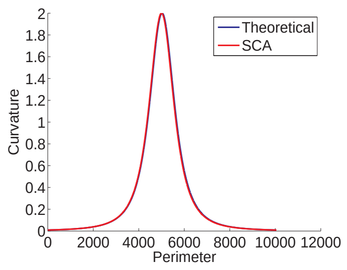
(e) Circle fits.



(f) Circle curvature.



(g) Parabola fits.



(h) Parabola curvature.

Figure 5.6: Accuracy experiments.

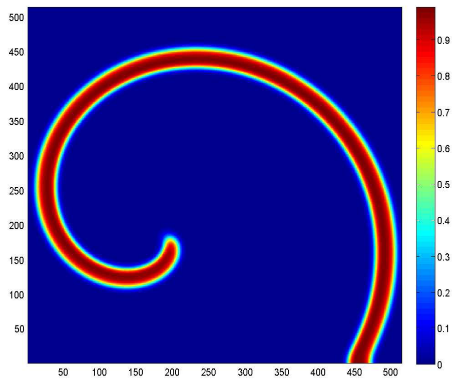
5.8 Case Studies

To demonstrate our isopotential tracking and curvature estimation techniques, we simulated six different types of tachycardia and one case of fibrillation. In an additional case study, the spiral waves resulting due to an obstacle in the tissue were investigated. We applied isopotential extraction on regularly-spaced (in time) frames from the simulation results. Then we estimated the curvature along the cardiac wave to capture the characteristic trend in time. In this section we show for each case study: 1) one typical frame from the simulation results, 2) the corresponding isopotential that was tracked, 3) weighted average-based Bézier curve fit, and 4) curvature trend, in time, for that simulation. We refer to the curve traced by the wave, along which the curvature was measured, as the wave's *perimeter*. The curvature trend plots the curvature of the wave along its perimeter, as it evolved in time. The perimeter for our case studies is measured in microns, as the simulations were performed at that scale. Curvature is therefore measured in $(\text{microns})^{-1}$ units. We used the Barkley model [2] in the first case study and thus the spatial dimension (and therefore the perimeter and the curvature) is unitless. It should be noted that to generate 4) we plot the closed-form function corresponding to curvature. The function itself can be evaluated at any resolution of the Bézier curve parameter t.

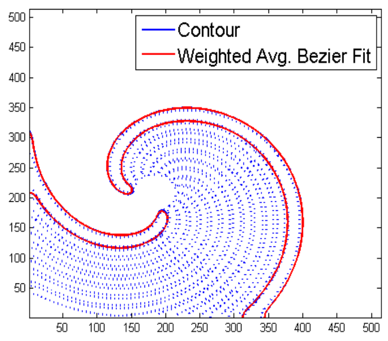
Our case studies support two findings. First we demonstrate the correctness of the isopotential extraction and the curvature estimation techniques explained above. Second, the results show the feasibility of using the curvature trend as a feature to classify different arrhythmic spiral excitation waves. For this section, we define the tip of a spiral-shaped cardiac wave as the point with the highest curvature and it separates the wavefront and the *waveback* regions of the wave. The case studies were generated such that the tips of the spirals trace trajectories of different shapes resulting in different types of abnormal propagation.

Case 1. Reentry with circular core:
In this case study we simulated the Barkley model [2] on a tissue of 514×514 cells. Simulation frames were processed at a rate of once every 10 ms in simulation time, i.e the frames were sampled at the rate of once every 10 ms, from the simulation. The core of the spiral shaped excitation waves traced a circular trajectory. One sample frame of simulation result is shown in Fig. 5.7(a). After the initial excitation, the spiral-shaped isopotentials were extracted at a scaled level of $u = 0.7$ (where u is a state-variable of the Barkley model).

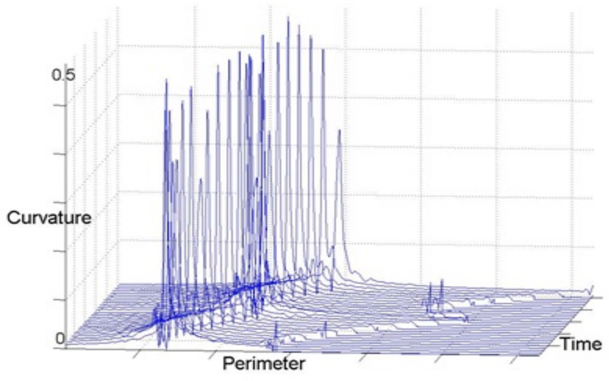
Fig. 5.7(b) shows the propagation of the isopotential with the tip tracking a circular trajectory. The waves in the first and last time steps of simulation are shown in solid blue. The overlapping red curve is the weighted average-based Bézier fit. Isopotentials are shown for intermediate frames in dotted blue. As the spiral rotates, its tip tracing a circular trajectory, the basic shape of the spiral isopotential does not change. As the basic shape of the wave remains unchanged, we would expect the curvature trend to remain the same over time. The curvature trend is shown is Fig. 5.7(c). The region of maximum curvature corresponds to the spiral tip which remains around the center of the isopotential throughout the simulation. The rest of the isopotential shows a relatively lower curvature. This trend characterizes circular-core reentrant spiral waves in the heart.



(a) Simulation, colorbar: unitless scaled potential u of the Barkley model [2].



(b) Isopotential evolution.

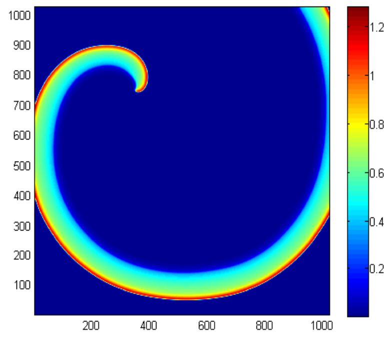


(c) Curvature Trend.

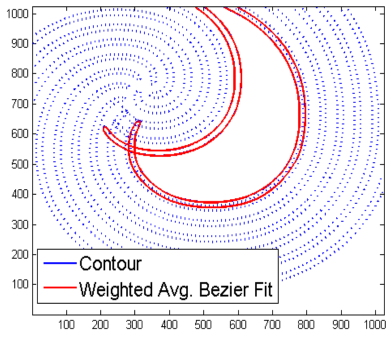
Figure 5.7: Results for case study 1: Reentry with circular core. For the curvature trend, the perimeter and the curvature are unitless and time is in steps of 10 ms.

Case 2. Re-entry with hypocycloidal core:

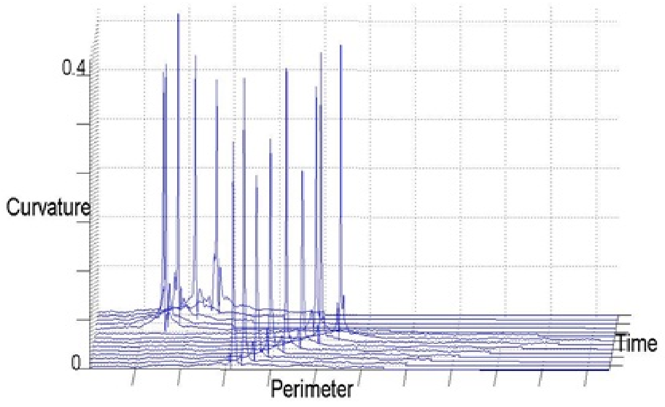
It is not always the case that the tip is located around the center of the spiral wave. To simulate asymmetric shapes, we generated waves whose tips trace hypocycloidal trajectories. In this case study, a tissue of size 1024×1024 was simulated using the Minimal model [8]. The isopotential was extracted again for $u = 0.7$ (where u is a state variable of the Minimal model) and the rate at which the frames were processed was once every 10 ms. A typical frame of the simulation can be seen in Fig. 5.8(a). Both the isopotentials extracted and the Bézier fits are shown for the first and the last frames of the simulation. The intermediate dotted blue lines are isopotentials for some of the intermediate frames.



(a) Simulation, colorbar: unitless scaled potential u of the Minimal model [8].



(b) Isopotential evolution.

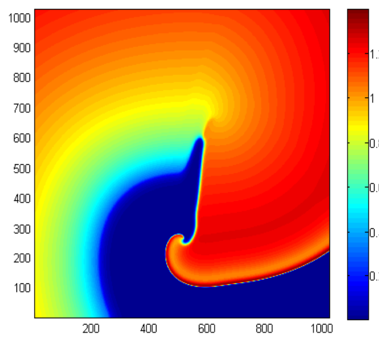


(c) Curvature Trend.

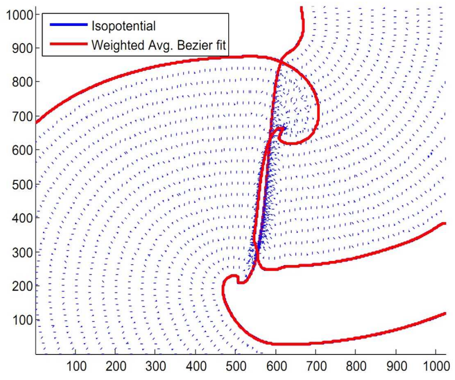
Figure 5.8: Results for case study 2: Re-entry with hypocycloidal core. For the curvature trend, perimeter is measured in μm, curvature in $(\mu m)^{-1}$ and time is steps of 10 ms.

As the tip rotates, the length of the wave changes and at times, the tip is not the center of the isopotential. This turning causes the shape of the wave to become asymmetric. The turn of the spiral is evident in the curvature trend in Fig. 5.8(c). As the length of the spiral changes, the region of highest curvature shifts on the curvature trend. This further demonstrates that a trend of morphological features like curvature, can capture the dynamics of different types of cardiac arrhythmias.

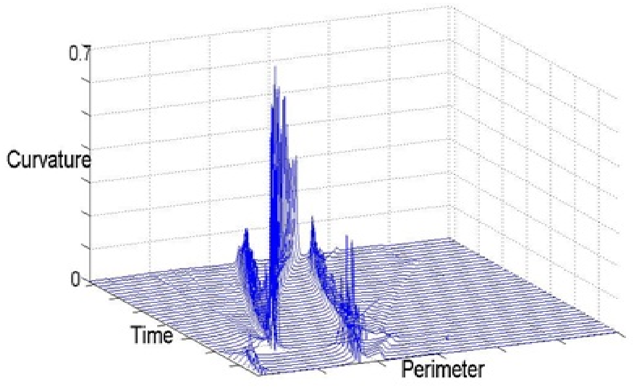
Case 3. Reentry with linear core:
In practice, spiral waves may exhibit more complex behavior. In the presence of an obstacle or deformity in the medium, the rotating waves may assume linear trajecto-



(a) Simulation, colorbar: unitless scaled potential u of the Minimal model [8].



(b) Isopotential evolution.



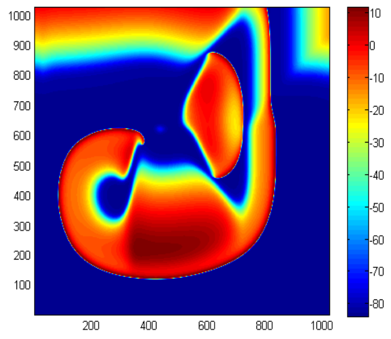
(c) Curvature trend.

Figure 5.9: Results for case study 3: Reentry with linear core. For the curvature trend, perimeter is measured in μm, curvature in $(\mu m)^{-1}$ and time is in steps of 10 ms.

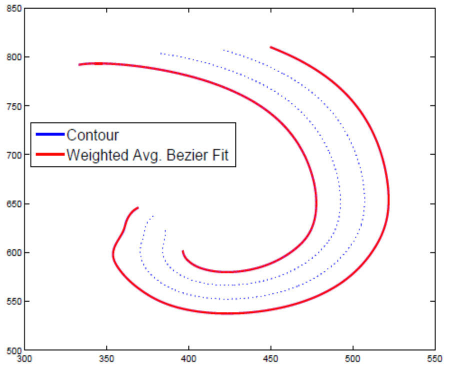
ries. We studied such waves in this case study by simulating a tissue of 1024×1024 cells using the Minimal model [8] and processed the simulation frames at a rate of once every 10ms. Isopotential extraction was done for a scaled level of $u = 1.0$. Fig. 5.9(a) shows a snapshot of the simulation. The linear trajectory along which the wave rotates can be seen with the tip of the wave at one end. Fig. 5.9(b) shows the evolving isopotential. It starts from the solid isopotential line of Fig. 5.9(b) and ends at the other solid isoline.

As shown in Fig. 5.9(c), during its linear motion, the wave has three regions of high curvature. The first two are present at the ends of the linear path of the tip. As always, the highest curvature is found at the tip which separates the wavefront and the wave-back. As the tip moves along the linear path, the separation between curvature peaks corresponding to the high curvature regions, changes. The highest peak starts near the left peak that corresponds to the first high curvature bend of the isopotential. With time, as the tip moves down to the other end of the linear path, the central curvature peak shifts toward the right peak.

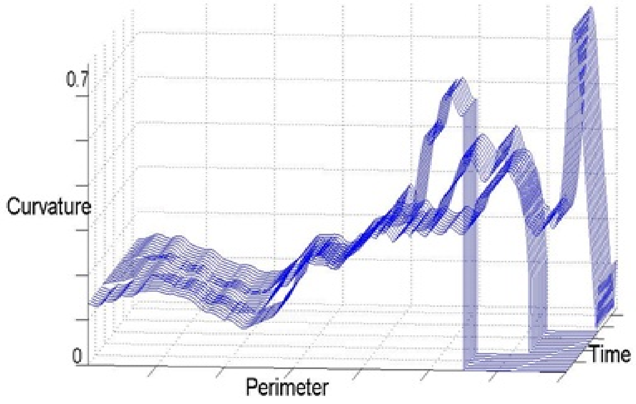
Case 4. Spiral wave breakup:
We study the onset of fibrillation in this case study. The spatio-temporal defini-



(a) Simulation, colorbar - membrane potential (mV).



(b) Isopotential evolution.



(c) Curvature trend.

Figure 5.10: Results for case study 4: Spiral break-up. For the curvature trend, perimeter is measured in μm, curvature in $(\mu m)^{-1}$ and time is in steps of 1 ms.

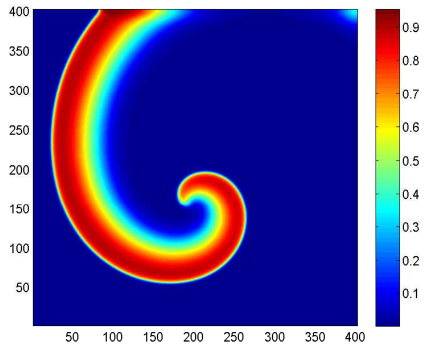
tion of the fibrillating myocardium involves the breakup of reentrant waves [49]. This breakup creates spirals which interact to produce emergent behavior. Thus, predicting the occurrence of spiral breakup is crucial to the problem of predicting fibrillation. A tissue of 1024×1024 cells was simulated using the Beeler-Reuter model [5]. Spiral breakup occurs at a very short time scale. Therefore, the frames (output) of the simulation were processed at the rate of once every 1 ms.

Fig. 5.10(a) shows one simulation frame, where the first breakup has already occurred. We tracked the isopotential of value $V = -3mV$ (where, V is the membrane potential variable in the model) until the first breakup occurred. As we approached the moment of detachment, the isopotential showed a dent near the site of break-up. This change in shape translated to the creation of a high curvature region. The changing shape of the isopotential is shown in Fig. 5.10(b). Again, the isopotential and the polynomial fits are shown for the first and last step, and the intermediate steps are shown in dashed lines. As compared to other studies, there seem to be fewer intermediate curves. This is because, with a small time step of 1 ms, most of the frames do not show any change, leading to overlapping isopotentials. The time scale at which detachment occurs forced us to increase the processing rate of the frames of the simulation.

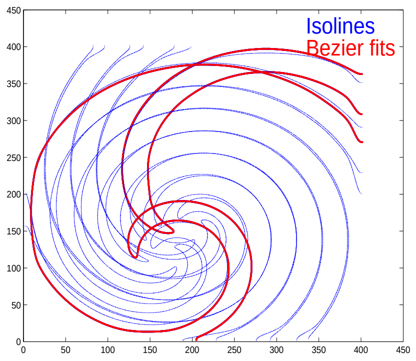
Fig. 5.10(c) shows the trend of curvature as the isopotential evolves toward breakup. Just before detachment, we see high curvature corresponding to the evolving site of breakup. Thus the gradual build-up of a high curvature site on the wave is a strong indication of future breakup eventually leading to fibrillation.

Case 5. Cycloidal core:
A tissue of 402×402 cells was simulated using the 3-variable model of [19] and the frames were processed at the rate of once every 10ms. The tip of the spiral followed a cycloidal trajectory. In this process, the length of the contour, estimated at a normalized level of $u = 0.7$ (where u is a state variable of the model), did not remain constant. The trajectory traversed by the tip can be observed in the isoline evolution shown in Fig. 5.11(b). The weighted average-based fit is shown only for the first and last frames. As shown in Fig. 5.11(c), a region of high curvature is found near the tip of the spiral. The changing position of the curvature peak, reflects the path traversed by the tip along its trajectory.

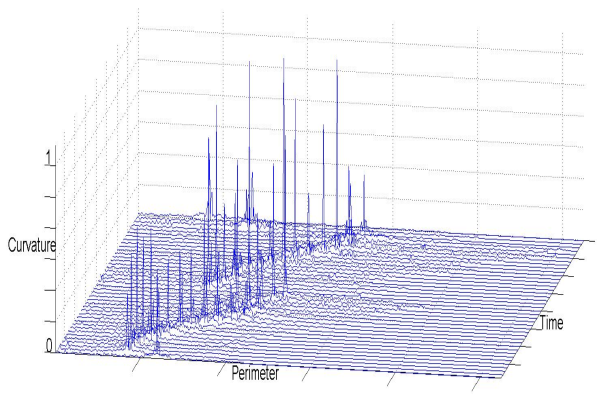
Case 6. Epicycloidal core:
In this case study, a tissue of 402×402 cells was simulated using the 3-variable model of [19], using a processing rate of once every 10 ms. In this case the tip follows an epicycloidal trajectory. Fig. 5.12(a) shows a typical frame of the simulation. The isopotential and curvature evolution show the traversal of the tip of the spiral along an epicycloidal ath. Isolines were calculated for a scaled level of $u = 0.7$ for this case study.



(a) Simulation, colorbar: unitless scaled potential u of the 3-variable model [19].



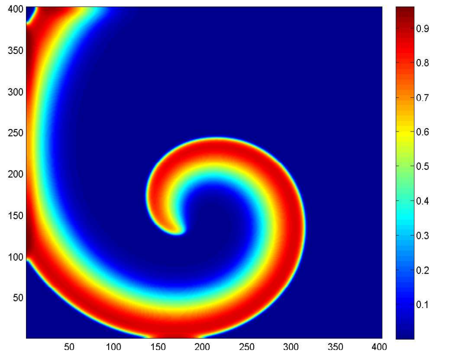
(b) Isopotential evolution.

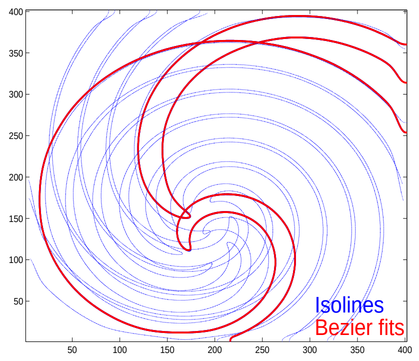


(c) Curvature trend.

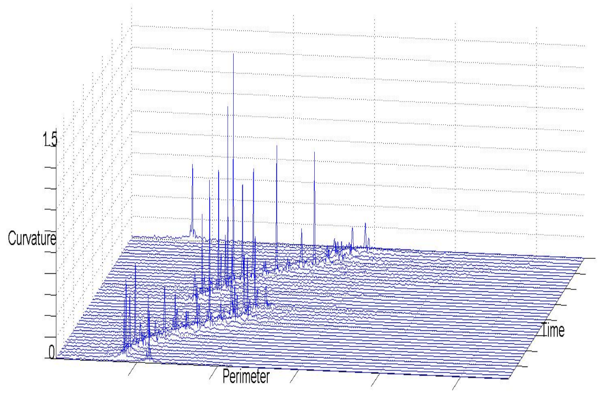
Figure 5.11: Results for case study 5: Cycloidal core. For the curvature trend, perimeter is measured in μm, curvature in $(\mu m)^{-1}$ and time in steps of 10 ms.

Case 7. Hypermeandering core:
Hypermeandering spirals involve irregular motion of the tip along various paths. For this case study, a tissue of 402 $\times$ 402 cells was simulated using the 3-variable model of [19] and the frames were processed at the rate of once every 10 ms. The isolines were estimated at a scaled level of $u = 0.9$. At this level, the spiral shape of the contour is not evident initially. This is reflected in the flat curvatures shown in Fig. 5.13(c). The starting isoline and its fit are diagrammed in Fig. 5.12(b). As this isoline moves along an unpredictable path, a spiral shape appears and results in high curvature peaks in the curvature plots of Fig. 5.13(c).


(a) Simulation, colorbar: unitless scaled potential u of the 3-variable model [19].



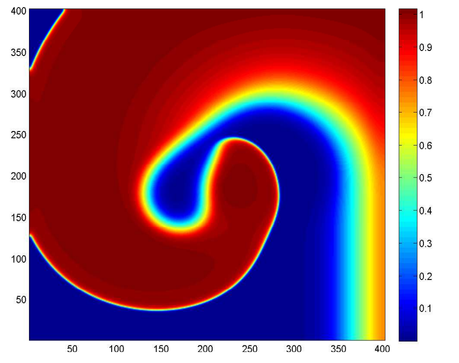
(b) Isopotential evolution.



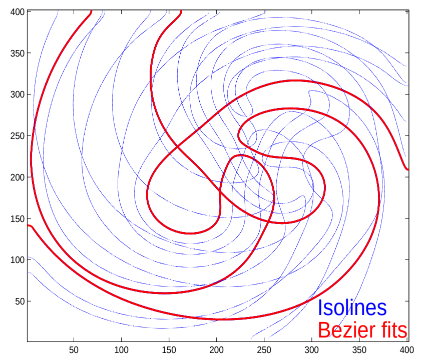
(c) Curvature trend.

Figure 5.12: Results for case study 6: Epicycloidal core. For the curvature trend, perimeter is measured in μm, curvature in $(\mu m)^{-1}$ and time is in steps of 10 ms.

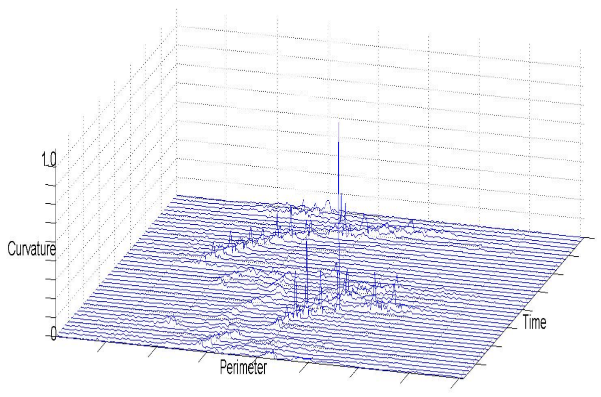
Case 8. Obstacle-induced reentry:
This case study explored the curvature trend of reentrant waves that develop due to tissue inhomogeneities. We simulated a tissue of 1038×1038 cells using the Minimal model [8] in two different settings. First, all the cells were simulated under nominal conditions, which resulted in planar excitation wavefronts shown in Fig. 5.14(a)-(e). Then, the cells lying in a circular region, were made inactive by decoupling them from the tissue, resulting in inhomogeneity. The inactive cells do not participate in the propagation of waves, and cause them to slow down. This results in spiral waves shown in Fig. 5.14(f)-(j). We only plot a part of the tissue consisting of 550×550 cells and some time steps (t) of the simulation.



(a) Simulation, colorbar: unitless scaled potential u of the 3-variable model [19].



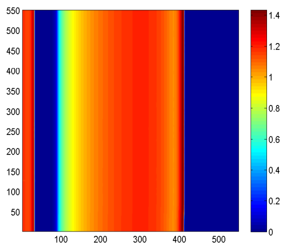
(b) Isopotential evolution.

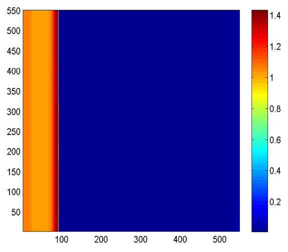


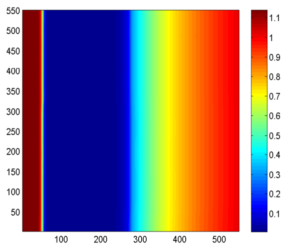
(c) Curvature trend.

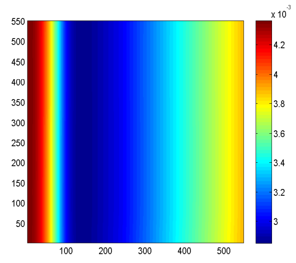
Figure 5.13: Results for case study 7: Hypermeandering core. For the curvature trend, perimeter is measured in μm, curvature in $(\mu m)^{-1}$ and time is in steps of 10 ms.

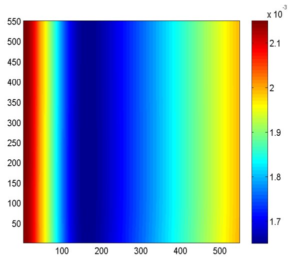
The curvature of the waves was recorded using our algorithm, as they transformed from planar to spirals. Fig. 5.8 plots the curvature trend for isopotentials of value $u = 0.7$. Initially, when the waves are planar the curvature is 0. The obstacle causes the slowing down of a part of the wave, resulting in small curvature peaks. These peaks developed into sustained patterns as the obstacle-induced arrhythmia sets in. The simulation frames were processed once every 10 ms.

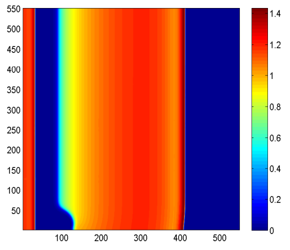

(a) t = 17.

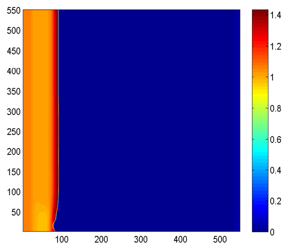

(b) t = 49.

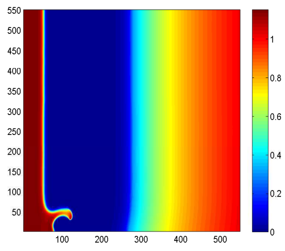

(c) t = 81.

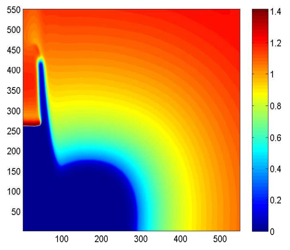

(d) t = 113.


(e) t = 145.


(f) t = 17.


(g) t = 49.


(h) t = 81.


(i) t = 113.

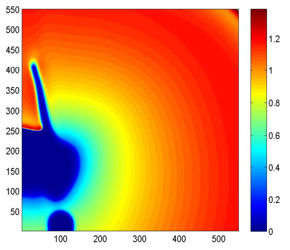

(j) t = 145.

Figure 5.14: Case study 8: Simulation frames for planar ((a) through (e)) and obstacle-induced reentrant waves ((f) through (j)).

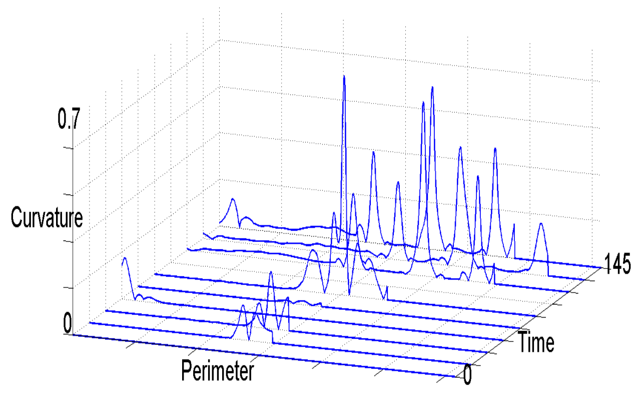


Figure 5.15: Curvature trend of reentry setting in due to tissue inhomogeneity, case study 8. The perimeter is measured in μm, curvature in $(\mu m)^{-1}$ and time in steps of 10 ms.

Chapter 6

Model-Predictive Control

In addition to modeling and analysis, control is the third important aspect of this thesis. In this chapter, we present explicit and online Model-Predictive Controllers (MPCs) for an excitable cell simulator based on the nonlinear FitzHugh-Nagumo model. Despite the plant's nonlinearity, we are able to formulate the model predictive control problem as an instance of quadratic programming, using a Piece-Wise Affine (PWA) abstraction of the plant. The speed-versus-accuracy tradeoff for the explicit and online versions is analyzed on various reference trajectories. Our MPC-based approach, enabled by the PWA abstraction, presents a framework for designing automated ***in silico*** biomedical control strategies for excitable cells, such as cardiac myocytes and neurons.

6.1 Introduction

Excitable cells, like neurons and cardiac myocytes, are building blocks of mammalian organ systems like the nervous and the cardiovascular systems. As explained Chapter 2 they exhibit characteristic cyclical APs to electrical stimuli, which could be provided externally or by neighboring cells via diffusion. The cells are arranged contiguously to form the corresponding tissue. As we saw in the previous chapter, the periodic electrical excitation and diffusion at the cell-level leads to emergent patterns of electrical-wave propagation at the tissue-level [75]. Anomalous patterns at the tissue-level are associated with potentially fatal disorders like epilepsy and cardiac arrhythmias. For example, *reentry*, which corresponds to spiral waves in the cardiac tissue, is a precursor of Atrial Fibrillation [49].

Controlling the cell-level response is critical for countering abnormal patterns at the tissue level. Excitable cells have the following distinguishing features that pose challenges to designing effective control strategies.

- **Nonlinearity**: The state space models for neurons and cardiac myocytes have highly nonlinear vector fields, which leads to multiple time scales.
- **Noise**: An actuator controlling a biological entity receives noisy readings corresponding to the state of the plant. Thus, robustness is critical while designing a control law.
- **Dimensionality**: Excitable cells are large dynamical systems and many state variables could be non-observable.

Model predictive control, a widely used process-control strategy, is well suited for biomedical applications involving excitable cells. It involves solving a finite horizon open-loop optimal control problem subject to the dynamics of the *plant*, which is the system to be controlled. Based on the measurements obtained at time T, the controller predicts the dynamic behavior of the system over a prediction horizon (T_p) and optimizes the control input over a control horizon ($Tc < T_p$) such that a predetermined open-loop performance objective function is minimized [21]. The objective function usually measures the plant's divergence from a prescribed reference trajectory, and thus is minimized by the controller. Disturbances and model mismatch constrain the controller's performance. The optimization can either be performed online (for accuracy) or can be done offline (for speed).

We present implementations of both the online and offline strategies for model predictive control of a neuron. Specifically, MPCs for a nonlinear model-based simulator of an neuron are presented. Fast and accurate model predictive control of excitable cells can be used for in-silico testing of biomedical control strategies, where a control law is designed and tested in software before fabrication. The biological entity being controlled is modeled using a simulator and the control law is tested on it, in software. Authors in [44] and [69] present novel strategies controlling and managing anomalous behaviors of neurons (epilepsy) and cardiac myocytes (ventricular tachycardia) respectively.

Model predictive control of plants with nonlinear dynamics, such as neurons, has garnered interest in the community, due to its unique challenges and wide-ranging applicability [21, 67]. The nonlinearity of the neuron dynamics results in an instance of nonlinear optimization to be solved during MPC. In general, nonlinear optimization is NP hard [50] and thus the implementation of online MPC is computationally expensive. Explicit MPC for nonlinear systems was proposed in [71], and has limited tool support. To circumvent these issues, *we adopt an approximately equivalent PieceWise Affine (PWA) abstraction of the nonlinear neuron model for both the online and explicit MPCs.* In [6], the Mixed Logical Dynamical (MLD) formalism was introduced for modeling systems whose state variables evolve continuously in time subject to logical constraints. The MPC problem for MLD systems was shown to be an instance of Mixed-Integer Quadratic Programming (MIQP). Later in [30], it was shown that PWA systems are equivalent to MLD systems. Thus, converting the nonlinear neuron model to an approximately equivalent PWA form transforms the corresponding MPC problem from an instance of nonlinear optimization to one of MIQP. Also, the PWA abstraction enables the design of explicit MPC by using the Multi-Parametric Toolbox (MPT) [53] in MATLAB.

Next, we outline the architecture of the controllers, see Fig. 6.1. The plant simulates an excitable cell and outputs the AP corresponding to the input stimuli provided by the MPC. The MPC's goal is to compute optimal inputs such that the plant tracks, in discrete time, a reference trajectory that consists of a nominal sequence of APs.

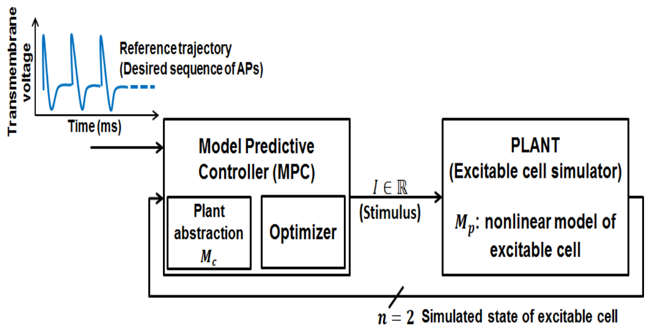

Transmembrane voltage
Time (ms)
Reference trajectory (Desired sequence of APs)
Model Predictive Controller (MPC)
Plant abstraction M_c
Optimizer
$I \in \mathbb{R}$
(Stimulus)
PLANT (Excitable cell simulator)
M_p: nonlinear model of excitable cell
$n = 2$ Simulated state of excitable cell

Figure 6.1: Architecture for tracking action potentials of nonlinear excitable cells using MPCs.

1. The plant uses a nonlinear model M_p of an excitable cell for its simulation. We use the FitzHugh-Nagumo (FHN) model [37], described in Section 6.2, as M_p.

2. The plant outputs the n-dimensional state of M_p as the state of the underlying excitable cell. For the FHN model $n = 2$, and one of the state variables is the dimensionless transmembrane potential, which tracks the reference trajectory.

3. The MPC uses a PWA abstraction, M_C, of the plant model, to predict the behavior of the cell under simulation. We use a modified version of the hybrid model proposed in [38], henceforth referred to as the (DR) model, as M_c. The PWA abstraction is used to cast the MPC's optimization problem as an instance of MIQP.

4. MPC is also equipped with an optimizer to compute the optimal stimulus input I, such that the observed state of the plant tracks a pre-defined reference trajectory.

The following simplifying assumptions are made in our implementation, and justified in the appropriate sections of the chapter:

1. The plant's state is completely visible to the MPC. Ideally, only the membrane potential is measurable and the internal state is hidden.
2. No exogenous inputs (noise) are considered in the current implementation.
3. The only mismatch between the plant and its model, M_p, used by the MPC is due to the PWA abstraction.

We summarize our contributions below before outlining the remaining sections.

1. MPCs for a nonlinear model-based neuron have been designed by using a PWA abstraction. The resulting MIQP optimization instance is solved using both online and explicit approaches.
2. The PWA abstraction is used to enable the design of explicit MPC in MPT. The toolbox has been extended to track moving reference trajectories by augmenting the state vectors and thus making the penalty matrices time-varying.
3. The online and explicit approaches to the PWA abstraction-based MPC are compared using several test cases to analyze the tradeoff between accuracy and speed.

The remainder of the chapter is organized as follows. The next section introduces the two models M_p and M_c in detail. We formulate the MPC problem for the FHN model-based plant in Section 6.3. The implementation details follow in Section 6.4. Then, we compare and contrast the online and explicit strategies in Section 6.5. We discuss related work in Section 6.6, and present concluding remarks in Section 6.7.

6.2 Physiological Background

As mentioned in the previous section, excitable cells are characterized by their response to an external electric current, called the stimulus. ODE models capture the behavior of excitable cells in terms of the change in the transmembrane potential in time, as the cell oscillates between depolarization and repolarization in response to the stimulus.

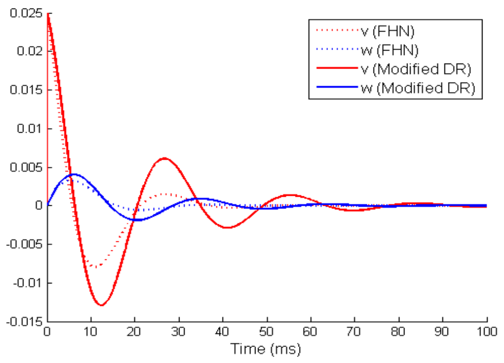


Figure 6.2: Simulations of the FHN and the MDR model. Maximum absolute L1 error for v = 0.0056 and w = 0.0013.

The FHN model [37] is a two-dimensional system of differential equations, representing the dynamics of a neuron:

$$\dot{v} = v(1 - v)(v - a) - w + I(t), \tag{6.1a}$$
$$\dot{w} = bv - cw, \tag{6.1b}$$

where v is the dimensionless transmembrane potential, w is a dimensionless recovery variable, I is the magnitude of the stimulus current and the parameters a, b and c are given in Table 6.2.

Parameter	a	b	c
Value	0.20	0.05	0.01

Table 6.1: Parameters of the FHN model (M_p) used by the plant.

The MPC uses a Modified Dumas-Rondepierre (MDR) model [38], a PWA version of the FHN model, to predict the plant's behavior (simulation of the excitable cell). The cubic term in (6.1a) is linearized to obtain the PWA dynamics (6.2):

$$\dot{v} = \tilde{p}(v) - w + I(t), \tag{6.2}$$

where

$$\tilde{p}(v) = \begin{cases} \frac{p(v_-)}{v_-} v & v < v_- \\ \left[\frac{p(v_+)-p(v_-)}{v_+-v_-}\right] v + \left[p(v_+) - \frac{p(v_+)-p(v_-)}{v_+-v_-} v_+\right] & v_- \leq v \leq v_+ \\ \frac{p(v_+)}{1-v_+}(1 - v) & v > v_+. \end{cases} \tag{6.3}$$

The constants v_+ and v_- are given by

$$v_- = \frac{a + (1 - \sqrt{a^2 - a + 1})}{3}, \text{and} \tag{6.4a}$$

$$v_+ = \frac{a + (1 + \sqrt{a^2 - a + 1})}{3}. \tag{6.4b}$$

The function $p(.)$ is given by

$$p(v_-) = v_-(1 - v_-)(v_- - a), \text{and} \tag{6.5a}$$

$$p(v_+) = v_+(1 - v_+)(v_+ - a). \tag{6.5b}$$

The MDR model model represents M_c used by the controller to predict the plant's behavior and compute optimal stimuli values. It can be viewed as a hybrid model consisting of three modes and can be written in the following format:

$$\dot{\mathbf{x}} = A_i\mathbf{x} + B_i\mathbf{u} + \mathbf{f_i} \tag{6.6}$$

$$\mathbf{y} = C_i\mathbf{x} + D_i\mathbf{u} + \mathbf{g_i} \tag{6.7}$$

where

- i = index of the mode (piece),
- $\mathbf{x(t)} = \mathbb{R}^n$ state vector,
- $\mathbf{u(t)} = \mathbb{R}^m$ input vector,
- $\mathbf{y(t)} = \mathbb{R}^p$ output vector,
- $A_i = n \times n$ the *Dynamics matrix* for mode i,
- $B_i = n \times m$ the Input matrix for mode i,
- $C_i = p \times n$ the Output matrix for mode i,
- $D_i = p \times m$ the Feed-through matrix,
- f_i = non-homogeneity in dynamics - real vector of size $n \times 1$ and
- g_i = real vector of size $p \times 1$.

In the MDR model, we have:

1. Three modes, i.e., $1 \leq i \leq 3$.
2. Two states v and w, i.e., $n = 2$.
3. One input, Stimulus I, i.e., $m = 1$.

4. Two outputs v and w produced by plant, i.e., $p = 2$ (all the states are observable).

The MPC works in discrete time. MATLAB's *c2dm* function was used, with a sampling rate of 0.01 sec/sample and the *zero-order hold (zoh)* method, to generate the discrete time version of the MDR model. We obtain the following matrices based on the parameters in Table 6.2. The superscript "d" denotes the corresponding matrix in the discrete time version.

1. <u>Mode 1, i = 1</u>

   (a) Invariant: $v < 0.0945$.

   (b) $A_1 = \begin{bmatrix} -0.0955 & -1 \\ 0.05 & -0.01 \end{bmatrix}$,
   $A_1^d = \begin{bmatrix} -0.9990 & -0.01 \\ 0.0005 & 0.9999 \end{bmatrix}$.

   (c) $B_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $B_1^d = \begin{bmatrix} 0.01 \\ 0 \end{bmatrix}$.

   (d) $f_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $f_1^d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

   (e) $C_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $C_1^d = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

   (f) $D_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $D_1^d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

   (g) $g_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $g_1^d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

2. <u>Mode 2, i = 2</u>

   (a) Invariant $= 0.0945 \le v \le 0.7055$.

   (b) $A_2 = \begin{bmatrix} 0.1867 & -1 \\ 0.05 & -0.01 \end{bmatrix}$,
   $A_2^d = \begin{bmatrix} 1.0019 & -0.01 \\ 0.0005 & 0.9999 \end{bmatrix}$.

   (c) $B_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $B_2^d = \begin{bmatrix} 0.01 \\ 0 \end{bmatrix}$.

(d) $f_2 = \begin{bmatrix} -0.0267 \\ 0 \end{bmatrix}$, $f_2^d = \begin{bmatrix} -0.0267 \\ 0 \end{bmatrix}$.

(e) $C_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $C_2^d = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

(f) $D_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $D_2^d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

(g) $g_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $g_2^d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

3. <u>Mode 3, i = 3</u>

(a) Invariant = $v > 0.7055$.

(b) $A_3 = \begin{bmatrix} -0.3566 & -1 \\ 0.05 & -0.01 \end{bmatrix}$,

$A_3^d = \begin{bmatrix} 0.9964 & -0.01 \\ 0.0005 & 0.9999 \end{bmatrix}$.

(c) $B_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $B_3^d = \begin{bmatrix} 0.01 \\ 0 \end{bmatrix}$.

(d) $f_3 = \begin{bmatrix} 0.3566 \\ 0 \end{bmatrix}$, $f_3^d = \begin{bmatrix} 0.3566 \\ 0 \end{bmatrix}$.

(e) $C_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $C_3^d = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

(f) $D_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $D_3^d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

(g) $g_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $g_3^d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

Fig. 6.2 compares the FHN model and the MDR model in continuous time. A stimulus consisting of a spike of height 2.5 at the first time step was used to excite the model. The simulation was performed in MATLAB using the Euler method using a time step of 0.01 ms till 100 ms. Initial conditions were $v = 0$ and $w = 0$.

6.3 MPC Problem Formulation

Based on the $n \times 1$ state measurement (assuming that the complete state of the plant is observable: a state estimator would be needed in case of partial observability.)

$\mathbf{x}(t)$ obtained at time t, the MPC predicts the dynamic behavior of the system and optimizes the control inputs such that the objective function in Eq. (6.8) is minimized:

$$\begin{aligned}
&\underset{U=[\mathbf{u}(t),\ldots,\mathbf{u}(t+N-1)]}{\text{minimize}} \quad J(U,\mathbf{x}(t)) = \\
&\sum_{k=1}^{N}[(\mathbf{x}(t+k)-\mathbf{x}_{ref}(t+k))'\lambda^k.Q(\mathbf{x}(t+k)-\mathbf{x}_{ref}(t+k)) \\
&+(\Delta\mathbf{u}(t+k-1))'R(\Delta\ \mathbf{u}(t+k-1))] \\
&\text{subject to:} \\
&\mathbf{x}(t+k+1) = A_i^d\mathbf{x}(t+k) + B_i^d\mathbf{u}(t+k) + \mathbf{f}_i^d, \\
&\mathbf{y}(t+k) = C_i^d\mathbf{x}(t+k) + D_i^d\mathbf{u} + \mathbf{g}_i, \\
&\text{where} \\
&\Delta\mathbf{u}(t+k-1) = \mathbf{u}(t+k-1) - \mathbf{u}(t+k-2) \\
&0 \le k \le N-1 \text{ and } 1 \le i \le 3.
\end{aligned} \tag{6.8}$$

Optimization is performed over a finite horizon of length N. Q is an $n \times n$ identity matrix and $0 < \lambda \le 1$ is a parameter that assigns exponentially receding weights to the predicted deviations, $(\mathbf{x}(t+k)-\mathbf{x}_{ref}(t+k))$, over the horizon. Thus, the scheme is also called receding horizon control. R is a positive definite matrix that determines the penalty on differences between consecutive inputs.

The optimization problem is solved at time t and the inputs are calculated for the next N time steps. Only the next input is passed on to the plant, before repeating the MPC process.

6.4 Implementation of Model Predictive Controllers for the FHN model-based Plant

Explicit and online MPCs were implemented for the FHN model-based plant using the MDR model for prediction purposes. The receding horizon parameter λ was fixed at 0.8 and $R = \left[10^{-3}\right]$ was the input penalty matrix. Next, we describe the implementation aspects of the online and explicit MPCs.

6.4.1 Online MPC

Online MPC involves solving Eq. (6.8) at every time step in runtime. The constrained nonlinear optimizer *fmincon* [56] was used to implement online MPC in MATLAB. At each time step t, the current state $\mathbf{x}(t)$ of the plant and the reference input $[\mathbf{x}_{ref}(t+1), ..., \mathbf{x}_{ref}(t+N)]$ over the finite horizon N were provided to the controller which then computed the optimal input for the FHN plant. An interior point algorithm was used for optimization. The FHN plant was then simulated using Euler method for one time step by applying the optimal input. This process was repeated for the whole simulation duration.

6.4.2 Explicit MPC

In explicit MPC, the optimization problem of Eq. (6.8) is cast as an instance of multi-parametric quadratic programming (mpQP) and solutions are computed offline for possibly overlapping polyhedral partitions, also known as *coverings*, of the state space. As shown in Fig. 6.3, the result of this one-time computation is a table of control laws corresponding to the partitions. At runtime, the current state sample is tested for membership in the list of partitions. The state may lie in more than one region due to possible overlap. In this case, the control law resulting in the most optimal value of the objective function is applied. The process is then repeated for the next state sample.

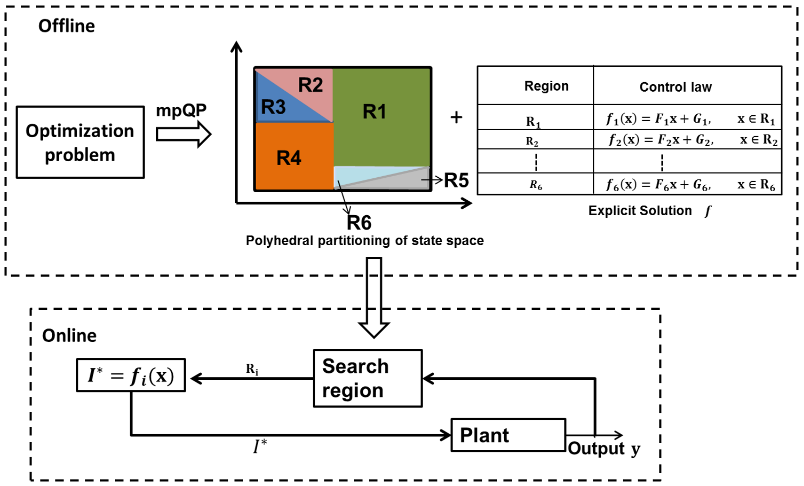

Offline
Optimization problem
mpQP
R2
R3
R1
R4
R5
R6
Polyhedral partitioning of state space
+
Region
Control law
R_1
$f_1(\mathbf{x}) = F_1\mathbf{x} + G_1, \quad \mathbf{x} \in R_1$
R_2
$f_2(\mathbf{x}) = F_2\mathbf{x} + G_2, \quad \mathbf{x} \in R_2$
R_6
$f_6(\mathbf{x}) = F_6\mathbf{x} + G_6, \quad \mathbf{x} \in R_6$
Explicit Solution f
Online
$I^* = f_i(\mathbf{x})$
R_i
Search region
Plant
I^*
Output y

Figure 6.3: Workflow for explicit MPC.

We implemented explicit MPC for the FHN model-based neuron simulator using the MDR model as its PWA abstraction. MATLAB's MPT [53] was used for the implementation. The current implementation of MPT supports time-varying reference trajectories, but it considers constant reference at every time step of prediction horizon. We extended the tool to overcome this limitation. In the remaining sections, we elaborate on these modifications.

The key idea for incorporating time-varying reference trajectories is as follows. The reference trajectory over the prediction horizon, $\mathbf{x}_{ref}$ is considered to be to be a sequence of unknown variables. Then these unknown variables are used to augment the state vector $\mathbf{x}$. Then, the dynamics of the augmented system is reformulated in a Δu-form, as the input necessary to keep the states at the reference are also not generally known. In this formulation, the input at time k is $\Delta u(k)$, where $u(k-1)$ is an additional state in the dynamical model. So, the original system input can be obtained as $u(k) = u(k-1) + \Delta u(k)$. The state update equation is then given by Eq. (6.9).

$$\begin{pmatrix} x(k+1) \\ u(k) \\ x_{ref}(k+1) \end{pmatrix} = \begin{pmatrix} A_i & B_i & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} x(k) \\ u(k-1) \\ x_{ref}(k) \end{pmatrix} + \begin{pmatrix} A_i \\ B_i \\ 0 \end{pmatrix} \Delta u(k), \tag{6.9}$$

As the state vector is augmented with new state variables, the penalty matrix Q also needs to be augmented. The newly augmented penalty matrix is given by

$$\begin{pmatrix} Q & 0 & -Q \\ 0 & 0 & 0 \\ -Q & 0 & Q \end{pmatrix}$$

In our modification scheme, we consider time-varying reference at all steps of the prediction horizon. We augment x with the reference state vector for all steps of the horizon. The newly modified state update equation is given by Eq. (6.10).

$$\begin{pmatrix} x(k+1) \\ u(k) \\ x^1_{ref}(k+1) \\ \vdots \\ x^N_{ref}(k+1) \end{pmatrix} = \begin{pmatrix} A_i & B_i & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} x(k) \\ u(k-1) \\ x^1_{ref}(k) \\ \vdots \\ x^N_{ref}(k) \end{pmatrix} + \begin{pmatrix} A_i \\ B_i \\ 0 \end{pmatrix} \Delta u(k), \tag{6.10}$$

where N is the number of steps in the horizon and $x^j_{ref}(k)$ is the reference vector in j^{th} time step of the horizon.

Due to the receding horizon principle, the penalty matrix will be different for each step of prediction horizon. The current implementation of the MPT is amenable to adding a variable penalty matrix. The general form of the penalty matrix is

$$\begin{pmatrix} Q(k) & 0 & -Q(k) & 0 \\ 0 & 0 & 0 & 0 \\ -Q(k) & 0 & Q(k) & 0 \end{pmatrix},$$

where $Q(k) = \lambda^k Q$ for $k-th$ step of the prediction horizon. The positions of $-Q(k)$ in the first row and $Q(k)$ in the third row are adjusted based on k.

6.5 Results

We conducted two sets of experiments on the online and the explicit MPCs to compare and contrast them.

Experiment Set 1: *Speed vs. Accuracy Tradeoff for Different Reference Trajectories*

The online and the explicit MPCs were tested against the same reference trajectory to study the speed vs. accuracy tradeoff. Two reference trajectories $[v_r^i(t), w_r^i(t)]$, $i = 1, 2$ were generated by simulating the FHN model. The protocols used to generate them were as follows.

S1 Protocol for generating $[v_r^1(t), w_r^1(t)]$

1. Initial conditions: $v = 0$, $w = 0$ (rest conditions).
2. Time step used in the simulation: 0.1 ms.
3. Total duration of simulation: 240 ms (2400 time steps).
4. Stimuli pattern: One time-step-long (0.1ms) supra-threshold stimulus pulses of intensity (height) 1.5 were applied every 80 ms, to produce three APs in the simulation. Thus, the pacing frequency was 12.5 Hz.

S2 Protocol for generating $[v_r^2(t), w_r^2(t)]$

1. Initial conditions: $v = 0$, $w = 0$ (rest conditions).
2. Time step used in the simulation: 0.1 ms.
3. Total duration of simulation: 240 ms (2400 time steps).
4. Stimuli pattern: 10-steps-long (1 ms) supra-threshold stimulus pulses of intensity (height) 1.5 were applied every 80 ms, to produce three APs in the simulation. Thus, the pacing frequency was again 12.5 Hz.

The simulation was carried out using the Euler's method of numerical integration in MATLAB. Both the MPCs were tested against the S1 and S2 reference trajectories using a 3-step lookahead horizon. Their performance was compared using the following two metrics:

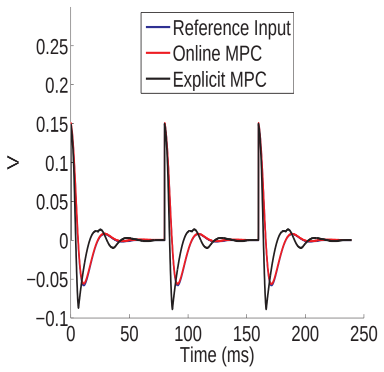
1. **Accuracy of the plant's operation** (μ_{l2}^v, μ_{l2}^w): measured using the mean L2 error between the reference trajectory and the output of the simulation carried out by the plant.

2. **Timeliness constraint on the MPC**: dictates that the working of the MPC must be fast enough to cope with the plant's operation. The degree to which the two MPCs met this constraint was measured as follows. The time taken by the Euler method-based simulation for producing the reference trajectory was noted, say t_1 secs. The plant + MPC combination was run on a single thread in a lock-step fashion, i.e., the plant was halted till the MPC finished its computation and provided the stimuli value for the next time step. The total time taken for tracking the reference trajectory was noted, say t_2 secs. Then, $t_{12} = (t_2 - t_1)$ provided an estimate of the time taken by the MPC to compute the stimuli. Ideally, $t_{12} < t1$, which ensures that the MPC's computation runs faster than the rate at which the plant evolves (simulates the FHN model).

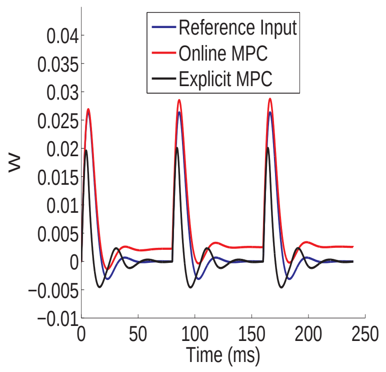
Table 6.2 provides performance metrics for the two MPCs. Fig. 6.4 and Fig. 6.5 plot the evolution of v and w for protocols S1 and S2, respectively.

Protocol	**Controller**	μ^v_{l2}	μ^w_{l2}	$t_1(s)$	$t_2(s)$
S1	**Online MPC**	2.8×10^{-5}	4.9×10^{-5}	0.023	136.8
	Explicit MPC	4.3×10^{-4}	1.1×10^{-4}	0.023	88.8
S2	**Online MPC**	1.8×10^{-4}	2.2×10^{-4}	0.023	147.3
	Explicit MPC	2.7×10^{-2}	1.1×10^{-2}	0.023	88.3

Table 6.2: Performance metrics for assessing the speed vs. accuracy tradeoff across different reference trajectories.

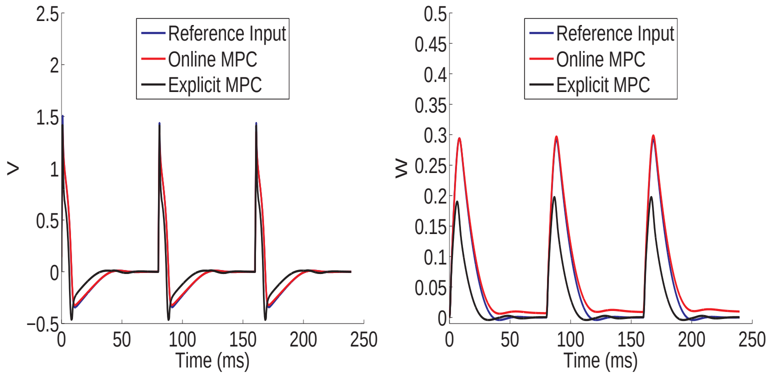


(a) Evolution of v.



(b) Evolution of w.

Figure 6.4: Performance of the online and explicit MPCs on spike-shaped stimuli produced by the S1 protocol.



(a) Evolution of v. (b) Evolution of w.

Figure 6.5: Performance of the online and explicit MPCs on rectangular pulse-shaped stimuli produced by the S2 protocol.

Discussion

Following inferences can be made from the results shown in the preceding paragraphs:

1. *The accuracy of the fmincon-based online MPC is better than the MPT-based explicit MPC.* This can be attributed to an accurate solution to the optimization problem, found by fmincon at run time, for the specific current state of the plant. The explicit MPC on the other hand, partitions the state space and finds a common control law for the whole partition. Accuracy is lost in this process.

2. *The fmincon-based online MPC is much slower than the MPT-based explicit MPC.* FMINCON exhaustively explores the whole state space at every time step of the plant's operation. This leads to its slow operation. The most time consuming step for the explicit MPC is searching for the partition corresponding to the current state, and this is achieved much faster than the online MPC's operation.

Experiment Set 2: *Effect of Horizon Length on Explicit MPC*
Explicit MPC is enabled for time-varying reference trajectories by augmenting the state vectors and reformulating the dynamics. State space augmentation leads to an exponential increase in the number of polyhedral partitions. Table 6.3 compares the build-time and the number of partitions for different horizon lengths. Increasing the horizon length N is expected to improve the predictive accuracy of MPC. In the case of MPT-based design of explicit MPC, we observed that accuracy did not improve considerably on changing the horizon length from 2-step to 3-step lookahead. For

Horizon length	Build-time in MPT (secs.)	Number of partitions
0	0.45	3
1	2.78	30
2	51.36	277
3	576.41	2581

Table 6.3: Effect of N on explicit MPC design in MPT. Horizon length of 0, which corresponds to 0-step lookahead, is specified for comparison purposes.

both cases, the mean L2 errors for v and w were recorded to be around 0.027 and 0.001 respectively (for S2-type stimuli).

Having a smaller horizon leads to significant reduction in the search space, while searching for the partition corresponding to a given state sample. This reduction is critical as the search operation is performed at every time step during operation. In our case, t_2 reduced from 88.3 secs. to 11.2 secs when the horizon length was changed from 3 to 2 for the S2 protocol.

6.6 Related Work

MPC has been widely used in many domains like the chemical, food-processing, automotive, and aerospace industries. An exhaustive survey of both the theoretical and the practical aspects can be found in [24]. Explicit MPC, which is relatively new, has been surveyed in [1]. Recently MPC has found interesting biological and biomedical applications. In [39, 73], a platform for in silico realtime closed-loop control of gene expression in yeast has been proposed. It uses MPC to perturb inducible promoters in a systematic way to gain insights about gene expression. MPC has been successfully applied to devise therapeutic strategies in [9, 10, 63, 41]. In [59] MPC is used for functional electrical stimulation to estimate stimulation patterns for muscles that have been paralyzed due to spinal cord injury. Controllers for tracking neuron APs are designed in [62, 7] and thus are closest to our work. We compare and contrast each of them with our MPC-based approach below.

In [62], an adaptive input-output feedback linearization controller is presented to track a nominal AP using the FHN model. In contrast, MPC is a feed-forward control technique. Its predictive capability allows the controller to quickly adapt to a model mismatch caused due to the degradation/aging of excitable cells. Parameter estimation and tuning the model are the only steps involved in adapting to the changes, whereas a feedback controller needs complete redesign. Also, explicit MPC can track arbitrary fast-changing APs whereas the controller in [62] is designed for a nominal AP.

In [7], the authors present a sophisticated controller for a neuron based on the Hodgkin-Huxley (HH) model [31]. The HH model is augmented with random variables to capture stochastic behavior and external disturbances. Membrane potential is treated as the only observable and a state-estimator is employed for the other hidden variables of the HH model. On the other hand, we focus on comparing the online and explicit approaches to MPC in the case of a nonlinear plant being modeled using a PWA abstraction. The realistic setting of [7] is complementary to our work and provides directions for extending our scheme. Also, the FHN and the MDR models used in our work are order-reduced versions of the HH model.

6.7 Conclusions

Explicit and online MPC was presented for tracking a reference sequence of APs using an FHN model-based neuron simulator. The controllers employ a PWA abstraction of the nonlinear plant, thus enabling a QP formulation of the model predictive control optimization problem. The speed versus accuracy tradeoff was assessed using several test cases. The online approach provides excellent accuracy, but fails to satisfy the timeliness constraint. Offline MPC on the other hand, satisfies the timeliness constraint for a limited set of reference trajectories, but provides relatively lower accuracy than the online version.

Chapter 7

Conclusions and Directions for Future Research

We begin with concluding remarks, and then present some directions for future research.

Conclusions

In Chapter 3, we presented a model-order reduction technique to enable the construction of a tower of abstraction for cardiac models, as described in Chapter 1. Reduction entails approximating the subsystems of detailed cardiac models using HH-type models. We constructed two-state HH-type models, Σ_{HI} and Σ_{HK}, corresponding to the detailed ion channel components, Σ_I (13-state) and Σ_K (10-state) respectively, of the IMW model.

We then presented a proof technique to show that the detailed subsystem can be substituted by the HH-type abstraction within the whole-cell model. The proofs entail computing Lyapunov-like BFs. In Chapter 4, we presented two algorithms for computing the BFs. We used the small-gain theorem to show that Σ_I and Σ_K can be replaced by Σ_{HI} and Σ_{HK} within the CCMs. This substitution of (approximately) equals for equals is "safe" in the sense that, despite the feedback nature of the composition of the detailed ion-channel component and the rest of the IMW model, the approximation error remains bounded in this context.

After identifying a model with the requisite level of detail, its analysis becomes important. One of the pertinent analysis questions is the parameter identification problem. Technological developments within the graphics processing community, NVIDIA in particular, coupled with theoretical advances in the computer-aided verification community, have set the stage for fast simulation-based parameter identification for cardiac models. In Chapter 5, we presented a key component of such a framework: a parallel curvature analysis algorithm that given a series of frames generated via simulation or optical mapping, produces a curvature-based signature of the wave/spiral captured in the frames. Our isopotential reconstruction algorithm takes advantage of NVIDIA's Tesla and Fermi graphics processing cards, and the associated CUDA architecture. Our results demonstrate speed-up by a factor of 444.44 for isopotential reconstruction compared to the MATLAB-based contour algorithm. Our case studies identified distinct signatures for various forms of cardiac arrhythmias (eight in total), which may be used to classify a wave by its spiral type.

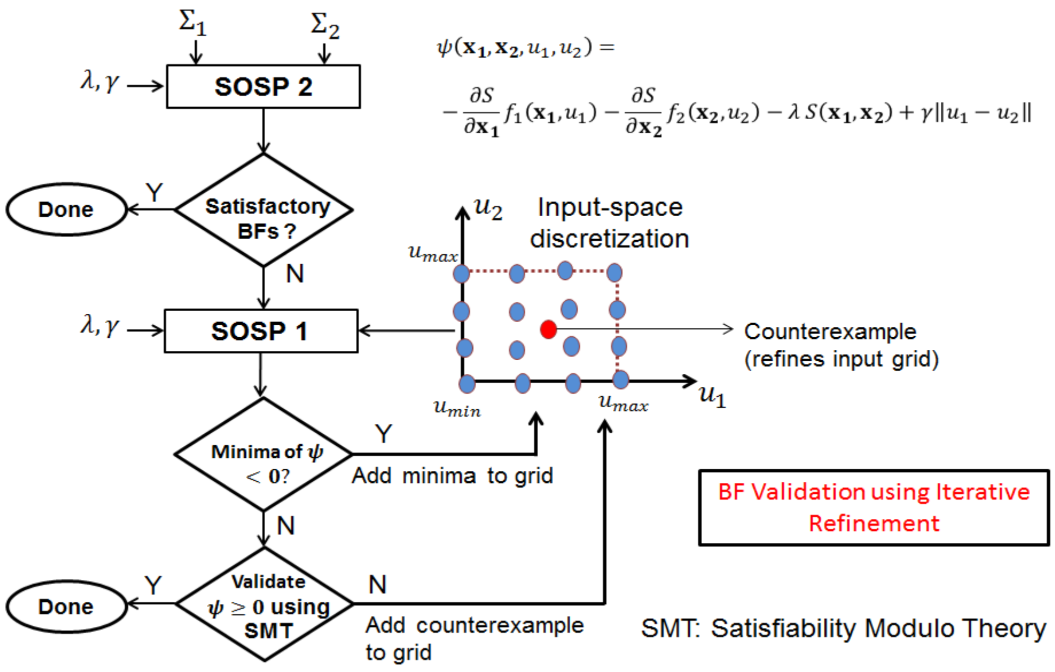
In Chapter 6, explicit and online MPCs were presented for tracking a reference sequence of APs using an FHN model-based neuron simulator. The MPCs employ a PWA abstraction of the nonlinear plant, thus enabling a QP formulation of the model predictive control optimization problem. The speed versus accuracy tradeoff was assessed using several test cases. The online approach provides excellent accuracy, but fails to satisfy the timeliness constraint. The offline MPC, on the other hand, satisfies the timeliness constraint for a limited set of reference trajectories, but provides relatively lower accuracy than the online version.

Directions for Future Research

In Chapter 4, we presented SOSP 1 and SOSP 2, two SOS optimization-based BF-computation algorithms. SOSP 1 produces functions that satisfy Eq. 4.3 only on a sampled subset of the input space, which is defined by grid-based discretization. SOSP 2 overcomes this limitation by adding the unknowns σ_1 and σ_2, see Section 4.7, Eqs. (4.11)-(4.13). Despite overcoming the restriction of SOSP 1, SOSP 2 provides relatively weaker bounds on the OD. Thus, the algorithms present a tradeoff: soundness of BFs versus the bound on the OD.

Fig. 7.1 illustrates a revised workflow, which combines SOSP 1 and SOSP 2, along with a validation procedure for BFs computed using SOSP 1. The workflow is as follows. Given the two dynamical systems Σ_1 and Σ_2, SOSP 2 is implemented to obtain a BF. If the BF provides satisfactory bounds on the OD, then we are done. If the user seeks a tighter bound, SOSP 1 is implemented, using an input-space discretization, as per Section 4.5. The resulting function, now known as a *candidate BF* satisfies the decay condition only on the grid points. In other words, the $\psi \geq 0$, in Fig. 7.1 holds on the grid points shown in blue.

The candidate BF is then validated using a technique based on [45]. First, a local minima is computed for ψ, if it is negative, then the input pair (u_1, u_2)



Σ_1
Σ_2
λ, γ
SOSP 2
$\psi(\mathbf{x_1}, \mathbf{x_2}, u_1, u_2) =$
$-\frac{\partial S}{\partial \mathbf{x_1}} f_1(\mathbf{x_1}, u_1) - \frac{\partial S}{\partial \mathbf{x_2}} f_2(\mathbf{x_2}, u_2) - \lambda S(\mathbf{x_1}, \mathbf{x_2}) + \gamma \|u_1 - u_2\|$
Done
Y
Satisfactory BFs ?
N
u_2
Input-space discretization
u_{max}
λ, γ
SOSP 1
Counterexample (refines input grid)
u_{min}
u_{max}
u_1
Minima of ψ < 0?
Y
Add minima to grid
BF Validation using Iterative Refinement
N
Done
Y
Validate $\psi \geq 0$ using SMT
N
Add counterexample to grid
SMT: Satisfiability Modulo Theory

Figure 7.1: Validating BFs using Satisfiability Modulo Theory.

represents a counterexample to $\psi \geq 0$. This input-pair is then used to augment the input-space discretization, and SOSP 1 is repeated. If the minima is non-negative, then Satisfiability Modulo Theory (SMT) is used to validate $\psi \geq 0$. In [45], authors use an ensemble of SMT-based tools to validate Lyapunov functions. A similar approach can be used to validate the candidate BFs. Any counterexample to the $\psi \geq 0$ claim is used to further refine the input-space quantization. Thus, the BF computation involves iterating between SOSP 1 and generating counterexamples. The process terminates when no further counterexamples are found; in other words, the candidate BF is validated.

Appendix A

Stability properties of voltage-controlled CTMCs for Ion Channels

In this chapter, we use compartmental systems theory [74] to state and prove stability properties of the constant-voltage ion channel models Σ_I^v and Σ_K^v. This in turn justifies the simulation strategy used in PEFT, i.e. finite-length simulations of Σ_I^v and Σ_K^v are sufficient to obtain the approximately equivalent HH-type abstractions Σ_{HI}^v and Σ_{HK}^v.

Theorem A.0.1. *Let $A_x(v)$, be the rate matrix of the corresponding dynamical system Σ_x, $x \in \{I, K\}$, as per Definitions 2.1.1 and 2.1.2. For all the values of the bounded input $v \in [V_{res}, V_{max}]$, $A_x(v)$ has exactly one eigenvalue that is 0 and the real part of all the other eigenvalues is negative.*

Proof. First, we will prove a lemma showing that the rate matrices $A_x(v)$ are *compartmental matrices* for $v \in [V_{res}, V_{max}]$.

Lemma A.0.2. *The rate matrix $A_x(v)$ is a compartmental matrix for $v \in [V_{res}, V_{max}]$.*

Proof. A square matrix $M \in \mathbb{R}^{n\times n}$ is called a compartmental system if it satisfies the following properties:

1. All the non-diagonal entries are greater than or equal to 0, i.e $M_{ij} \geq 0$ for $i = 1, \dots, n, j = 1, \dots, n, i \neq j$.

2. Sum of the entries along all the columns is less than or equal to 0, i.e. $\sum_{i=1}^{n} M_{ij} \leq 0$, $j = 1, \dots, n$.

The M_{ij} entry of the matrix is interpreted as the rate of flow of mass from the j^{th} compartment to the j^{th} compartment, $i \neq j$. The diagonal entry M_{ii} is the total rate of outflow from the i^{th} compartment.

For the rate matrices $A_x(v)$, the first property is satisfied as the non-diagonal entries on the i^{th} row of $A_x(v)$ represent incoming transfer rates for state i. These rates are positive as they are exponential functions of the input v. For Σ_I and Σ_K, these functions are listed in Tables 2.1 and 2.2 respectively.

Consider the j^{th} column of $A_x(V)$. The entry $A_{x,ij}$, $i \neq j$ denotes the transfer rate from state j to state i. The diagonal entry of this column $A_{x,jj}$ is the negated sum of all the outgoing rates from state j. Thus, the sum of every column is 0. □

Lemma A.0.3. *The rate matrix $A_x(v)$ is irreducible for $v \in [V_{res}, V_{max}]$.*

Proof. Irreducibility of $A_x(v)$ can be proved using a graph-theoretic argument. We construct a directed graph $G_x^v(W, E)$, where the set of vertices W corresponds to the states of M_x. The set of edges E is constructed as follows. An edge, e_{ij}, from vertex i to vertex j, $i, j = 1, \ldots, n$ exists if $A_{x,ij}(v) \neq 0$.

From linear algebra, we know that the matrix $A_x(v)$ is irreducible if and only if $G_x^v(V, E)$ is connected, i.e. there is a path between every pair of vertices.

If there is an edge from state i to state j of M_x, then the transfer rate $A_{x,ij}(v)$ does not become 0 for any value of v as it an exponential function of v. Also, for a given value of $v \in [V_{res}, V_{max}]$, the corresponding graph G_x^v always remains connected. Thus $A_x(v)$ is irreducible for all $v \in [V_{res}, V_{max}]$. □

Now we introduce the concept of a *trap* of a compartmental system. A trap is a compartment or a set of compartments from which there are no transfers or flows to the environment nor to the compartments that are not in that set. A formal definition is as follows. Let S be a linear compartmental system consisting of compartments $C_1, C_2, \ldots, C_n$. Let $T \subseteq S$, be a subset of the compartments. We number the compartments such that T consists of the compartments $C_m, C_{m+1}, \ldots, C_n$ for $m \leq n$. Let $F \in \mathbb{R}^{n \times n}$ be the rate matrix consistent with the new numbering. The subset T is a trap if and only if $F_{ij} = 0$ for (i, j) such that $j = m, m+1, \ldots, n$ and $i = 0, 1, \ldots, m-1$. A trap is said to be *simple* is it does not strictly contain any traps.

Lemma A.0.4. *The only trap in $A_x(v)$ is the set of all states.*

Proof. As $A_x(v)$ is irreducible, as per Lemma A.0.3, flow between any pair of compartments is nonzero. Thus the only trap is the set of all compartments, □

The proof of the Theorem A.0.1 now follows from Theorems 2.2.4 and 2.2.6 of [74]. Prerequisite conditions have been proved in Lemmas A.0.3 and A.0.4. □

Appendix B

Proofs of Theorems 4.3.2 and 4.3.3

A proof of Theorem 4.3.2 is as follows.

Proof. From Eq. 4.2, we have the first inequality. From Eq. 4.3, we have

$$\begin{aligned}\frac{dS(\mathbf{x}_1(t),\mathbf{x}_2(t))}{dt} &\leq -\lambda S(\mathbf{x}_1(t),\mathbf{x}_2(t)) + \gamma \parallel \mathbf{u}_1(t) - \mathbf{u}_2(t) \parallel \\ &\leq -\lambda S(\mathbf{x}_1(t),\mathbf{x}_2(t)) + \gamma \parallel \mathbf{u}_1 - \mathbf{u}_2 \parallel_\infty\end{aligned}$$

Let $\eta(t) = e^{-\lambda t} S(\mathbf{x}_1(0),\mathbf{x}_2(0)) + \frac{\lambda}{\gamma} \parallel \mathbf{u}_1 - \mathbf{u}_2 \parallel_\infty$. It is a solution of the differential equation $\dot{\eta}(t) = -\lambda\eta(t) + \gamma \parallel \mathbf{u}_1 - \mathbf{u}_2 \parallel_\infty$. Moreover, $S(\mathbf{x}_1(0),\mathbf{x}_2(0)) \leq \eta(0)$; then from the funnel theorem [42], it follows that $\forall t \geq 0, S(\mathbf{x}_1(t),\mathbf{x}_2(t)) \leq \eta(t)$.

□

A proof of Theorem 4.3.3 is as follows.

Proof. Consider S be the function as per the theorem. We will find conditions on α_1 and α_2 such that S is a BF between Σ_{A1} and Σ_{B2}. If $\alpha_1 \geq 1$ and $\alpha_2 \geq 1$, then

$$\begin{aligned}S(\mathbf{x}_{A1},\mathbf{x}_{B2}) &\geq S_{AB}(\mathbf{x}_A,\mathbf{x}_B) + S_{12}(\mathbf{x}_1,\mathbf{x}_2) \\ &\geq \parallel g_A(\mathbf{x}_A) - g_B(\mathbf{x}_B) \parallel + \parallel g_1(\mathbf{x}_1) - g_2(\mathbf{x}_2) \parallel,\end{aligned}$$

because S_{AB} and S_{12} satisfy Eq. 4.2. The output difference of Σ_{A1} and Σ_{B2}:

$$\begin{aligned}\parallel g_{A1}(\mathbf{x}_{A1}) - g_{B2}(\mathbf{x}_{B2}) \parallel &= \sqrt{\parallel g_A(\mathbf{x}_A) - g_B(\mathbf{x}_B) \parallel^2 + \parallel g_1(\mathbf{x}_1) - g_2(\mathbf{x}_2) \parallel^2} \\ &\leq \parallel g_A(\mathbf{x}_A) - g_B(\mathbf{x}_B) \parallel + \parallel g_1(\mathbf{x}_1) - g_2(\mathbf{x}_2) \parallel\end{aligned}$$

Therefore, it shows that S satisfies Eq. 4.2. Applying similar steps as in [25], we can write the following:

$$\begin{aligned}&\frac{\partial S}{\partial \mathbf{x}_{A1}} f_{A1}(\mathbf{x}_{A1}, \mathbf{u}_{A1}) + \frac{\partial S}{\partial \mathbf{x}_{B2}} f_{B2}(\mathbf{x}_{B2}, \mathbf{u}_{B2}) \\ &\leq -(\alpha_1 \lambda_{AB} - \alpha_2 \gamma_{12}) S_{AB}(\mathbf{x}_A, \mathbf{x}_B) - (\alpha_2 \lambda_{12} - \alpha_1 \gamma_{AB}) S_{12}(\mathbf{x}_1, \mathbf{x}_2)\end{aligned}$$

If $(\alpha_1 \lambda_{AB} - \alpha_2 \gamma_{12}) > 0$ and $(\alpha_2 \lambda_{12} - \alpha_1 \gamma_{AB}) > 0$ and
$\lambda = \min(\frac{(\alpha_1 \lambda_{AB} - \alpha_2 \gamma_{12})}{\alpha_1}, \frac{(\alpha_2 \lambda_{12} - \alpha_1 \gamma_{AB})}{\alpha_2})$, then

$$\frac{\partial S}{\partial \mathbf{x}_{A1}} f_{A1}(\mathbf{x}_{A1}, \mathbf{u}_{A1}) + \frac{\partial S}{\partial \mathbf{x}_{B2}} f_{B2}(\mathbf{x}_{B2}, \mathbf{u}_{B2}) \leq -\lambda S(\mathbf{x}_{A1}, \mathbf{x}_{B2}).$$

Therefore, S will be a BF if $\alpha_1 \geq 1$, $\alpha_2 \geq 1$, $(\alpha_1 \lambda_{AB} - \alpha_2 \gamma_{12}) > 0$ and $(\alpha_2 \lambda_{12} - \alpha_1 \gamma_{AB}) > 0$. As shown in [25], these four conditions can be expressed as $\frac{\lambda_{AB} \lambda_{12}}{\gamma_{AB} \gamma_{12}} < 1$.
$\square$

Bibliography

[1] A. Alessio and A. Bemporad. A survey on explicit model predictive control. *Nonlinear Model Predictive Control, Lecture Notes in Control and Information Sciences*, 384:345–369, 2009.

[2] D. Barkley. A model for fast computer simulation of waves in excitable media. *Physica*, 49(D):61–70, August 1991.

[3] E. Bartocci, E. M. Cherry, J. Glimm, R. Grosu, S. A. Smolka, and F. H. Fenton. Toward real-time simulation of cardiac dynamics. In *Proceedings of the 9th International Conference on Computational Methods in Systems Biology*, CMSB '11, pages 103–112. ACM, 2011.

[4] E. Bartocci, R. Singh, F. B. von Stein, A. Amedome, A. J. J. Caceres, J. Castillo, E. Closser, G. Deards, A. Goltsev, R. S. Ines, C. Isbilir, J. K. Marc, D. Moore, D. Pardi, S. Sadhu, S. Sanchez, P. Sharma, A. Singh, J. Rogers, A. Wolinetz, T. Grosso-Applewhite, K. Zhao, A. B. Filipski, R. F. Gilmour, R. Grosu, J. Glimm, S. A. Smolka, E. M. Cherry, E. M. Clarke, N. Griffeth, and F. H. Fenton. Teaching cardiac electrophysiology modeling to undergraduate students: laboratory exercises and GPU programming for the study of arrhythmias and spiral wave dynamics. *Advances of Physiology Education*, 35(4):427–437, December 2011.

[5] G. W. Beeler and H. Reuter. Reconstruction of the action potential of ventricular myocardial fibres. *Journal of Physiology*, 268(1):177–210, June 1977.

[6] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407 – 427, 1999.

[7] B.S. Chen and C.W. Li. Robust observer-based tracking control of hodgkin-huxley neuron systems under environmental disturbances. *Neural computation*, 22(12):3143–3178, 2010.

[8] A. Bueno-Orovio, E. M. Cherry, and F. H. Fenton. Minimal model for human ventricular action potentials in tissue. *Journal of Theoretical Biology*, 253(3):544–560, Sept. 2008.

[9] H. Chang, A. Astolfi, and H. Shim. A control theoretic approach to venom immunotherapy with state jumps. In *Proceedings of the International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 742–745, 2010.

[10] T. Chen, N. F. Kirkby, and R. Jena. Optimal dosing of cancer chemotherapy using model predictive control and moving horizon state/parameter estimation. *Computer Methods and Programs in Biomedicine*, 108(3):973 – 983, 2012.

[11] E. M. Cherry and F. H. Fenton. A tale of two dogs: Analyzing two models of canine ventricular electrophysiology. *American Journal of Physiology - Heart and Circulatory Physiology*, 292:H43–H55, 2007.

[12] E. M. Clarke Jr., O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.

[13] P. Cousot and R. Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proceedings of the Fourth ACM Symposium on Principles of Programming Languages*, POPL'77, pages 238–252. ACM, 1977.

[14] T. Dang, C. L. Guernic, and O. Maler. Computing reachable states for nonlinear biological models. *Theoretical Computer Science*, 412(21):2095–2107, March 2011.

[15] Denis Noble. Modeling the heart - from genes to cells to the whole organ. *Science*, 295(5560):1678–1682, March 2002.

[16] Denis Noble. Modeling the heart. *Physiology*, 19:191–197, August 2004.

[17] Denis Noble. From the Hodgkin-Huxley axon to the virtual heart. *Journal of Physiology*, 580(1):15–22, April 2007.

[18] V. G. Fast and A. Kléber. Role of wavefront curvature in propagation of cardiac impulse. *Journal of Cardiovascular Research*, 33(2):258–271, February 1997.

[19] F. Fenton and A. Karma. Vortex dynamics in three-dimensional continuous myocardium with fiber rotation: Filament instability and fibrillation. *Chaos*, 8(1):20–47, 1998.

[20] F. H. Fenton and E. M. Cherry. Models of cardiac cell. *Scholarpedia*, 3:1868, 2008.

[21] R. Findeisen and F. Allgöwer. An introduction to nonlinear model predictive control. In *Proceedings of the 21st Benelux Meeting on Systems and Control, Veidhoven*, pages 1–23, 2002.

[22] J. Fisher, N. Piterman, and M. Y. Vardi. The only way is up. In *Proceedings of the 17th International Conference on Formal methods*, FM'11, pages 3–11, Berlin, Heidelberg, 2011. Springer-Verlag.

[23] S. N. Flaim, W. R. Giles, and A. D. McCulloch. Contributions of sustained I_{Na} and I_{Kv43} to transmural heterogeneity of early repolarization and arrhythmogenesis in canine left ventricular myocytes. *American Journal of Physiology - Heart and Circulatory Physiology*, 291:H2617–H2629, 2006.

[24] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practice - A survey. *Automatica*, 25(3):335–348, 1989.

[25] A. Girard. A composition theorem for bisimulation functions. *Pre-print*, 2007. arXiv:1304.5153.

[26] A. Girard and G. J. Pappas. Approximate bisimulations for nonlinear dynamical systems. In *Proceedings of 44th IEEE Conference on Decision and Control*, Serville, Spain, December 2005.

[27] A. Girard and G. J. Pappas. Approximate bisimulation relations for constrained linear systems. *Automatica*, 43(8):1307 – 1317, August 2007.

[28] A. Girard and G. J. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, 52(5):782 –798, May 2007.

[29] R. Grosu, G. Batt, F. Fenton, J. Glimm, C. L. Guernic, S. Smolka, and E. Bartocci. From cardiac cells to genetic regulatory networks. In *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV 2011)*, LNCS, Cliff Lodge, Snowbird, Utah, USA, July 2011. Springer.

[30] W. Heemels, B. D. Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085 – 1091, 2001.

[31] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

[32] M. Huth and M. Ryan. *Logic in Computer Science, Modeling and Reasoning about Systems.* Cambridge University Press, 2004.

[33] M. A. Islam. Supplement to compositionality results for cardiac cell dynamics. `www.cs.sunysb.edu/~amurthy/hscc14_supp.htm`, also available on `arxiv.org`, 2013.

[34] M. A. Islam, A. Murthy, E. Bartocci, E. M. Cherry, F. H. Fenton, J. Glimm, S. A. Smolka, and R. Grosu. Model-order reduction of ion channel dynamics using approximate bisimulation. *Theoretical Computer Science*, 2014.

[35] M. A. Islam, A. Murthy, A. Girard, S. A. Smolka, and R. Grosu. Compositionality results for cardiac cell dynamics. In *Proceedings of HSCC'14, the 17th International Conference on Hybrid Systems: Computation and Control*, pages 243–252, New York, NY, USA, 2014. ACM.

[36] V. Iyer, R. Mazhari, and R. L. Winslow. A computational model of the human left-ventricular epicardial myocytes. *Biophysical Journal*, 87(3):1507–1525, 2004.

[37] E. M. Izhikevich. *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting.* The MIT Press, 2007.

[38] J. G. Dumas and A. Rondepierre. Modeling the electrical activity of a neuron by a continuous and piecewise affine hybrid system. In *Proceedings of the 6th international conference on Hybrid systems: computation and control*, pages 156–171, Berlin, Heidelberg, 2003. Springer-Verlag.

[39] J. Uhlendorf et al. Long-term model predictive control of gene expression at the population and single-cell levels. *Proceedings of the National Academy of Sciences*, 109(35):14271–14276, 2012.

[40] Jasmin Fisher and David Harel and Thomas A. Henzinger. Biology as reactivity. *Communications of the ACM*, 54(10):72–82, October 2011.

[41] Jeffry A. Florian Jr., J. L. Eiseman, and R. S. Parker. Nonlinear model predictive control for dosing daily anticancer agents using a novel saturating-rate cell-cycle model. *Computers in Biology and Medicine*, 38(3):339 – 347, 2008.

[42] J.H. Hubbard and B.H. West. *Differential Equations: A Dynamical Systems Approach.* Springer, 1991.

[43] A. A. Julius, A. D'Innocenzo, M. D. D. Benedetto, and G. J. Pappas. Approximate equivalence and synchronization of metric transition systems. *Systems and Control Letters*, 58(2):94 – 101, 2009.

[44] K. N. Fountas et al. Implantation of a closed-loop stimulation in the management of medically refractory focal epilepsy, a technical note. *Stereotactic and Functional Neurosurgery*, 83(4):153–158, 2005.

[45] J. Kapinski, J. V. Deshmukh, S. Sankaranarayanan, and N. Arechiga. Simulation-guided Lyapunov analysis for hybrid dynamical systems. In *Proceedings of HSCC'14, the 17th International Conference on Hybrid Systems: Computation and Control*, pages 133–142, New York, NY, USA, 2014. ACM.

[46] M. Kay and R. Gray. Measuring curvature and velocity vector fields for waves of cardiac excitation in 2-d media. *IEEE Transactions on Biomedical Engineering*, 52(1):50–63, January 2005.

[47] J. Keener. Invariant manifold reductions for markovian ion channel dynamics. *Journal of Mathematical Biology*, 58(3):447–457, July 2009.

[48] D. M. Khan. Cubic Bézier least square fitting. *Matlab File Exchange*, July 2009.

[49] A. Kléber. The fibrillating atrial myocardium. what can the detection of wave breaks tell us? *Journal of cardiovascular research*, 48(2):181–184, August 2000.

[50] S. O. Krumke. Nonlinear optimization. Lecture Notes, 2004.

[51] D. Lloyd-Jones, R. J. Adams, T. M. Brown, M. Carnethon, S. Dai, G. D. Simone, T. B. Ferguson, E. Ford, K. Furie, C. Gillespie, A. Go, K. Greenlund, N. Haase, S. Hailpern, P. M. Ho, V. Howard, B. Kissela, S. Kittner, D. Lackland, L. Lisabeth, A. Marelli, M. M. McDermott, J. Meigs, D. Mozaffarian, M. Mussolino, G. Nichol, V. L. Roger, W. Rosamond, R. Sacco, P. Sorlie, R. Stafford, T. Thom, S. Wasserthiel-Smoller, N. D. Wong, and J. Wylie-Rosett. Heart disease and stroke statistics 2010 update: A report from the American Heart Association. *Circulation*, December 2009.

[52] C. H. Luo and Y. Rudy. A dynamic model of the cardiac ventricular action potential. I. Simulations of ionic currents and concentration changes. *Circulation Research*, 74(6):1071–1096, June 1994.

[53] Multi-Parametric Toolbox (MPT), 2004. URL: `http://control.ee.ethz.ch/~mpt` [accessed: 2014-03-25].

[54] MATLAB. Algorithm *isocontour*. `http://www.mathworks.com/matlabcentral/fileexchange/30525-isocontour`.

[55] MATLAB. Symbolic math toolbox. `http://www.mathworks.com/help/toolbox/symbolic`.

[56] MATLAB Optimization Toolbox, 2014. URL: `http://www.mathworks.com/help/toolbox/optim` [accessed: 2014-03-25].

[57] MATLAB ODE45 solver. *Version 7.10.0 (R2010a).* The MathWorks Inc., Natick, Massachusetts, 2010.

[58] MATLAB Open curve fitting toolbox (cftool). *Version 7.10.0 (R2010a).* The MathWorks Inc., Natick, Massachusetts, 2010.

[59] S. Mohammed, P. Poignet, P. Fraisse, and D. Guiraud. Toward lower limbs movement restoration with input-output feedback linearization and model predictive control through functional electrical stimulation. *Control Engineering Practice*, 20(2):182 – 195, 2012.

[60] S. L. Murphy, J. Xu, and K. D. Kochanek. Deaths: Final data for 2010. *National Vital Statistics Report*, 61(4), 2013.

[61] A. Murthy, Md. A. Islam, E. Bartocci, E. Cherry, F. H. Fenton, J. Glimm, S. A. Smolka, and R. Grosu. Approximate bisimulations for sodium channel dynamics. In *Proceedings of CMSB'12, the 10th Conference on Computational Methods in Systems Biology*, LNCS, London, U.K., October 2012. Springer.

[62] R. Naderi, M. J. Yazdanpanah, A. Azemi, and B. Roaia. Tracking normal action potential based on the FHN model using adaptive feedback linearization technique. In *Proceedings of the IEEE International Conference on Control Applications (CCA)*, pages 1458–1463, 2010.

[63] S. L. Noble, E. Sherer, R. E. Hannemann, D. Ramkrishna, T. Vik, and A. E. Rundell. Using adaptive model predictive control to customize maintenance therapy chemotherapeutic dosing for childhood acute lymphoblastic leukemia. *Journal of Theoretical Biology*, 264(3):990 – 1002, 2010.

[64] Paul A. Heidenreich, Justin G. Trogdon, Olga A. Khavjou, Javed Butler, Kathleen Dracup, Michael D. Ezekowitz, Eric Andrew Finkelstein, Yuling Hong, S. Claiborne Johnston, Amit Khera, Donald M. Lloyd-Jones, Sue A. Nelson, Graham Nichol, Diane Orenstein, Peter W.F. Wilson, and Y. Joseph Woo. Forecasting the future of cardiovascular disease in the United States: a policy statement from the American Heart Association. *Circulation*, 123:933–44, 2011.

[65] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, 2004.

[66] L. Priebe and D. J. Beuckelmann. Simulation study of cellular electric properties in heart failure. *Circulation Research*, 82:1206–1223, 1998.

[67] S. J. Qin and T. A. Badgwell. An overview of nonlinear model predictive control applications. In *Nonlinear Predictive Control*, pages 369–392. Verlag, 2000.

[68] D. F. Rogers and J. A. Adams. *Mathematical Elements of Computer Graphics*. McGraw-Hill Science/Engineering/Math, New York, 1989.

[69] S. Luther et al. Low-energy control of electrical turbulence in the heart. *Nature*, 475:235–239, 2011.

[70] S. H. Strogatz. *Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry, And Engineering (Studies in Nonlinearity)*. Westview Press, 2001.

[71] S. Summers, D. M. Raimondo, C. N. Jones, J. Lygeros, and M. Morari. Fast explicit nonlinear model predictive control via multiresolution function approximation with guaranteed stability. In *8th IFAC Symposium on Nonlinear Control Systems*, pages 533–538, 2010.

[72] K. H. ten Tusscher, D. Noble, P. J. Noble, and A. V. Panfilov. A model for human ventricular tissue. *American Journal of Physiology*, 286:H1573–H1589, 2004.

[73] J. Uhlendorf, P. Hersen, and G. Batt. Towards real-time control of gene expression: *in silico* analysis. In *Proceedings of the IFAC World Congress*, volume 18, pages 14844–14850, 2011.

[74] J. van den Hof. *System theory and system identification of compartmental systems*. PhD thesis, University of Groningen, Netherlands, November 1996.

[75] A. T. Winfree. Heart muscle as a reaction - diffusion medium: The roles of electric potential diffusion, activation front curvature, and anisotropy. *International Journal of Bifurcation and Chaos*, 7(3):487–526, March 1997.