APPROVAL SHEET

Title of Thesis: Efficiency Improvements in Numerical Methods for Studying Connectivity in a Model of a Pancreatic Islet of Heterogeneous Beta Cells

Name of Candidate: Samuel Khuvis
Master of Science, 2013

Thesis and Abstract Approved: ____________________________________
Dr. Matthias K. Gobbert
Professor
Department of Mathematics and Statistics

Date Approved: ______________________________

CURRICULUM VITAE

Name: Samuel Khuvis.

Permanent Address: 47 Ironwood Circle
Baltimore, MD, 21209.

Degree and Date to be Conferred: Master of Science, 2013.

Date of Birth: September 18, 1991.

Place of Birth: Chernivtsi, Ukraine.

Collegiate Institutions Attended:

- University of Maryland, Baltimore County, 2007–2011, B.S. Mathematics May 2011.
- University of Maryland, Baltimore County, 2011–2013, M.S. Applied Mathematics May 2013.

Courses graded:

- Math 430 Matrix Analysis, Fall 2011
- Math 385 Mathematical Modelling, Spring 2012
- Math 302/600 Real Analysis, Spring 2012
- Math 385 Mathematical Modelling, Fall 2012
- Math 404 Partial Differential Equations, Fall 2012
- Math 385 Mathematical Modelling, Spring 2013
- Math 430/603 Matrix Analysis, Spring 2013

ABSTRACT

Title of Thesis: Efficiency Improvements in Numerical Methods for
Studying Connectivity in a Model of a
Pancreatic Islet of Heterogeneous Beta Cells

Samuel Khuvis, Master of Science, 2013

Thesis directed by: Dr. Matthias K. Gobbert, Professor
Department of Mathematics and Statistics
University of Maryland, Baltimore County

Diabetes is a disease characterized by a high concentration of glucose in a person's blood stream caused by either an insulin deficiency or insulin resistance. Insulin is a hormone produced in the pancreas that regulates the concentration of glucose in the blood stream. The endocrine system of the pancreas contains clusters of cells called islets of Langerhans. The most common type of cell in an islet of Langerhans is the beta-cell, which is responsible for the production of insulin. Accurately modeling the metabolic and electrical activities of beta-cells would be invaluable in better understanding diabetes.

One beta-cell model is given by a system of seven coupled ordinary differential equations (ODEs) which models the metabolic and electrical dynamics. In our simulation we model the dynamics for a computational islet represented by a three-dimensional cube of beta-cells that are electrically coupled through complexes of proteins called gap junctions.

Cells of different bursting rates exist in a single islet with no known pattern of distribution. As a physiological example, we present an investigation of five different distributions of slow and fast bursting cells, called islet structure. We introduce a measure of connectivity based on the fraction of neighboring cells with speed different from the cell's own speed. This quantitative measure of connectivity provides a rational ordering of the different islet structures under consideration. For each islet structure, we run simulations for a range of coupling strength and find that the burst

period initially increases as coupling strength increases before eventually decreasing. We observe that as connectivity increases the coupling strength at which the burst period is the greatest decreases. Interestingly, for a range of connectivity the maximum burst period exceeds that for an uncoupled slow bursting cell.

The investigation of different islet structures over a range of coupling strengths is a prototypical example of a parameter study that require large numbers of runs of the underlying simulation code. We use Matlab code that implements the seven variable beta-cell model on a computational islet. This code uses Matlab's `ode15s` function that implements the numerical differentiation formulas, a well-respected and efficient ODE method for stiff initial value problems. The focus of the research is on demonstrating that several orders of magnitude improvement in runtimes for simulations of this type are possible that will enable a scale of parameter studies that were not possible before. We use a three species model as stand-in for this purpose, since it shares dynamic features with the full seven species model and allows us to show the key ingredient of a sparse storage of the Jacobian matrix most clearly.

EFFICIENCY IMPROVEMENTS IN NUMERICAL METHODS FOR STUDYING CONNECTIVITY IN A MODEL OF A PANCREATIC ISLET OF HETEROGENEOUS BETA CELLS

by

Samuel Khuvis

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
2013

© Copyright Samuel Khuvis 2013

To my family

ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. Matthias K. Gobbert, for his guidance and encouragement while working on this research project. I would also like to thank him for clarifying various concepts in numerical methods. I am also grateful to Dr. Bradford E. Peercy, in particular for his help in explaining the biological processes and models relevant to this research project.

The hardware used in the computational studies is part of the UMBC High Performance Computing Facility (HPCF). The facility is supported by the U.S. National Science Foundation through the MRI program (grant nos. CNS–0821258 and CNS–1228778) and the SCREMS program (grant no. DMS–0821311), with additional substantial support from the University of Maryland, Baltimore County (UMBC). See `www.umbc.edu/hpcf` for more information on HPCF and the projects using its resources.

TABLE OF CONTENTS

Page

LIST OF TABLES vi

LIST OF FIGURES viii

CHAPTER

1 INTRODUCTION 1

1.1 Background 1

1.2 Numerical Results 6

2 MODEL 12

2.1 Physiological Background 12

2.2 Properties of Both Models 14

2.3 Mathematical Properties of the Seven Variable Model Equations . . 19

2.4 Mathematical Properties of the Three Variable Model Equations . . 24

3 NUMERICAL METHOD 29

3.1 Original ode15s with Numerical Jacobian 33

3.2 Memory Modified ode15s with Numerical Jacobian 33

3.3 Memory Modified ode15s with Dense Jacobian 34

3.4 Memory Modified ode15s with Sparse Jacobian 36

3.5 Explicit ode45 37

3.6 Implicit ode113 37

4 RESULTS 39

4.1 Physiological Results 39

4.1.1 Seven Variable Model 39

4.1.2 Three Variable Model 49

4.2 Numerical Performance Results 58

4.2.1 Original ode15s with Numerical Jacobian 59

4.2.2 Memory Modified Ode15s with Numerical Jacobian 63

4.2.3 Memory Modified Ode15s with Dense Jacobian 67

4.2.4 Memory Modified Ode15s with Sparse Jacobian 71

4.2.5 Ode45 77

4.2.6 Ode113 80

5 CONCLUSIONS 83

BIBLIOGRAPHY 85

LIST OF TABLES

TABLE Page

1.2.1 Runtimes for the case of no coupling strength 9

1.2.2 Runtimes for equally distributed islet structure 10

2.3.1 Constants and variables for the seven variable model, where (f) or (s) denote which parameter value goes with a fast or slow cell, respectively. 23

2.4.1 Constants and variables for the three variable model. 28

4.2.1 Performance results for uncoupled cells using original ode15s with numerical Jacobian. 60

4.2.2 Runtimes for original ode15s with numerical Jacobian. 61

4.2.3 Performance results for islet with equally distributed structure using original ode15s with numerical Jacobian. 62

4.2.4 Performance results for uncoupled cells using memory modified ode15s with numerical Jacobian. 64

4.2.5 Runtimes for memory modified ode15s with numerical Jacobian. 65

4.2.6 Performance results for islet with equally distributed structure using memory modified ode15s with numerical Jacobian. 66

4.2.7 Performance results for uncoupled cells using memory modified ode15s with dense Jacobian. 68

4.2.8 Runtimes for memory modified ode15s and dense Jacobian. 69

4.2.9 Performance results for islet with equally distributed structure using memory modified ode15s with dense Jacobian. 70

4.2.10 Performance results for uncoupled cells using memory modified ode15s with sparse Jacobian. 74

4.2.11 Runtimes for memory modified ode15s and sparse Jacobian. 75

4.2.12 Performance results for islet with equally distributed structure using memory modified ode15s with sparse Jacobian. 76
4.2.13 Performance study for uncoupled cells using ode45. 78
4.2.14 Performance study for islet with equally distributed structure using ode45. 79
4.2.15 Performance study for uncoupled cells using ode113. 81
4.2.16 Performance study for islet with equally distributed structure using ode113. 82

LIST OF FIGURES

FIGURE Page

1.1.1 Peak coupling strength vs. connectivity for the $5 \times 5 \times 5$ islet for the (a) seven variable model and (b) three variable model. 11

2.1.1 Schematic of two electrically coupled β-cells. 13

2.2.1 Structure for a $5 \times 5 \times 5$ islet. 16

4.1.1 Bursting behavior of uncoupled cells for the seven variable model. . . . 42

4.1.2 Bursting behavior of cells in an islet with grouped structure ($c = 0.0928$) for the seven variable model. 43

4.1.3 Bursting behavior of cells in an islet with split grouped structure ($c = 0.1712$) for the seven variable model. 44

4.1.4 Bursting behavior of cells in an islet with layered middle structure ($c = 0.2693$) for the seven variable model. 45

4.1.5 Bursting behavior of cells in an islet with layered split structure ($c = 0.4261$) for the seven variable model. 46

4.1.6 Bursting behavior of cells in an islet with equally distributed structure ($c = 1$) for the seven variable model. 47

4.1.7 Average burst period over a range of coupling strengths for cells for the seven variable model. 48

4.1.8 Bursting behavior of uncoupled cells for the three variable model. . . . 51

4.1.9 Bursting behavior of cells in an islet with grouped structure ($c = 0.0928$) for the three variable model. 52

4.1.10 Bursting behavior of cells in an islet with split grouped structure ($c = 0.1712$) for the three variable model. 53

4.1.11 Bursting behavior of cells in an islet with layered middle structure ($c = 0.2693$) for the three variable model. 54

4.1.12 Bursting behavior of cells in an islet with layered split structure ($c = 0.4261$) for the three variable model. 55

4.1.13 Bursting behavior of cells in an islet with equally distributed structure ($c = 1$) for the three variable model. 56

4.1.14 Average burst period over a range of coupling strengths for cells for the three variable model. 57

4.2.1 Numerical performance data for uncoupled cells in a $10 \times 10 \times 10$ computational islet using original ode15s with numerical Jacobian: (a) time step Δt vs. t, (b) method order k vs. t. 60

4.2.2 Numerical performance data for uncoupled cells in a $10 \times 10 \times 10$ computational islet using memory modified ode15s with numerical Jacobian: (a) time step Δt vs. t, (b) method order k vs. t. 64

4.2.3 Numerical performance data for uncoupled cells in a $10 \times 10 \times 10$ computational islet using memory modified ode15s with dense Jacobian: (a) time step Δt vs. t, (b) method order k vs. t. 68

4.2.4 Numerical performance data for uncoupled cells in a $10 \times 10 \times 10$ computational islet using memory modified ode15s with sparse Jacobian: (a) time step Δt vs. t, (b) method order k vs. t. 73

4.2.5 Numerical performance data for uncoupled cells in a $10 \times 10 \times 10$ computational islet using memory modified ode15s with dense and sparse Jacobian: (a) time step Δt vs. t, (b) method order k vs. t. 73

CHAPTER 1

INTRODUCTION

1.1 Background

Diabetes mellitus is a group of diseases characterized by a high concentration of glucose in a person's blood stream. The concentration of glucose in the blood is regulated by insulin, a hormone produced by cells in the pancreas. So, if the concentration of glucose in the blood stream is too high, it is caused mainly by an insulin deficiency or an insulin resistance which means that insulin does not properly interact with cells. Type 1 Diabetes corresponds to an insulin deficiency, while Type 2 diabetes is caused by either an insulin resistance or insulin deficiency. With statistics from January 2011 showing that 23.6 million people in the United States suffer from diabetes, being able to model the cells which play a large role in diabetes would be very valuable (Centers for Disease Control and Prevention [8]).

The endocrine system of the pancreas contains clusters of cells called islets of Langerhans, which consist of four different types of cells [10]. The most common type of cell is the β-cell, which is responsible for the secretion of insulin.

One β-cell model using a system of seven coupled ordinary differential equations (ODEs) which models the metabolic and electrical dynamics. The metabolic model was developed by Smolen [16]. It was then coupled by Bertram and Sherman [2] with Sherman's update [14] of the model developed by Chay and Keizer [4] and includes voltage data collected by Rorsman and Trube [11]. We apply this model on a computational islet, represented by a three-dimensional $N \times N \times N$ cube of β-cells. These cells are electrically coupled through complexes of proteins called gap junctions. The dynamics of the voltage V, the fraction of open K^+ channels n, and five chemical species are modeled by

$$
\begin{aligned}
\frac{dV}{dt} &= \frac{-(I_{\mathrm{K}} + I_{\mathrm{Ca}} + I_{\mathrm{K\,(Ca)}} + I_{\mathrm{K\,(ATP)}})}{C_m},\\
\frac{dn}{dt} &= \frac{(n_\infty - n)}{\tau_n},\\
\frac{d[\mathrm{Ca}]}{dt} &= f_{\mathrm{cyt}}\,(J_{\mathrm{mem}} + J_{\mathrm{er}}),\\
\frac{d[\mathrm{Ca_{er}}]}{dt} &= -\sigma_V\, f_{\mathrm{er}}\, J_{\mathrm{er}},\\
\frac{d[\mathrm{ADP}]}{dt} &= \frac{[\mathrm{ATP}] - [\mathrm{ADP}]\ \exp\left\{(r+\gamma)\left(1 - \frac{[\mathrm{Ca}]}{r_1}\right)\right\}}{\tau_a},\\
\frac{d[\mathrm{G6P}]}{dt} &= \kappa\,(R_{\mathrm{GK}} - R_{\mathrm{PFK}}),\\
\frac{d[\mathrm{FBP}]}{dt} &= \kappa\,(R_{\mathrm{PFK}} - 0.5\,R_{\mathrm{GPDH}}),
\end{aligned}
$$

where [XX] denotes the concentration of species XX. Let

$$y = (V, n, [\mathrm{Ca}], [\mathrm{Ca_{er}}], [\mathrm{ADP}], [\mathrm{G6P}], [\mathrm{FBP}])^T,$$

where each of V, n, [Ca], [Ca$_{\mathrm{er}}$], [ADP], [G6P], and [FBP] are $N^3 \times 1$ vectors containing the values of the corresponding variable for each of the N^3 cells in the computational islet. The vector relating $f(t, y)$, the right-hand side of the system, is similarly given by

$$
f(t,y) = \begin{bmatrix} f_V(t,y)\\ f_n(t,y)\\ f_{[\mathrm{Ca}]}(t,y)\\ f_{[\mathrm{Ca_{er}}]}(t,y)\\ f_{[\mathrm{ADP}]}(t,y)\\ f_{[\mathrm{G6P}]}(t,y)\\ f_{[\mathrm{FBP}]}(t,y) \end{bmatrix}.
$$

A matrix G contains the coupling strengths between cells. So, the matrix form of the system of ODEs is given by

$$\frac{dy}{dt} = f(t, y) + Gy.$$

The focus of my research was on demonstrating several orders of magnitude improvement in runtimes for simulations of this type that will enable parameter studies that were simply not possible before. We used a three variable model as stand-in for this purpose, since it shares the dynamic features with the full model and allows us to show the key ingredient of a Jacobian matrix and its efficient implementation very clearly.

The system of ODEs for the three variable model is given by

$$\frac{dV}{dt} = \frac{-(I_{\mathrm{K}} + I_{\mathrm{Ca}} + I_s)}{C_m}, \tag{1.1.1}$$

$$\frac{dn}{dt} = \frac{\lambda(n_\infty(V) - n)}{\tau}, \tag{1.1.2}$$

$$\frac{ds}{dt} = \frac{s_\infty(V) + \beta - s}{\tau_s}, \tag{1.1.3}$$

where

$$\nu_\infty(V) = \frac{1}{1 + \exp[\frac{V_\nu - V}{\theta_\nu}]} \text{ for } \nu = m, n, s. \tag{1.1.4}$$

As in the seven variable model, the matrix form of the ODEs is given by the equation

$$\frac{dy}{dt} = f(t, y) + Gy. \tag{1.1.5}$$

Numerical methods appropriate for this stiff system of ODEs will necessarily need the Jacobian with respect to the unknown vector y of the right-hand side of the ODEs

$$\nabla_y(f(t, y) + Gy) = J + G = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} + G, \tag{1.1.6}$$

where J has a 3×3 block structure with

$$\begin{aligned}
J_{11} &= \operatorname{diag}\Big(-\frac{\hat{g}_{\mathrm{K}}}{C_m}\, n - \frac{\hat{g}_{\mathrm{Ca}}}{C_m}\, m_\infty(V) - \frac{\hat{g}_{\mathrm{Ca}}}{C_m}\, (V - V_{\mathrm{Ca}})\, m'_\infty(V) - \frac{\hat{g}_{\mathrm{s}}}{C_m}\, s \Big), \\
J_{12} &= \operatorname{diag}\Big(\frac{-\hat{g}_{\mathrm{K}}}{C_m} (V - V_{\mathrm{K}}) \Big), \\
J_{13} &= \operatorname{diag}\Big(\frac{\hat{g}_{\mathrm{s}}}{C_m} (V - V_{\mathrm{K}}) \Big), \\
J_{21} &= \operatorname{diag}\Big(\frac{\lambda}{\tau} n'_\infty(V) \Big), \\
J_{22} &= -\frac{\lambda}{\tau} I_{N^3 \times N^3}, \\
J_{23} &= 0_{N^3 \times N^3}, \\
J_{31} &= \frac{1}{\tau_s} s'_\infty(V) I_{N^3 \times N^3}, \\
J_{32} &= 0_{N^3 \times N^3}, \\
J_{33} &= -\frac{1}{\tau_s} I_{N^3 \times N^3}
\end{aligned}$$

with

$$\nu'_\infty(V) = \frac{\exp\left(\frac{V_\nu - V}{\theta_\nu}\right)}{\theta_\nu \left[1 + \exp\left(\frac{V_\nu - V}{\theta_\nu}\right)\right]^2}. \tag{1.1.7}$$

Each $N^3 \times N^3$ block of J is either a diagonal matrix or the zero matrix. Here, $0_{N^3 \times N^3}$ denotes an $N^3 \times N^3$ matrix in which all elements are zeros, and the identity $I_{N^3 \times N^3}$ is an $N^3 \times N^3$ matrix in which the diagonal elements are ones and all other elements are zeros. Some blocks of J are diagonal matrices with the same constant in all diagonal elements, represented above by blocks involving the product of a constant with $I_{N^3 \times N^3}$. Other blocks have different values in the diagonal elements, indicated by the notation $\operatorname{diag}(v)$, which constructs a diagonal matrix with the components of the $N^3 \times 1$ vector v along the diagonal.

In [5], existing code was used to simulate the seven variable model. This code has been modified to simulate the three variable model and also used to test the effect of distribution of slow and fast cells on the behavior. The cell behavior and burst period for β-cells were simulated for five different spatial distributions of slow and fast cells.

Each islet contains β-cells with varying burst rates. The distribution of these cells is not known so we are interested in the effect that the distribution of slow bursting and fast bursting cells has on the behavior of cells in the islet.

The five distributions are:

- The grouped structure has a block of $\frac{N^3}{2}$ fast cells and a block of $\frac{N^3}{2}$ slow cells. It has a connectivity of 0.0928.
- The split grouped structure has two non-adjacent blocks of $\frac{N^2}{4}$ fast cells and two non-adjacent blocks of $\frac{N^3}{4}$ slow cells. It has a connectivity of 0.1712.
- The layered middle structure has N alternating layers of N^2 slow and then N^2 fast cells. However, the middle layer is $\frac{N^2}{2}$ slow cells and $\frac{N^2}{2}$ fast cells. It has a connectivity of 0.2693.
- The layered split structure has N alternating layers where a layer with the first $\frac{N^2}{2}$ cells are slow the last $\frac{N^2}{2}$ cells are fast is followed by a layer with the first $\frac{N^2}{2}$ cells are fast and the last $\frac{N^2}{2}$ cells are are slow and vice versa. It has a connectivity of 0.4261.
- In the equally distributed structure, all neighbors of fast cells are slow cells and all neighbors of slow cells are fast cells. It has a connectivity of 1.0000.

Each islet structure has a connectivity associated with the distribution of slow and fast cells and computed based on the average number of different cells connected to each cell in the islet. For instance, the equally distributed structure, has a connectivity of 1 since each slow cell is only connected to fast cells and each fast cell is only connected to slow cells.

We introduce a measure of connectivity based on the fraction of neighboring cells with speed different from the cell's own speed. This quantitative measure of connectivity provides a rational ordering of the different islet structures under consideration. The following was used to compute connectivity

$$c = \frac{1}{N^3} \sum_{i=1}^{N^3} \frac{d_i}{v_i}, \tag{1.1.8}$$

where

$$d_i = \begin{cases} \text{number of slow cells connected to cell } i \text{ if cell } i \text{ is fast,} \\ \text{number of fast cells connected to cell } i \text{ if cell } i \text{ is slow,} \end{cases}$$

and v_i is the total number of cells connected to i.

We found in our simulations that for any given distributed structure, as the coupling strength between cells in an islet changes, the burst period of each cell changes. We also observe that there is an initial increase in burst period as coupling strength is increased followed by a decrease in burst period. Figure 1.1.1(a) contains a plot of the coupling strength at which the highest burst period is reached vs. connectivity of the islet structure for the seven variable model. Figure 1.1.1(b) contains the corresponding plot for the three variable model. Figure 1.1.1 demonstrates that both models exhibit the same behavior of decreasing peak coupling strength as connectivity increases. Since the simpler three variable model has analogous behavior, it is a suitable stand-in for this class of models in the study of the numerical methods.

1.2 Numerical Results

The ODE systems in Section 1.1 are solved using the Numerical Differentiation Formulas NDFk from [13], as detailed in Chapter 3. The NDFk are a family of state-of-the art methods with a rigorous error estimator and automatic selection of method order k and time step Δt, specifically designed to solve stiff systems of ODEs. Stiff ODE solvers necessitate the use of the Newton method for the system of non-linear equations at every time step, which is turn requires the Jacobian matrix (1.1.6) of the right-hand side of the ODE system. We also solve the ODE systems with the non-stiff solvers `ode45` and `ode113` for contrast.

Section 4.2 reports on the performance of four different versions of Matlab's `ode15s` that implements the NDFk family:

Original ode15s with numerical Jacobian : This version uses Matlab's original implementation of `ode15s`. The NDFk methods require the Jacobian (1.1.6). If it is not supplied, the solver computes a numerical approximation automatically, which is an expensive computation.

Memory modified ode15s with numerical Jacobian: This version makes two improvements on the management of memory in `ode15s`. The first improvement is in the allocation and reallocation of memory for vectors stored by the function. The second improvement is to stop `ode15s` from storing the 3-D array `dif3d` which contains all gradient approximations for all solution components at all timesteps. This reduces the memory needed for the simulation significantly and enables the solution of larger problems. It also has the potential for speedup for smaller problems, if the reduction in memory usage lets a portion of the Jacobian fit into the cache of the CPU.

Memory modified ode15s with dense Jacobian: We supply the `ode15s` function with an analytical representation of the Jacobian of the ODE system that implements the block structure of diagonal matrices noted in (1.1.6) in this implementation. Evaluating an analytic Jacobian is much cheaper computationally than computing its approximation numerically and also increases its accuracy.

Memory modified ode15s with sparse Jacobian: We supply the `ode15s` function with the Jacobian in sparse storage mode in this implementation. This recognizes the mathematical fact in (1.1.6) that most matrix elements of diagonal matrices are zero. Since Matlab stores a sparse matrix more efficiently than a dense matrix, we can expect a speedup in the runtime of simulations.

Explicit ode45: This ODE solver uses the Runge-Kutta-Fehlberg 5(4) pair of Dormand and Prince. This is an explicit, non-stiff solver [7].

Implicit ode113: This ODE solver is a fully variable step size, predictor-corrector implementation of the Adams-Bashforth-Moulton family of formulas of orders 1 to 12. This is an implicit, non-stiff solver [12].

Table 1.2.1 contains the runtimes for an islet with no coupling between cells using each of the four implementations of NDFk; other cases exhibit analogous behavior, as detailed in Section 4.2. We observe that each new implementation improves the runtime. This is particularly evident for the largest desired case $N = 10$, where runtimes go from nearly 4 hours (13,815 seconds) using the original `ode15s` to under 1.5 hours (5,046 seconds) using the Memory Modified `ode15s` to under 1 hour (3,023 seconds) using Memory Modified `ode15s` with a Dense Jacobian and finally to just over a minute (84 seconds) using the Memory Modified `ode15s` with a Sparse Jacobian. This brings out the importance of using Matlab appropriately by using its sparse storage mode to respect correctly the sparsity of the Jacobian as well as the opportunity for significant improvements in efficiency by modifying the memory management in Matlab's `ode15s` function. These improvements together make overall runtimes for large parameter studies far more reasonable. In particular, physiological studies for islet of size larger than $5 \times 5 \times 5$ can be more easily carried out.

Table 1.2.1 also contains a column for the runtimes of `ode45` and `ode113`, two of Matlab's non-stiff ODE solvers. We observe that `ode45` has better runtime than

N	Original	Memory Modified	Dense Jacobian	Sparse Jacobian	ode45	ode113
2	18	17	13	14	12	14
3	43	32	16	14	13	44
4	114	60	23	17	21	179
5	295	126	43	20	33	690
6	765	282	116	26	58	1,767
7	2,140	654	303	34	111	5,552
8	3,498	1,264	630	46	159	8,229
9	8,689	2,511	1,394	61	267	11,705
10	13,815	5,046	3,023	84	366	16,036

Table 1.2.1 Runtimes (in seconds) for the case of no coupling strength for original ode15s, memory modified ode15s with numerical Jacobian, memory modified ode15s with dense Jacobian, Memory Modified ode15s with sparse Jacobian, ode45, ode113.

all of our versions of `ode15s` except the memory modified `ode15s` with sparse Jacobian. We also observe that `ode113` has a worse runtime than any version of `ode15s`. Tables 1.2.2(a) and (b) contain runtimes for the equally distributed islet structure with coupling strengths between cells of 0.1 and 1. We observe that increasing the coupling strength has a far greater affect on runtime for the non-stiff solvers than for the stiff solver, `ode15s`. In fact, for $N = 10$ and coupling strength of 1, `ode113` runs out of memory while `ode45` takes over 3 hours (11,249 seconds), longer than even original `ode15s` with numerical Jacobian, which took over 2 hours (7,649 seconds). Increasing the coupling strength increases the magnitude of the coupling matrix G, thus increasing the stiffness of the system. This shows the benefits of using a stiff ODE solver over non-stiff solvers for systems of ODEs with potential for progressively stiffer dynamics.

Simulations were run on one node of the cluster tara in the UMBC High Performance Computing Facility (`www.umbc.edu/hpcf`). This node contains two quad-core Intel Nehalem X5550 processors (2.66 GHz, 8,192 kB cache) and 24 GB of memory.

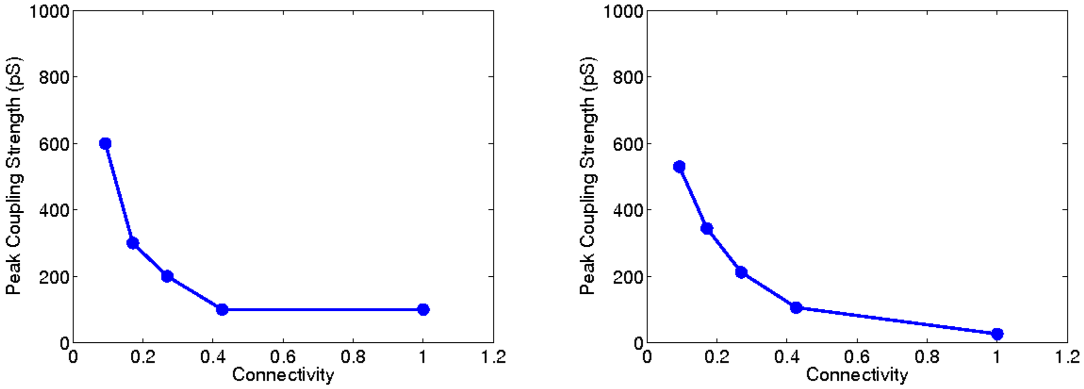
(a) Coupling Strength 0.1

N	Original	Memory Modified	Dense Jacobian	Sparse Jacobian	ode45	ode113
2	5	5	4	5	7	7
3	8	8	5	4	9	24
4	24	16	9	7	14	117
5	57	31	16	9	27	481
6	175	86	48	14	52	1,394
7	508	199	119	18	102	4,607
8	995	458	288	28	157	7,480
9	1,512	676	440	30	221	7,958
10	4,096	1,848	1,304	59	354	14,444

(b) Coupling Strength 1

N	Original	Memory Modified	Dense Jacobian	Sparse Jacobian	ode45	ode113
2	10	10	7	8	34	49
3	20	17	9	8	74	544
4	48	33	13	10	216	3,324
5	117	64	25	13	580	14,263
6	295	131	70	17	1,356	40,160
7	806	239	168	24	2,846	128,185
8	1,545	467	354	35	6,965	Out of Memory
9	4,432	936	741	50	8,105	Out of Memory
10	7,649	2,061	1,604	70	11,249	Out of Memory

Table 1.2.2 Runtimes (in seconds) for equally distributed islet structure for original ode15s, memory modified ode15s with numerical Jacobian, memory modified ode15s with dense Jacobian, memory modified ode15s with sparse Jacobian, ode45, ode113.



(a) (b)

Figure 1.1.1 Peak coupling strength vs. connectivity for the $5 \times 5 \times 5$ islet for the (a) seven variable model and (b) three variable model.

CHAPTER 2

MODEL

This chapter explains the models. Section 2.1 discusses the physiological background of the models. Section 2.2 describes the mathematical properties that are common for both models; in particular, the five different islet structures are defined here. Section 2.3 gives a few mathematical properties of the seven variable model equations and Section 2.4 gives a few mathematical properties of the three variable model equations.

2.1 Physiological Background

The pancreas is an organ in the body which is part of both the endocrine system and digestive system. In the endocrine system in the pancreas are clusters of cells called islets of Langerhans. These clusters of cells contain α-cells, β-cells, δ-cells, and PP cells along with capillaries, where β-cells are the most common type of cell in an islet of Langerhans. An islet's production of insulin, the key hormone in blood glucose maintenance released by β-cells, is related to both its metabolic and electrical activities.

Figure 2.1.1 illustrates how a β-cell responds to glucose entering the cell based on the consensus model of stimulus-secretion coupling. In the consensus model, after glucose enters the β-cell through the glucose transporter GLUT2 it is converted into pyruvate through glycolosis and then metabolized inside mitochondria. This process produces adenosine triphospate (ATP) and cellular energy. The increase in ATP results in the closing of $\mathrm{K_{ATP}}$ channels. This results in the depolarization of the β-cell which allows calcium to enter the cell. The calcium triggers the exocytosis of insulin containing secretory granules. The insulin causes the blood glucose to return back to

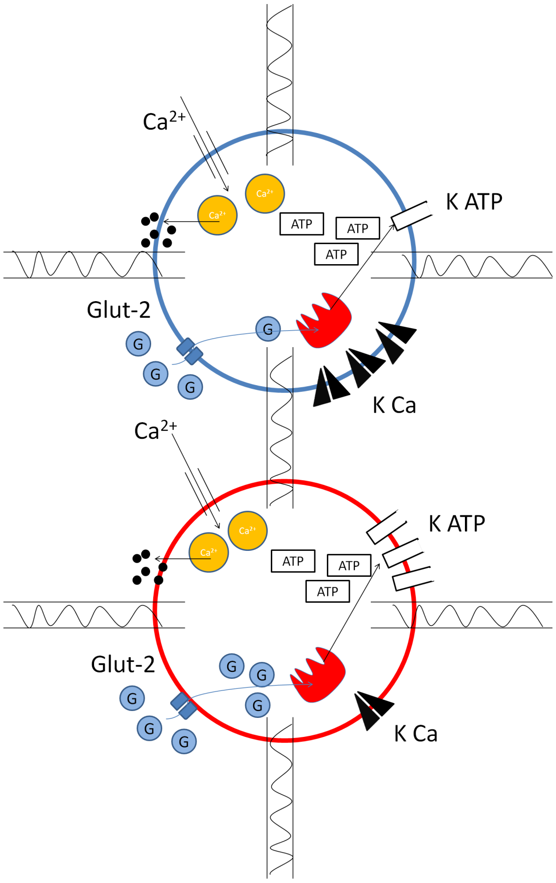


Figure 2.1.1 Schematic of two electrically coupled β-cells.

basal levels by signaling to cells throughout the body. As the glucose levels drop at the β-cell, ATP levels also tend to recover, allowing the K_{ATP} channels to open back up. The opening of these channels stops the electrical activity [3].

The change in voltage that happens during this process occurs in bursts which repeat every few seconds to minutes, depending on the properties of the cell. Cells of different bursting rates can exist in a single islet with no known pattern of distribution. The cells which burst every few seconds, usually on the order of tens of seconds, are categorized as fast β-cells. The cells which burst every few minutes, usually around every 4 to 6 minutes, are categorized as slow β-cells [14].

In an islet, cells are connected to each other through a complex of proteins called a gap junction. These gap junctions allow both ions and small molecules to travel between cells. So, cells in an islet affect each other's electrical activity through the ions traveling through the gap junction. Due to this coupling between cells, a single-cell model can be thought of as describing the behavior of a single cell in a synchronized islet [14]. This also means that the secretion of insulin in one cell is affected by other cells that are connected through a gap junction due to the ions and small molecules traveling between the two cells. So, we will use a model which takes into the electrical activity in the cell to describe the bursting of a β-cell.

2.2 Properties of Both Models

In an actual pancreas, β-cells are found in clusters in an islet of Langerhans. So, our models discretize this cluster into an $N \times N \times N$ cube of cells.

Since the distribution of slow and fast cells in an islet of Langerhans is not known, we simulate several distributions to determine the effects of distribution on bursting behavior. The five islet structures studied were: grouped structure, split grouped structure, layered middle structure, layered split structure, and the equally distributed structure.

- The grouped structure has a block of $\frac{N^3}{2}$ fast cells and a block of $\frac{N^3}{2}$ slow cells. Figure 2.2.1(a) has a visual representation of this structure.

- The split grouped structure has two non-adjacent blocks of $\frac{N^2}{4}$ fast cells and two non-adjacent blocks of $\frac{N^3}{4}$ slow cells. Figure 2.2.1(b) has a visual representation of this structure.

- The layered middle structure has N alternating layers of N^2 slow and then N^2 fast cells. However, the middle layer is $\frac{N^2}{2}$ slow cells and $\frac{N^2}{2}$ fast cells.

Figure 2.2.1(c) has a visual representation of this structure.

- The layered split structure has N alternating layers where a layer with the first $\frac{N^2}{2}$ cells are slow the last $\frac{N^2}{2}$ cells are fast is followed by a layer with the first $\frac{N^2}{2}$ cells are fast and the last $\frac{N^2}{2}$ cells are slow and vice versa. Figure 2.2.1(d) has a visual representation of this structure.
- In the equally distributed structure, all neighbors of fast cells are slow cells and all neighbors of slow cells are fast cells. Figure 2.2.1(e) has a visual representation of this structure.

We introduce a measure of connectivity based on the fraction of neighboring cells with speed different from the cell's own speed. This quantitative measure of connectivity provides a rational ordering of the different islet structures under consideration. The following was used to compute connectivity

$$c = \frac{1}{N^3} \sum_{i=1}^{N^3} \frac{d_i}{v_i}, \tag{2.2.1}$$

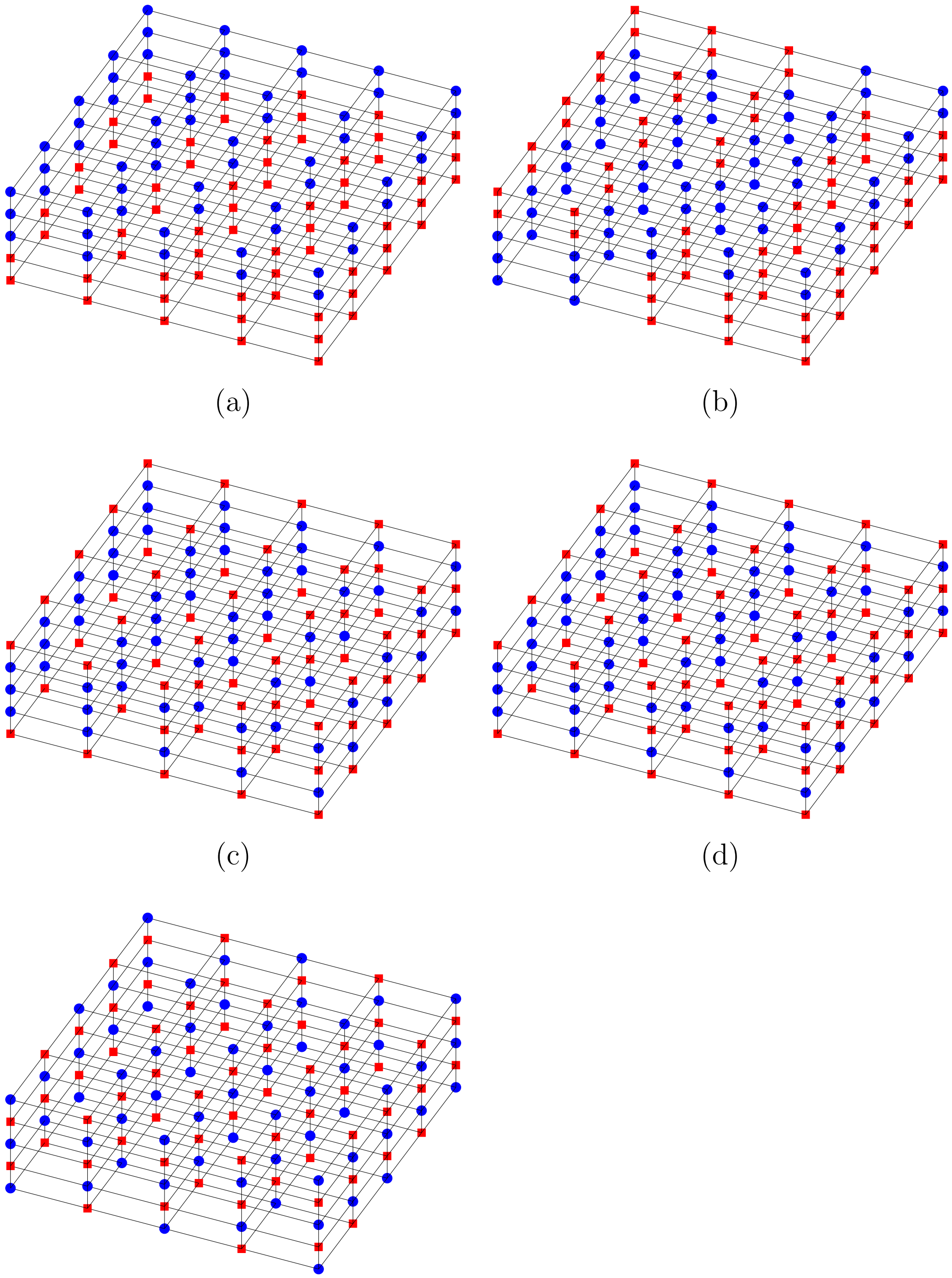
where

$$d_i = \begin{cases} \text{number of slow cells connected to cell } i \text{ if cell } i \text{ is fast,} \\ \text{number of fast cells connected to cell } i \text{ if cell } i \text{ is slow,} \end{cases}$$

and v_i is the total number of cells connected to i.

For example, in the case of a $5 \times 5 \times 5$ computational islet with 50% slow cells and 50% fast cells, we have the following connectivities for our five islet structures. The grouped structure has a connectivity of 0.0928. The split grouped structure has a connectivity of 0.1712. The layered middle structure has a connectivity of 0.2693. The layered split structure has a connectivity of 0.4261. Finally, the equally distributed structure has a connectivity of 1.0000.

In an islet, cells are joined together by proteins called gap junctions which allow both ions and small molecules to move between cells. We assume nearest neighbor



(a)

(b)

(c)

(d)

(e)

Figure 2.2.1 Structure for a 5 × 5 × 5 islet where a square represents a fast cell and a circle represents a slow cell. (a) grouped structure (b) split grouped structure (c) layered middle structure (d) layered split structure (e) equally distributed structure.

coupling in our models, which means that only cells directly adjacent to each other have a gap junction between them.

Let S be the set of $n_s = 7$ or 3 state variables per cell for the seven variable model or three variable model, respectively. For a given state variable s in S, if there is coupling between cells for this state variable, the effect of coupling between a cell and its neighbor is described by

$$\begin{aligned} g_s = g^A_{i,j,k}(c^{i+1,j,k} - c^{i,j,k}) + g^B_{i,j,k}(c^{i-1,j,k} - c^{i,j,k}) + g^C_{i,j,k}(c^{i,j+1,k} - c^{i,j,k}) \\ +g^D_{i,j,k}(c^{i,j-1,k} - c^{i,j,k}) + g^E_{i,j,k}(c^{i,j,k+1} - c^{i,j,k}) + g^F_{i,j,k}(c^{i,j,k-1} - c^{i,j,k}), \end{aligned}$$

for $i, j, k = 1, ..., N$ where $g^X_{i,j,k}$ is the coupling strength between the cell in the (i, j, k)-th cell in the islet, $X = A, ..., F$ and its respective neighbor for state variable s and X distinguishes between the coupling for different neighbors.

Our simulation uses a matrix, C, where the (a, b) entry of C is the coupling strength between cell a and cell b. Here, a and b are indexed by an array derived from the $N \times N \times N$ cube of cells. The array, A, is created by using the following indexing: $A(i + Nj + N^2k) = (i, j, k)$-th cell of our islet.

C is a matrix with seven non-zero bands. In our model, the coupling strength between cells is the same throughout the islet. So, the elements on the main diagonal are the number of neighbors each cell has multiplied by -1 and the coupling strength of the islet. On the first superdiagonal and subdiagonal, the N superdiagonal and subdiagonal, and the N^2 superdiagonal and subdiagonal are the coupling strength of the islet. This C matrix is used to define the G matrix as

$$G = \begin{pmatrix} C & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 \end{pmatrix}.$$

G is a block matrix with 7×7 blocks for the seven variable model and a block matrix with 3×3 blocks for the three variable model. The upper left block is for the voltage variable and contains the C matrix.

2.3 Mathematical Properties of the Seven Variable Model Equations

We simulate the glycolytic oscillator model for β-cells using a deterministic model developed by Smolen [16]. This model was then coupled by Bertram and Sherman [2] with Sherman's update [14] of the model developed by Chay and Keizer [4] and includes voltage data collected by Rorsman and Trube [11].

This model uses 7 independent variables: V, n, [Ca], [$\mathrm{Ca_{er}}$], [ADP], [G6P], and [FBP], where [XX] denotes the concentration of species XX in moles per liter.

V represents the electric potential of the cell membrane in mV.

n represents the fraction of K^+ channels that are open in the cell.

[Ca] represents the concentration of free intracellular calcium ions.

[$\mathrm{Ca_{er}}$] represents the concentration of calcium ions in the endoplasmic reticulum.

[ADP] represents the concentration of Adenosine Diphosphate (ADP) in the cell.

[G6P] represents the concentration of Glucose 6-Phosphate in the cell.

[FBP] represents the concentration of Fructose 1,6-Biphosphate in the cell.

There are also the following ionic currents that are used in the individual β-cell model: I_{K}, I_{Ca}, $I_{\mathrm{K\,(Ca)}}$, $I_{\mathrm{K\,(ATP)}}$, where $I_{\mathrm{XX(YY)}}$ is the current of ion XX and modulated by YY. These currents are defined as

$$
\begin{aligned}
I_{\mathrm{K}} &= \overline{g}_{\mathrm{K}}\, n\, (V - V_{\mathrm{K}}),\\
I_{\mathrm{Ca}} &= \overline{g}_{\mathrm{Ca}}\, m_{\infty}(V)\, (V - V_{\mathrm{Ca}}),\\
I_{\mathrm{K\,(Ca)}} &= g_{\mathrm{K\,Ca}}\, (V - V_{\mathrm{K}}),\\
g_{\mathrm{K\,Ca}}([\mathrm{Ca}]) &= \frac{\overline{g}_{\mathrm{K\,Ca}}([\mathrm{Ca}])}{1 + \left(\frac{K_d}{[\mathrm{Ca}]}\right)^2},\\
I_{\mathrm{K\,(ATP)}} &= g_{\mathrm{K\,ATP}}\, (V - V_{\mathrm{K}}),\\
g_{\mathrm{K\,ATP}} &= \overline{g}_{\mathrm{K,ATP}}\, o_{\infty}(V),\\
m_{\infty}(V) &= \frac{1}{1 + \exp\left[\frac{-(20+V)}{12}\right]},\\
o_{\infty}([\mathrm{ADP}]) &= \frac{\left((8.4\times 10^{-5})\,[\mathrm{ADP}]^2 + 0.0016\,[\mathrm{ADP}] + 0.08\right)\overline{g}_{\mathrm{K,ATP}}}{(0.05\,[\mathrm{ATP}] + 0.0052\,[\mathrm{ADP}] + 1)\,(1 + 0.0098\,[\mathrm{ADP}])^2},\\
[\mathrm{ATP}] &= 1500 - \frac{1}{2}[\mathrm{ADP}] + \frac{1}{2}\left(-3[\mathrm{ADP}]^2 - 6000[\mathrm{ADP}] + 9000000\right)^{\frac{1}{2}},\\
J_{\mathrm{er}} &= 0.0002([\mathrm{Ca}_{\mathrm{er}}] - [\mathrm{Ca}]) - 0.4[\mathrm{Ca}],\\
J_{\mathrm{mem}} &= (4.5\times 10^{-6}) I_{\mathrm{Ca}} - 0.2[\mathrm{Ca}],
\end{aligned}
$$

where R_{GK}, $\overline{g}_{\mathrm{K\,Ca}}$, and $\overline{g}_{\mathrm{K,ATP}}$ are parameters relating to the bursting rate of the cell. These are the parameters altered for fast or slow cells.

The following constants are used in this model: $C_m = 5300$ fF, $V_{\mathrm{K}} = -75$ mV, $g_{\mathrm{K}} = 2700$ pS, $\tau_n = 20$ ms, $g_{\mathrm{Ca}} = 1000$ pS, $V_{\mathrm{Ca}} = 25$ mV. Table 2.3.1 collects parameters of the system.

The seven variable model for the dynamics of one β-cell, which takes into account both metabolic and electrical processes, is the set of ordinary differential equations

$$\frac{dV}{dt} = \frac{-(I_{\mathrm{K}} + I_{\mathrm{Ca}} + I_{\mathrm{K\,(Ca)}} + I_{\mathrm{K\,(ATP)}})}{C_m}, \tag{2.3.1}$$

$$\frac{dn}{dt} = \frac{(n_\infty - n)}{\tau_n}, \tag{2.3.2}$$

$$\frac{d[\mathrm{Ca}]}{dt} = f_{\mathrm{cyt}}\,(J_{\mathrm{mem}} + J_{\mathrm{er}}), \tag{2.3.3}$$

$$\frac{d[\mathrm{Ca_{er}}]}{dt} = -\sigma_V\, f_{\mathrm{er}}\, J_{\mathrm{er}}, \tag{2.3.4}$$

$$\frac{d[\mathrm{ADP}]}{dt} = \frac{[\mathrm{ATP}] - [\mathrm{ADP}]\ \exp\left\{(r+\gamma)\left(1 - \frac{[\mathrm{Ca}]}{r_1}\right)\right\}}{\tau_a}, \tag{2.3.5}$$

$$\frac{d[\mathrm{G6P}]}{dt} = \kappa\,(R_{\mathrm{GK}} - R_{\mathrm{PFK}}), \tag{2.3.6}$$

$$\frac{d[\mathrm{FBP}]}{dt} = \kappa\,(R_{\mathrm{PFK}} - 0.5\,R_{\mathrm{GPDH}}). \tag{2.3.7}$$

R_{PFK} is the rate of phosphofructokinase activity. It is given as a combinatorial function of [G6P] and [FBP] that can be found in [16]. Each cell in our cluster has a certain state based on the 7 independent variables. We call this set of independent variables S.

Our model also assumes that each cell is either a fast or slow bursting cell. This relates to the rate at which the cell bursts with a certain oscillatory pattern. For a slow cell, we define the following parameters: $R_{\mathrm{GK}} = 0.2$, $\overline{g}_{\mathrm{K\,Ca}} = 27000$ pS, and $\overline{g}_{\mathrm{K,ATP}} = 100$ pS. For a fast cell we define these parameters as: $R_{\mathrm{GK}} = .4$, $\overline{g}_{\mathrm{K\,Ca}} = 25000$ pS, and $\overline{g}_{\mathrm{K,ATP}} = 600$ pS.

Let V, n, [Ca], [Ca$_{\mathrm{er}}$], [ADP], [G6P], and [FBP] each represent a vector corresponding to the value of the variable for each of the N^3 cells. For each vector the ℓ-th element is the value of the parameter for cell ℓ using the following indexing: $A(i + Nj + N^2k) = (i, j, k)$-th cell of our islet, where $\ell = i + Nj + N^2k$. Since the matrix G of coupling strengths groups all coupled terms together, our solution vector,

y, is the $7N^3 \times 1$ vector

$$y = (V, n, [\mathrm{Ca}], [\mathrm{Ca_{er}}], [\mathrm{ADP}], [\mathrm{G6P}], [\mathrm{FBP}])^T ,$$

The vector relating $f(t, y)$, the right-hand side of (2.3.1)–(2.3.7), is similarly defined:

$$f(t, y) = \left(f_V(t, y), f_n(t, y), f_{[\mathrm{Ca}]}(t, y), f_{[\mathrm{Ca_{er}}]}(t, y), f_{[\mathrm{ADP}]}(t, y), f_{[\mathrm{G6P}]}(t, y), f_{[\mathrm{FBP}]}(t, y)\right)^T$$

So, in matrix form our initial value problem will be

$$\frac{dy}{dt} = f(t, y) + Gy, \qquad 0 < t \le t_f, \quad y(0) = y_0.$$

The initial condition is $y_0 = (V_0, n_0, [\mathrm{Ca}]_0, [\mathrm{Ca_{er}}]_0, [\mathrm{ADP}]_0, [\mathrm{G6P}]_0, [\mathrm{FBP}]_0)^T$ with $V_0 = -60$ mV, $n_0 = 0.00016$, $[\mathrm{Ca}]_0 = 0.08$ nM, $[\mathrm{Ca_{er}}]_0 = 789$ nM, $[\mathrm{ADP}]_0 = 170$ nM, $[\mathrm{G6P}]_0 = 187$ nM, $[\mathrm{FBP}]_0 = 16$ nM.

Symbol	Name	Example or Range	Units
t	time	$0 \leq t \leq 500000$	ms
C_m	capacitance	5300	fF
V_{K}	K^+ Nernst Potential	-75	mV
V_{Ca}	Ca^{2+} Nernst Potential	25	mV
g_{K}	channel conductance	2700	pS
τ_n	time constant	20	ms
g_{Ca}	channel conductance	1000	pS
R_{GK}	rate of glucose influx	0.2 (s) or 0.4 (f)	nM/ms
$\overline{g}_{\mathrm{K\,Ca}}$	maximum K Ca channel conductance	25000 (f) or 27000 (s)	pS
$\overline{g}_{\mathrm{K,ATP}}$	maximum K ATP channel conductance	100 (s) or 600 (f)	pS
f_{cyt}	rapid calcium buffering in the cytosol	0.01	1
f_{er}	rapid calcium buffering in the ER	0.01	1
σ_V	cytosolic to ER volume ratio	31	1
κ	conversion parameter for glycolytic subsystem	0.005	1

Table 2.3.1 Constants and variables for the seven variable model, where (f) or (s) denote which parameter value goes with a fast or slow cell, respectively.

2.4 Mathematical Properties of the Three Variable Model Equations

Sherman and Rinzel developed a three variable model to simulate the behavior of two coupled β-cells in [15]. This model uses 3 independent variables: V, n, and s.

V represents the electric potential of the cell membrane in mV.

n represents the fraction of open potassium channels for I_K.

s is the slow variable.

There are also the following ionic currents that are used in the two cell model: I_{Ca}, I_{K}, I_{s}. These currents are defined as

$$\begin{aligned} I_{\mathrm{K}} &= \hat{g}_{\mathrm{K}} n (V - V_{\mathrm{K}}), \\ I_{\mathrm{Ca}} &= \hat{g}_{\mathrm{Ca}} m_\infty(V)(V - V_{\mathrm{Ca}}), \\ I_{\mathrm{s}} &= \hat{g}_{\mathrm{s}} s (V - V_{\mathrm{K}}), \end{aligned}$$

where

$$\nu_\infty(V) = \frac{1}{1 + \exp[\frac{V_\nu - V}{\theta_\nu}]} \quad \text{for } \nu = m, n, s. \tag{2.4.1}$$

The following constants are used in this model: $g_{\mathrm{Ca}} = 3.6$, $g_{\mathrm{K}} = 10$, $g_{\mathrm{s}} = 4$, $V_{\mathrm{Ca}} = 25$ mV, $V_{\mathrm{K}} = -75$ mV, $V_m = -20$ mV, $V_n = -17$ mV, $V_s = -38$ mV, $\theta_m = 12$ mV, $\theta_n = 5.6$ mV, $\theta_s = 10$ mV, $\tau = 20$ ms, $\lambda = 0.9$, $\tau_s = 35000$ ms, $C_m = 5300$ fF, $\hat{g}_{\mathrm{Ca}} = \frac{C_m g_{\mathrm{Ca}}}{\tau}$ pS=954 pS, $\hat{g}_{\mathrm{K}} = \frac{C_m g_{\mathrm{Ca}}}{\tau}$ pS= 2650 pS, $\hat{g}_{\mathrm{s}} = \frac{C_m g_{\mathrm{s}}}{\tau}$ pS=1060 pS. Table 2.4.1 collects parameters of the system.

The two cell model for the dynamics of a β-cell is the set of ordinary differential equations

$$\frac{dV}{dt} = \frac{-(I_{\mathrm{K}} + I_{\mathrm{Ca}} + I_s)}{C_m}, \tag{2.4.2}$$

$$\frac{dn}{dt} = \frac{\lambda(n_\infty(V) - n)}{\tau}, \tag{2.4.3}$$

$$\frac{ds}{dt} = \frac{s_\infty(V) + \beta - s}{\tau_s}, \tag{2.4.4}$$

for each cell, where β differs to introduce heterogeneity between cells.

We define y as the 3×1 vector $[V, n, s]^T$ and $f(t, y)$ as a 3×1 vector consisting of the right-hand side of the ODEs (2.4.2)–(2.4.4), which becomes

$$f(t,y) = \begin{bmatrix} \frac{-(I_{\mathrm{K}}+I_{\mathrm{Ca}}+I_s)}{C_m} \\ \frac{\lambda(n_\infty(V)-n)}{\tau} \\ \frac{s_\infty(V)+\beta-s}{\tau_s} \end{bmatrix}.$$

This vector function can be rewritten as

$$f(t,y) = \begin{bmatrix} \frac{-\hat{g}_{\mathrm{K}}}{C_m}(V - V_{\mathrm{K}})n - \frac{\hat{g}_{\mathrm{Ca}}}{C_m}(V - V_{\mathrm{Ca}})m_\infty(V) - \frac{\hat{g}_{\mathrm{s}}}{C_m}(V - V_{\mathrm{K}})s \\ \frac{\lambda}{\tau}(n_\infty(V) - n) \\ \frac{1}{\tau_s}(s_\infty(V) + \beta - s) \end{bmatrix}, \tag{2.4.5}$$

with $\nu_\infty(V)$ in (2.4.1). So, the Jacobian of the ODE system for each cell is given by

$$J = \begin{bmatrix} J_{11} & \frac{-\hat{g}_{\mathrm{K}}}{C_m}(V - V_{\mathrm{K}}) & \frac{\hat{g}_{\mathrm{s}}}{C_m}(V - V_{\mathrm{K}}) \\ \frac{\lambda}{\tau}n'_\infty(V) & -\frac{\lambda}{\tau} & 0 \\ \frac{1}{\tau_s}s'_\infty(V) & 0 & -\frac{1}{\tau_s} \end{bmatrix}$$

with $J_{11} = -\frac{\hat{g}_{\mathrm{K}}}{C_m}\, n - \frac{\hat{g}_{\mathrm{Ca}}}{C_m}\, m_\infty(V) - \frac{\hat{g}_{\mathrm{Ca}}}{C_m}\,(V - V_{\mathrm{Ca}})\, m'_\infty(V) - \frac{\hat{g}_{\mathrm{s}}}{C_m}\, s$ and

$$\nu'_\infty(V) \equiv \frac{d\nu_\infty}{dV} = \frac{d}{dV}\left(\left[1 + \exp\left(\frac{V_\nu - V}{\theta_\nu}\right)\right]^{-1}\right) = \frac{\exp\left(\frac{V_\nu - V}{\theta_\nu}\right)}{\theta_\nu\left[1 + \exp\left(\frac{V_\nu - V}{\theta_\nu}\right)\right]^2}, \tag{2.4.6}$$

for $\nu = m, n, s$.

Now, we extend this two cell model to an $N \times N \times N$ computational islet. Let V, n, and s each represent a vector corresponding to the value of the variable for each of the N^3 cells. For each vector, the ℓ-th element is the value of the parameter for cell ℓ using the following indexing: $A(i + Nj + N^2k) = (i, j, k)$-th cell of our islet, where $\ell = i + Nj + N^2k$. Since the matrix G of coupling strengths groups all coupled terms together, our solution vector, y, is the following $3N^3 \times 1$ vector

$$y = \begin{bmatrix} V \\ n \\ s \end{bmatrix}.$$

For a model with N^3 cells, $f(t, y)$ is a $3N^3 \times 1$ vector where the first N^3 elements are the values of the right-hand side of (2.4.2) for each cell, the second N^3 elements are the values of the right-hand side of (2.4.3) for each cell, and the third N^3 elements are the values of the right-hand side of (2.4.4) for each cell. So, in matrix form our initial value problem will be

$$\frac{dy}{dt} = f(t, y) + Gy, \qquad 0 < t \leq t_f, \quad y(0) = y_0, \tag{2.4.7}$$

The initial condition is $y_0 = (V_0, n_0, s_0)^T$ with $V_0 = -60$ mV, $n_0 = 0.0001$, $s_0 = 0.4$. The Jacobian of the right-hand side of the ODE is then given by:

$$\nabla_y(f(t, y) + Gy) = J + G = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix} + G, \tag{2.4.8}$$

where

$$J_{11} = \operatorname{diag}\Big(-\frac{\hat{g}_{\mathrm{K}}}{C_m}\, n - \frac{\hat{g}_{\mathrm{Ca}}}{C_m}\, m_\infty(V) - \frac{\hat{g}_{\mathrm{Ca}}}{C_m}\, (V - V_{\mathrm{Ca}})\, m_\infty'(V) - \frac{\hat{g}_{\mathrm{s}}}{C_m}\, s \Big), \tag{2.4.9}$$

$$J_{12} = \operatorname{diag}\Big(\frac{-\hat{g}_{\mathrm{K}}}{C_m}(V - V_{\mathrm{K}}) \Big), \tag{2.4.10}$$

$$J_{13} = \operatorname{diag}\Big(\frac{\hat{g}_{\mathrm{s}}}{C_m}(V - V_{\mathrm{K}}) \Big), \tag{2.4.11}$$

$$J_{21} = \operatorname{diag}\Big(\frac{\lambda}{\tau} n_\infty'(V) \Big), \tag{2.4.12}$$

$$J_{22} = -\frac{\lambda}{\tau} I_{N^3 \times N^3}, \tag{2.4.13}$$

$$J_{23} = 0_{N^3 \times N^3}, \tag{2.4.14}$$

$$J_{31} = \frac{1}{\tau_s} s_\infty'(V) I_{N^3 \times N^3}, \tag{2.4.15}$$

$$J_{32} = 0_{N^3 \times N^3}, \tag{2.4.16}$$

$$J_{33} = -\frac{1}{\tau_s} I_{N^3 \times N^3}. \tag{2.4.17}$$

with $\nu_\infty'(V)$ defined in (2.4.6). Each $N^3 \times N^3$ block of J is either a diagonal matrix or the zero matrix. Here, $0_{N^3 \times N^3}$ denotes an $N^3 \times N^3$ matrix in which all elements are zeros, and the identity $I_{N^3 \times N^3}$ is an $N^3 \times N^3$ matrix in which the diagonal elements are ones and all other elements are zeros. Some blocks of J are diagonal matrices with the same constant in all diagonal elements, represented above by blocks involving the product of a constant with $I_{N^3 \times N^3}$. Other blocks have different values in the diagonal elements, indicated by the notation $\text{diag}(v)$, which constructs a diagonal matrix with the components of the $N^3 \times 1$ vector v along the diagonal.

Symbol	Name	Example or Range	Units
t	time	$0 \leq t \leq 200000$	ms
V	electric potential	$-70 \leq V \leq -20$	mV
n	fraction of open K^+ channels for I_K	$0 \leq n \leq 1$	1
s	slow variable	$0 \leq s \leq 1$	1
g_K	channel conductance	10	1
g_s	channel conductance	4	1
g_{Ca}	channel conductance	3.6	1
$\hat{g}_{Ca}$	channel conductance	954	pS
$\hat{g}_K$	channel conductance	2650	pS
$\hat{g}_s$	channel conductance	1060	pS
V_K	K^+ Nernst Potential	-75	mV
V_{Ca}	Ca^{2+} Nernst Potential	25	mV
V_m	Half-maximal activation of channel	-20	mV
V_n	Half-maximal activation of channel	-17	mV
V_s	Half-maximal activation of channel	-38	mV
θ_m	m-gate sensitivity	12	mV
θ_n	n-gate sensitivity	5.6	mV
θ_s	s-gate sensitivity	10	mV
τ	n-gate time constant	20	ms
τ_s	s-gate time constant	35000	ms
λ	n-gate time constant modifier	$0 \leq \lambda \leq 1$	1
β	heterogeneity constant	$-1 \leq \beta \leq 1$	1

Table 2.4.1 Constants and variables for the three variable model.

CHAPTER 3

NUMERICAL METHOD

This chapter introduces the numerical method used to solve both the seven variable and the three variable model, stated in Section 1.1 and described in detail in Sections 2.3 and 2.4, respectively. Systems of ODEs of these types need to be handled as stiff equations, since the reactions modeled by the ODEs can have a wide range of speeds. The following is an explanation of the Numerical Differentiation Formulas (NDFk), which improve upon the well-known Backward Differentiation Formulas (BDFk), see, e.g., [1, Section 6.9]. The BDFk have long been the premier stiff ODE solver, implemented in codes such as EPISODE and LSODE. They continue to be used in codes such as VODE, CVODE, and PVODE, which are used in state-of-the-art packages such as SUNDIALS. The NDFk are an improved extension of the BDFk developed for Matlab's `ode15s` function [13].

We describe the method here for the general initial value problem

$$M\,y' = f^{(ode)}(t,y), \quad 0 < t \le t_f, \quad y(0) = y_0, \tag{3.0.1}$$

with a mass matrix M, as implemented in `ode15s`. We take M constant here. For our application problems, it is the identity matrix $M = I$, and the right-hand side function is $f^{(ode)}(t,y) = f(t,y) + G\,y$.

The idea of the BDFk family is to approximate the derivative $y'(t)$ on the left-hand side of the ODE by a k-th order accurate one-sided finite difference approximation. Since a method for stiff ODEs has to be necessarily implicit in time, the ODE is evaluated at $t = t_{n+1}$, and the times involved in the finite difference are $t_n, t_{n-1}, \ldots t_{n-k+1}$.

For example, the first order accurate approximation ($k = 1$) to the derivative $y'(t)$

at t_{n+1} backward in time involves only t_n and is

$$y'(t_{n+1}) \approx \frac{y_{n+1} - y_n}{\Delta t} + \frac{\Delta t}{2} y''(\xi),$$

where $t_{n-1} \leq \xi \leq t_n$, which will result in the implicit Euler method. The second order accurate approximation ($k = 2$) to the derivative $y'(t)$ at t_{n+1} backward in time involves t_{n-1} and t_n and is

$$y'(t_{n+1}) \approx \frac{y_{n-1} - 4y_n + 3y_{n+1}}{2\Delta t} + \frac{\Delta t^2}{3} y'''(\xi),$$

where $t_{n-1} \leq \xi \leq t_{n+1}$.

The general k-th order approximation to the derivative $y'(t)$ at $t = t_{n+1}$ backward in time can be written as

$$y'(t_{n+1}) \approx \frac{1}{\Delta t} \sum_{m=1}^{k} \frac{1}{m} \nabla^m y_{n+1} \tag{3.0.2}$$

with the backward difference notation defined recursively as

$$\nabla^m y_{n+1} := \nabla^{m-1} y_{n+1} - \nabla^{m-1} y_n, \qquad \text{for } m = 1, 2, \ldots,$$

and $\nabla^0 y_{n+1} := y_{n+1}$.

Evaluating the ODE at $t = t_{n+1}$ and inserting the k-th order approximation to $y'(t_{n+1})$ on the left-hand side gives the BDFk method of order k as

$$M \sum_{m=1}^{k} \frac{1}{m} \nabla^m y_{n+1} = \Delta t \, f^{(ode)}(t_{n+1}, y_{n+1}). \tag{3.0.3}$$

Using the identity

$$\sum_{m=1}^{k} \frac{1}{m} \nabla^m y_{n+1} = \gamma_k (y_{n+1} - \eta_n) + \sum_{m=1}^{k} \gamma_m \, \nabla^m y_n,$$

this equation can be rewritten as

$$M \, y_{n+1} = M \sum_{j=0}^{k-1} a_j \, y_{n-j} + \Delta t \, b \, f^{(ode)}(t_{n+1}, y_{n+1}) \tag{3.0.4}$$

with the values a_j and b tabulated for $k = 1, \ldots, 6$ in, for instance, [1, Table 6.19]. This BDFk has the truncation error

$$T_{n+1}(\Delta t) = -M \frac{\Delta t^{k+1}}{k+1} y^{(k+1)}(\xi_{n+1}) \tag{3.0.5}$$

and the local truncation error of order k given by

$$\tau_{n+1}(\Delta t) = \frac{T_{n+1}(\Delta t)}{\Delta t} = O(\Delta t^k).$$

The Numerical Differentiation Formulas (NDFk) are defined by adding a term to the BDFk method in (3.0.3) to get

$$M \sum_{m=1}^{k} \frac{1}{m} \nabla^m y_{n+1} = \Delta t\, f^{(ode)}(t_{n+1}, y_{n+1}) + \alpha_k\, \gamma_k\, M\, (y_{n+1} - \eta_n), \tag{3.0.6}$$

where $\alpha_k \geq 0$, $\gamma_k = \sum\limits_{j=1}^{k} \frac{1}{j}$, and $\eta_n = \sum\limits_{m=0}^{k} \nabla^m y_n$. The so-called predictor η_n is an approximation of y_{n+1} based on some explicit formula [13].

Using $y_{n+1} - \eta_n = \nabla^{k+1} y_{n+1}$ and the approximation $\frac{\Delta t^{k+1} y^{(k+1)}}{k+1} \approx \frac{\nabla^{k+1} y_{n+1}}{k+1}$, we obtain the truncation error for NDFk as

$$T_{n+1}(\Delta t) = -M \left(\alpha_k \gamma_k + \frac{1}{k+1} \right) \Delta t^{k+1} y^{(k+1)}(\xi_{n+1}). \tag{3.0.7}$$

Therefore, the local truncation error still has order k just as for BDFk, but the parameter α_k is a free parameter that may be chosen to improve efficiency of the NDFk method. Table 1 in [13] tabulates the optimal α_k for $k = 1, \ldots, 5$.

The method (3.0.6) can be rewritten as

$$f^{(newt)}(y_{n+1}) := M\,(y_{n+1} - \eta_n) + M\,\Psi_n - \frac{\Delta t}{(1-\alpha_k)\gamma_k}\, f^{(ode)}(t_{n+1}, y_{n+1}) = 0, \tag{3.0.8}$$

where $\Psi_n = \frac{1}{(1-\alpha_k)\gamma_k} \sum\limits_{m=1}^{k} \gamma_m \nabla^m y_n$ collects all terms independent of y_{n+1}. Here, the method is stated as a rootfinding problem $f^{(newt)}(y_{n+1}) = 0$ for the ODE approximation y_{n+1} at the new time step t_{n+1}. ODE solvers for stiff problems necessitate the solution of this system of non-linear equations at every time step by the Newton method [1, page 413]. It is natural to use the predictor as initial guess $y_{n+1}^{(0)} = \eta_n$. The Newton method to find y_{n+1} iterates then for $i = 0, 1, \dots$ by

$$y_{n+1}^{(i+1)} = y_{n+1}^{(i)} - \left(J^{(newt)}(y_{n+1}^{(i)})\right)^{-1} f^{(newt)}(y_{n+1}^{(i)}), \tag{3.0.9}$$

where $J^{(newt)}(y_{n+1})$ is the Jacobian of $f^{(newt)}(y_{n+1})$ given by

$$J^{(newt)}(y_{n+1}) := \nabla_{y_{n+1}} f^{(newt)}(y_{n+1}) = M - \frac{\Delta t}{(1-\alpha_k)\gamma_k} J^{(ode)}(t_{n+1}, y_{n+1})$$

and $J^{(ode)}(t, y)$ is defined as the Jacobian of $f^{(ode)}(t, y)$ defined by $J^{(ode)}(t, y) := \nabla_y f^{(ode)}(t, y)$.

For the general problem $My' = f^{(ode)}(t, y)$, the Newton method consists thus of setting $y_{n+1}^{(0)} = \eta_n$ and then iterating for $i = 0, 1, \dots$ the linear solve

$$\left(M - \frac{h}{(1-\alpha_k)\gamma_k}\, J^{(ode)}\right) \Delta^{(i)} = \frac{\Delta t}{(1-\alpha_k)\gamma_k} f^{(ode)}(t_{n+1}, y_{n+1}^{(i)}) - M\,\Psi_n - M\,(y_{n+1}^{(i)} - \eta_n)$$

for $\Delta^{(i)}$ and the update $y_{n+1}^{(i+1)} = y_{n+1}^{(i)} + \Delta^{(i)}$. In `ode15s` the Jacobian is held constant for many iterations and time steps. It only re-evaluates the Jacobian only when it is absolutely necessary. The iteration matrix is defined as

$$M^{\text{iter}} = M - \frac{\Delta t}{(1-\alpha_k)\gamma_k}\, J^{(ode)}.$$

This iteration matrix is re-evaluated when J changes or the step size h is changed. The LU decomposition of M^{iter} is computed each time M^{iter} is re-evaluated and used to solve the system.

`ode15s` estimates the truncation error to determine whether to accept the approximation $y_{n+1}^{(i)}$. The difference $d^{(i)} = y_{n+1}^{(i)} - \eta_n$ is an estimator for the truncation error [13]. Then the update $y_{n+1}^{(i+1)}$ is equivalent to $y_{n+1}^{(i+1)} = \eta_n + d^{(i+1)}$ where $d^{(i+1)} = d^{(i)} + \Delta^{(i)}$.

3.1 Original ode15s with Numerical Jacobian

Matlab's `ode15s` implements NDFk to solve a system of ODEs. Since the NDFk uses Newton's method, it requires the Jacobian of the system of ODEs. Since no Jacobian is provided to the system, the Jacobian is numerically approximated using divided differences for each element in the matrix.

3.2 Memory Modified ode15s with Numerical Jacobian

Two issues were found in the Matlab's code for `ode15s` that were affecting its performance.

The first issue was a problem with memory management in the following line of `ode15s`:

```
chunk = min(max(100,50*refine), refine+floor((2^11)/neq));
```

This `chunk` is used to both allocate and reallocate memory for `tout`,`yout`, `kvec`, and `dif3d`. This occurs in the following lines of code from `ode15s`:

```
tout = zeros(1, chunk);

tout = [tout, zeros(1, chunk)];

yout = zeros(neq, chunk);

yout = [yout, zeros(1, chunk)];

kvec = zeros(1, chunk);

kvec = [kvec,zeros(1, chunk);

dif3d = zeros(neq, maxk + 2, chunk);

dif3d = cat(3, dif3d, zeros(neq, maxk + 2, chunk));
```

Here, `neq` is the number of equations, so $7N^3$ for our full seven variable model. `refine` is set as 1 by default or the value passed in by the user using the `'Refine'` parameter in `odeset`. If we assume `refine=1`, since `max(100,50*refine);` will always evaluate to 100. So, we can rewrite the code for `chunk` as:

```
chunk = min(100,refine+floor(2^11)/neq);
```

Since `neq` = $7 \times N^3$ for the full seven variable model, `refine+floor((2^11/neq))` is 37 for $N = 2$, 10 for $N = 3$, 4 for $N = 4$, 2 for $N = 5$, and 1 for $N \geq 6$. Since the solutions vectors are expected to be on the order of 40000 to 60000 elements this is not a good method to assign memory. For $N \geq 6$ in particular, memory is being reallocated at each timestep since `chunk=1`. In the memory modified code chunk was set to be 2^{16} so that solution vector are only reallocated once or twice for the entire simulation.

The other issue is with `ode15s`, a three dimensional array that stores the gradient approximations of the system at every timestep. We do not use these gradients in our post-processing for this simulation so there is no need for us to store this array. This occurs in the following line

```
dif3d = cat(3, dif3d, zeros(neq,maxk+2,chunk);
```

In the case of the full seven variable model, `neq` is $7N^3$. For `ode15s`, `maxk` is usually 5. So, for $N = 10$ `dif3d` will be an array with at least $7 \times 10^3 \times 7 \times 40000 = 1{,}960{,}000{,}000$ elements. So, this array takes at least 14.6 GB of memory for $N = 10$. So, all occurrences and calls for this array were removed from `ode15s` while creating the memory modified `ode15s`.

These issues were discovered and improvements were tested in [5], during the REU Site: Interdisciplinary Program in High Performance Computing in the Department of Mathematics and Statistics at the University of Maryland, Baltimore County (UMBC), in the summer of 2010.

3.3 Memory Modified ode15s with Dense Jacobian

`ode15s` provides the option, through `odeset` to provide a function handle to a function that computes the Jacobian analytically. This function will be called whenever `ode15s` needs to re-evaluate the value of the Jacobian.

If a function handle is not provided for the Jacobian, `ode15s` numerically approximates the Jacobian using divided differences. Even for the three variable model if $N = 10$ the Jacobian will contain 9,000,000 elements. To numerically approximate a derivative for each element with divided differences is numerically expensive and causes the number of function evaluations to be very large, even if the Jacobian is only re-evaluated a few times. So, by providing the Jacobian we expect the number of function evaluations to be much smaller.

As we can see from (2.4.8)–(2.4.17), the Jacobian is a pentadiagonal matrix that can be defined using mathematically using diagonals. Matlab has a `diag` function that can create diagonal matrices. So the Jacobian, J, can be defined with the following code:

```
J11 = diag(-g_K.*n./tau-g_Ca.*minf/tau-...
      g_Ca.*(v-v_Ca).*dminf/tau-g_s.*s./tau);
J12 = diag(-g_K*(v-v_K)/tau);
J13 = diag(g_s*(v-v_K)/tau);
J21 = diag(lambda*dninf/tau);
J22 = diag(repmat((-lambda/tau),N^3,1));
J23 = zeros(N^3,N^3);
J31 = diag(dsinf/tau_s);
J32 = zeros(N^3,N^3);
J33 = diag(repmat((-1/tau_s),N^3,1));
J   = [J11,J12,J13;J21,J22,J23;J31,J32,J33];
```

3.4 Memory Modified ode15s with Sparse Jacobian

For the three variable model, most elements in the Jacobian which is provided to `ode15s` are 0. For our simulation the Jacobian is a pentadiagonal matrix, so it contains 5 diagonal bands. For $N = 10$, there are 7000 nonzero elements of the 9,000,000 elements in the matrix. If the Jacobian is stored as a sparse matrix in Matlab, it takes advantage of the fact that most of the entries are zeros. In our three variable model for $N = 10$, a dense matrix requires 68.67 MB of storage while a sparse matrix only requires 0.09 MB of memory of storage.

Since the Jacobian is recomputed several times by `ode15s` and it is used in its computations at each time, we expect the time it takes `ode15s` to run to be less with a sparse Jacobian than with a dense Jacobian due to the efficient storage of the matrix.

Equations (2.4.8)–(2.4.17) define the blocks of the Jacobian matrix using diagonals. So we use Matlab's `spdiags` command to create the sparse diagonal matrices necessary to form the Jacobian. A sparse Jacobian, J, can be defined with the following code:

```
J11 = spdiags(-g_K.*n./tau-g_Ca.*minf/tau-g_Ca.*(v-v_Ca).*...
      dminf/tau-g_s.*s./tau,0,N^3,N^3);
J12 = spdiags(-g_K*(v-v_K)/tau,0,N^3,N^3);
J13 = spdiags(g_s*(v-v_K)/tau,0,N^3,N^3);
J21 = spdiags(lambda*dninf/tau,0,N^3,N^3);
J22 = spdiags(repmat((-lambda/tau),N^3,1),0,N^3,N^3);
J23 = sparse(N^3,N^3);
J31 = spdiags(dsinf/tau_s,0,N^3,N^3);
J32 = sparse(N^3,N^3);
J33 = spdiags(repmat((-1/tau_s),N^3,1),0,N^3,N^3);
```

```
J   = [J11,J12,J13;J21,J22,J23;J31,J32,J33];
```

3.5 Explicit ode45

Matlab's `ode45` solver uses the explicit Runge-Kutta-Fehlberg 5(4) pair of Dormand and Prince [13].

Runge-Kutta (RK) methods are one-step methods that compute the solution y_{n+1} at the next time step t_{n+1} from the solution y_n at t_n via several intermediate stages. Runge-Kutta-Fehlberg (RKF) methods estimate the local truncation error by comparing solutions y_{n+1} and $\hat{y}_{n+1}$ that are computed by RK methods of different orders of accuracy; the key to their efficiency lies in the use of shared stage values. Matlab's `ode45` solver uses a fifth order accurate RK method for the solution y_{n+1} that is the solution at the next time step, comparing it to fourth order accurate RK method $\hat{y}_{n+1}$. The coefficients for these methods can be found in [9, Table 5.2].

In `ode45`, the step size Δt is selected automatically to control the error. The method is explicit and thus does not require a Jacobian. It is not suitable for stiff problems.

3.6 Implicit ode113

Matlab's `ode113` is a variable step size, predictor-corrector algorithm, based on the Adams-Bashforth-Moulton families of linear multi-step methods of orders 1 to 12, see [13].

Adams methods are based on the integral reformulation of $M\, y'(t) = f^{(ode)}(t, y(t))$ as

$$M\, y_{n+1} = M\, y_n + \int\limits_{t_n}^{t_{n+1}} f^{(ode)}(t, y(t))\, dt. \tag{3.6.1}$$

The idea of the methods is to interpolate the integrand $f^{(ode)}(t, y(t)) = y'(t)$ by a polynomial and then integrate this polynomial exactly.

The Adams-Bashforth methods interpolate $y'(t)$ in (3.6.1) at the nodes $t_{n-p+1}, .., t_n$ (not including t_{n+1}), which results in an explicit method in time, since y_{n+1} does not appear. By contrast, the Adams-Moulton methods interpolate $y'(t)$ in (3.6.1) at the nodes $t_{n-p+1}, .., t_n, t_{n+1}$ (including t_{n+1}), which results in an implicit method in time. Tables 6.11 and 6.13 in [1] give examples of Adams-Bashforth and Adams-Moulton methods, respectively, for $p = 0, 1, 2, 3$.

In a predictor-corrector algorithm, the corrector step uses an implicit method, here Adams-Moulton. The non-linear equations in the corrector methods are solved by functional iteration, with the initial guess provided by the predictor step that uses an explicit method, here Adams-Bashforth. In `ode113`, the step size Δt and method order k are selected automatically to control the error.

Due to the use of function iteration, no Jacobian is required for the algorithm. However, for the functional iterations to be stable, the time step needs to be restricted to be relatively small in the case of stiff ODEs [1, page 413]. This makes this family of methods unsuitable for stiff problems.

The Adams-Bashforth-Moulton families are the non-stiff solvers implemented in codes such as LSODE and VODE, CVODE, and PVODE, which are used in state-of-the-art packages such as SUNDIALS.

CHAPTER 4

RESULTS

This chapter explains the results of our simulations. Section 4.1 provides the physiological results from our simulations and Section 4.2 provides the numerical performance results from our simulations. Results were run using `ode15s`,`ode45`, and `ode113` with a relative tolerance of 10^{-3} and an absolute tolerance of 10^{-5}.

4.1 Physiological Results

We simulate several different structures of slow and fast cells to study the effects of these distributions on the average interburst period of cells in the islet. All physiological studies were done with a $5 \times 5 \times 5$ computational islet with 50% slow cells and 50% fast cells.

As described in Section 2.2, the five different islet structures studied were: grouped structure with connectivity $c = 0.0928$, split grouped structure with $c = 0.1712$, layered middle structure with $c = 0.2693$, layered split structure with $c = 0.4261$, and the equally distributed structure with $c = 1.0000$.

4.1.1 Seven Variable Model

The following results are for the seven variable model described in Section 2.3. The simulations were run from 0 ms to 500000 ms or about 8.3 minutes. For all structures simulations were run for coupling strengths from 0 pS to 1000 pS.

Figures 4.1.1–4.1.6 contain the bursting behavior of cells in an islet. The bursting behavior of a cell is shown by plotting the voltage in millivolts of a cell versus time in minutes. For each islet we see the behavior of a representative slow cell and fast since we observe that cells of the same type synchronize their bursting even for a coupling strength of 100 pS.

As we can see in Figure 4.1.1, when the cells are uncoupled the behavior of a fast cell is significantly different from the behavior of a slow cell. As we would expect, the bursting behavior for both slow and fast cells will be the same no matter the islet structure.

However, as we can see in Figures 4.1.2–4.1.6 the interburst period of fast and slow cells appear to synchronize after a coupling strength of just 100 pS. There are still some slight difference in bursting behavior with lower coupling strengths such as 100 pS but these differences are no longer visible at higher coupling strengths of 1000 pS. This rapid synchronization for increasing coupling strengths is visible for all five islet structures.

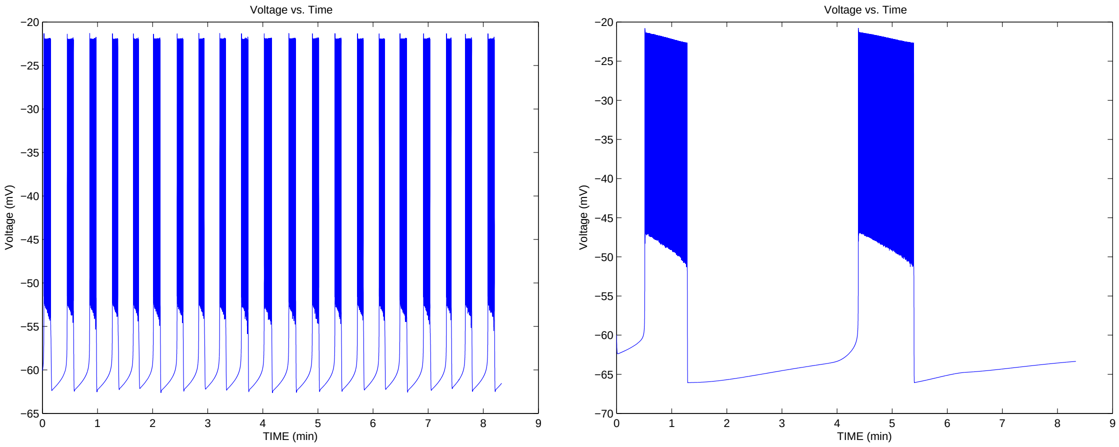
Figure 4.1.7 contains a plot of average burst period in minutes versus coupling strength in picosiemens for each islet structure. To compute the average burst period of a cell, we first find the times at which the voltage of the cell exceeds -60 mV. We then determine the time at which a burst ends by checking if it takes more than 1000 ms for the voltage to be greater than -60 mV again. The beginning of each burst is defined as the next time at which the voltage is above -60 ms. The burst period is then defined as the average of the differences between the start of bursts and the differences between the end of bursts.

We notice that the bursting behavior for the same coupling strength and cell type varies if we compare different islet structures. As demonstrated in Figure 4.1.7, this is more visible if we compare the average burst period of a representative fast cell and slow cell in each islet structure as we increase the coupling strength by increments of 100 pS. We notice that the fast cell and slow cells start at the same burst period for all structures, as we would expect. We also notice that with the exception of the grouped structure the average burst period is the same for high coupling strengths such as 1000 pS.

The difference in the structures is seen in the intermediate coupling strengths. As demonstrated in Figure 4.1.7(a), in the case of the grouped islet structure we notice that at the average burst period continues to increase until a coupling strength of 600 pS is reached, at which point the average burst period begins to decrease. Such non-monotonicities are also seen in the plots for the split grouped structure and the layered middle structure. However the average bursting begins to decrease much earlier in coupling strength for the other two islet structures, at 300 pS for the split grouped structure and at 200 pS for the layered middle structure. The other two islet structures, the layered split and equally distributed structures, are completely monotonic after the synchronization at 100 pS.

Figure 1.1.1(a) in Chapter 1 contains a plot of peak coupling strength versus connectivity of the islet. The peak coupling strength is defined as the coupling strength at which the average burst period is largest after the synchronization of slow and fast cells. This plot demonstrates that as connectivity increases, the peak coupling strength decreases.

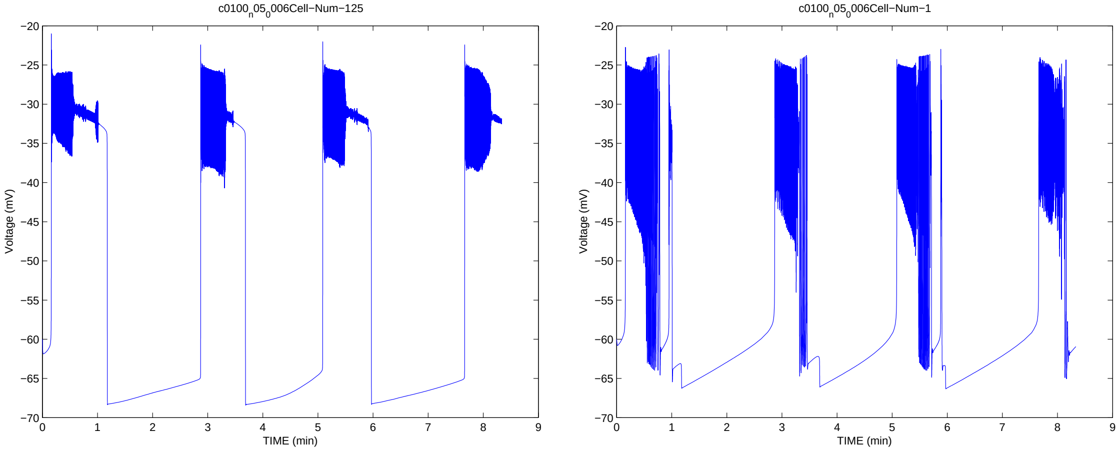
If we compare the connectivities of these structures, we notice that the structure with the largest connectivity, the grouped structure, also has the largest interval of increasing burst period. As connectivity increases, this interval decreases until a connectivity of 0.4261 is reached with the layered split structure and the bursting is entirely monotonic. So, it appears that as connectivity increases, the non-monotonicity of average bursting decreases. This means that for intermediate coupling strengths, cell behavior is affected by islet structure.



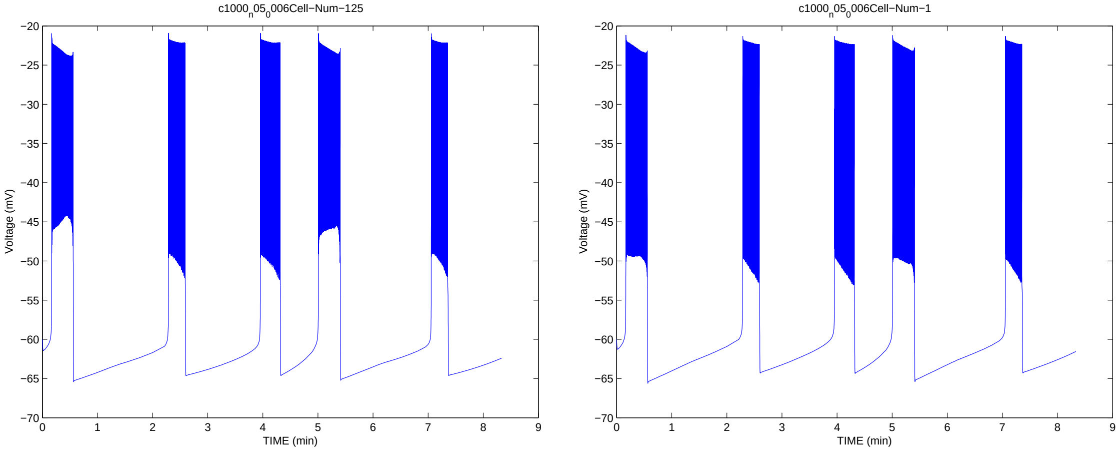
(a) fast cell

(b) slow cell

Figure 4.1.1 Bursting behavior of uncoupled cells for the seven variable model.

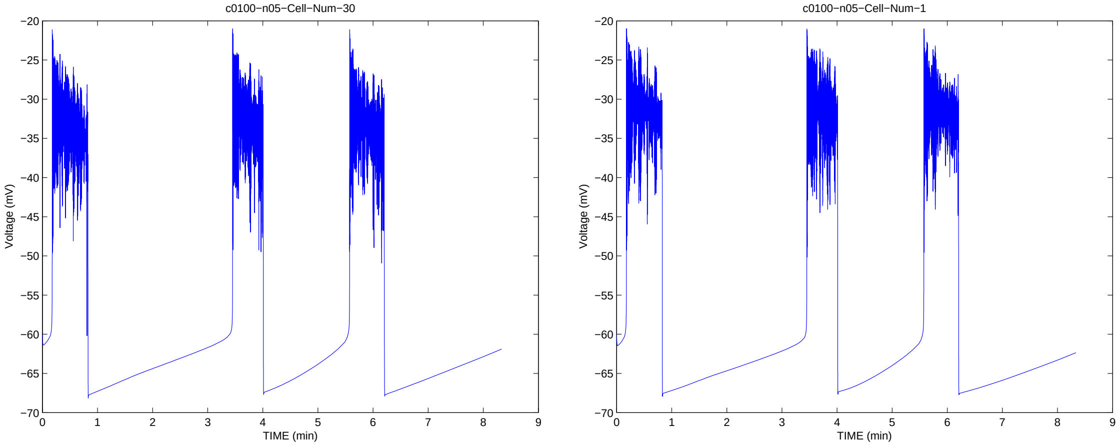


(a) fast cell (b) slow cell

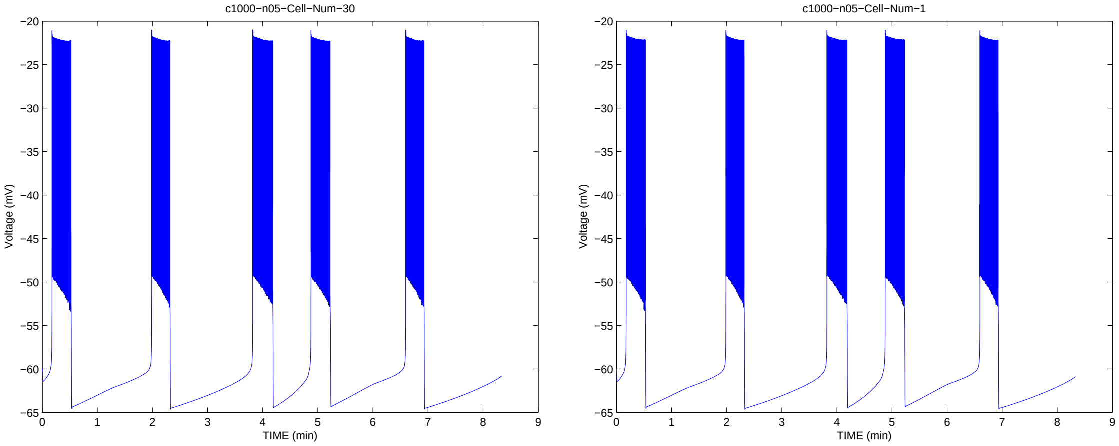


(c) fast cell (d) slow cell

Figure 4.1.2 Bursting behavior of cells in an islet with grouped structure ($c = 0.0928$) for the seven variable model. (a) and (b) islet with coupling strength of 100 pS. (c) and (d) islet with coupling strength of 1000 pS.

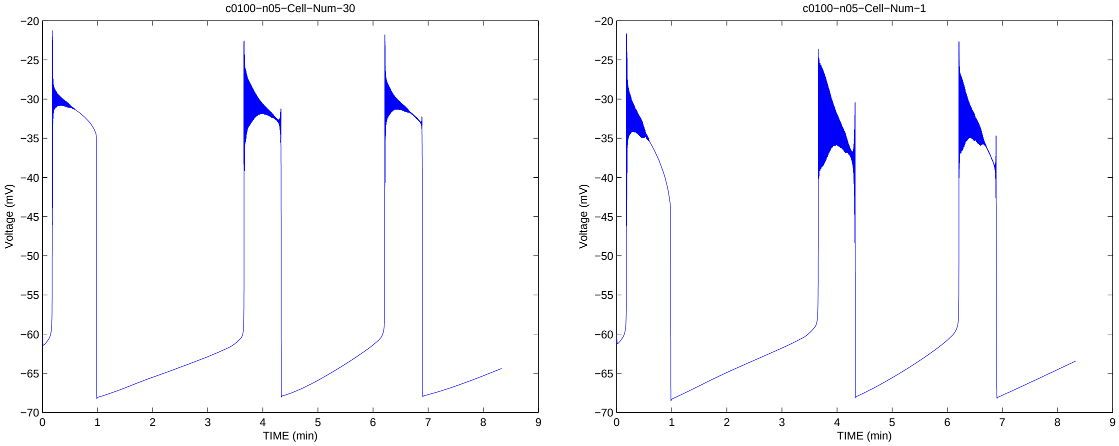


(a) fast cell (b) slow cell

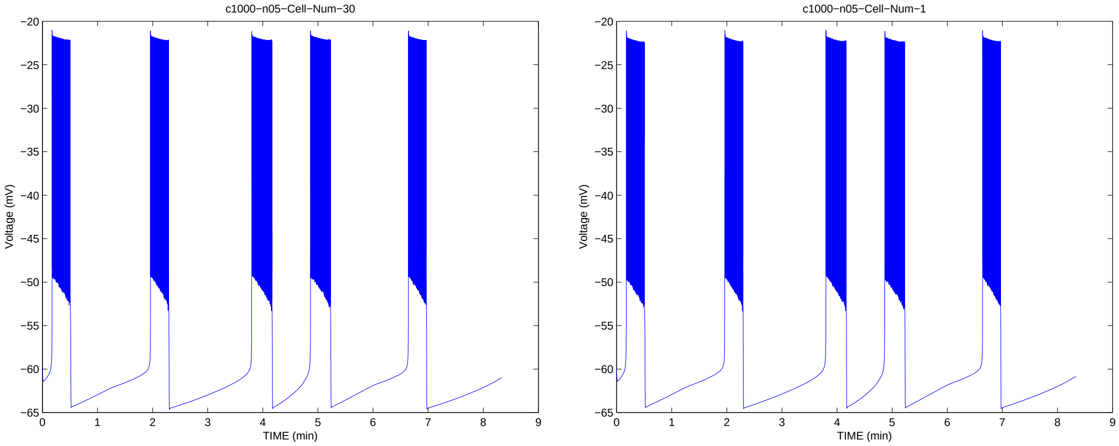


(c) fast cell (d) slow cell

Figure 4.1.3 Bursting behavior of cells in an islet with split grouped structure ($c = 0.1712$) for the seven variable model. (a) and (b) islet with coupling strength of 100 pS. (c) and (d) islet with coupling strength of 1000 pS.

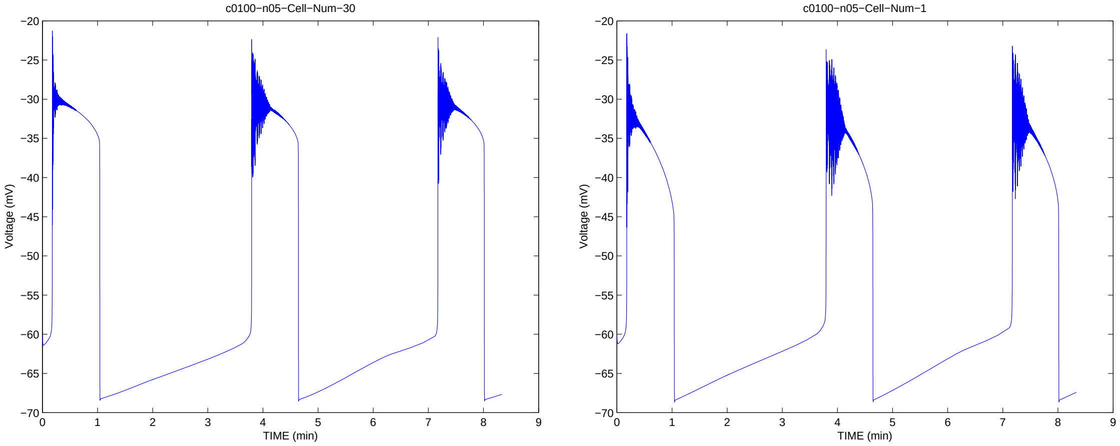


(a) fast cell (b) slow cell

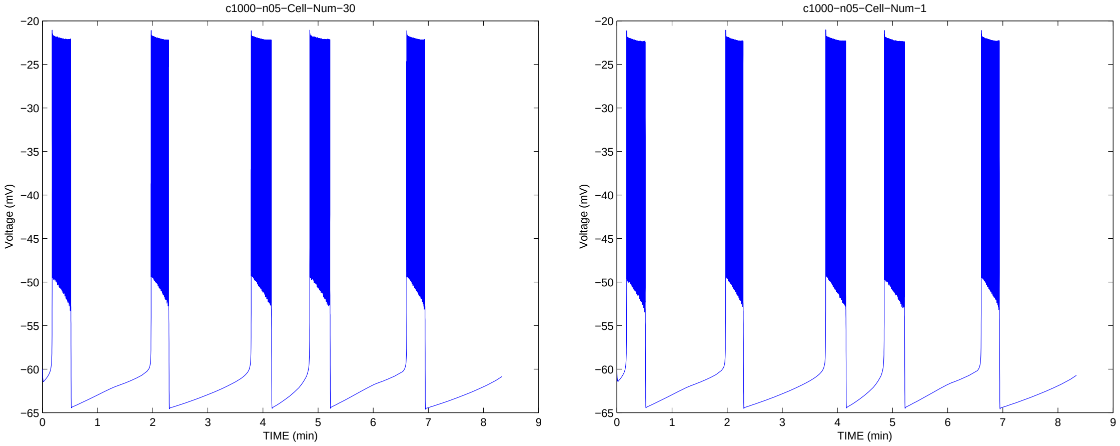


(c) fast cell (d) slow cell

Figure 4.1.4 Bursting behavior of cells in an islet with layered middle structure ($c = 0.2693$) for the seven variable model. (a) and (b) islet with coupling strength of 100 pS. (c) and (d) islet with coupling strength of 1000 pS.

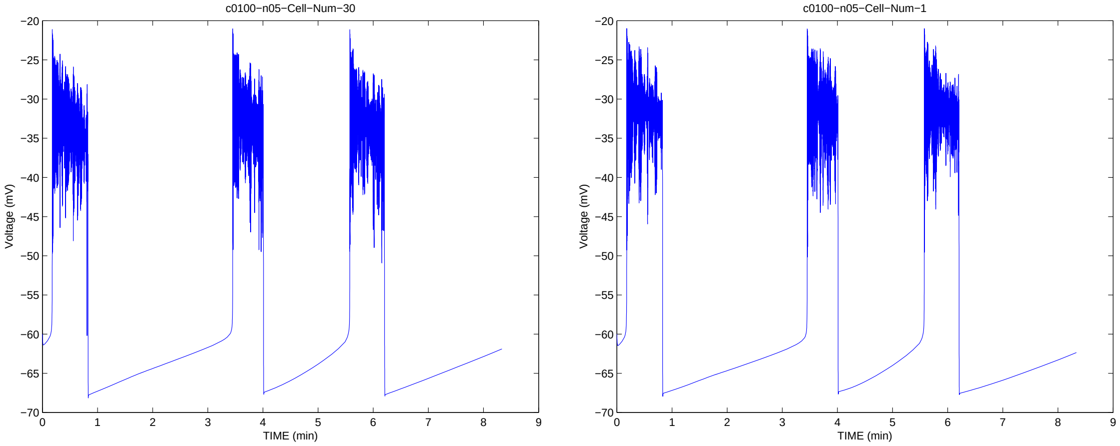


(a) fast cell (b) slow cell

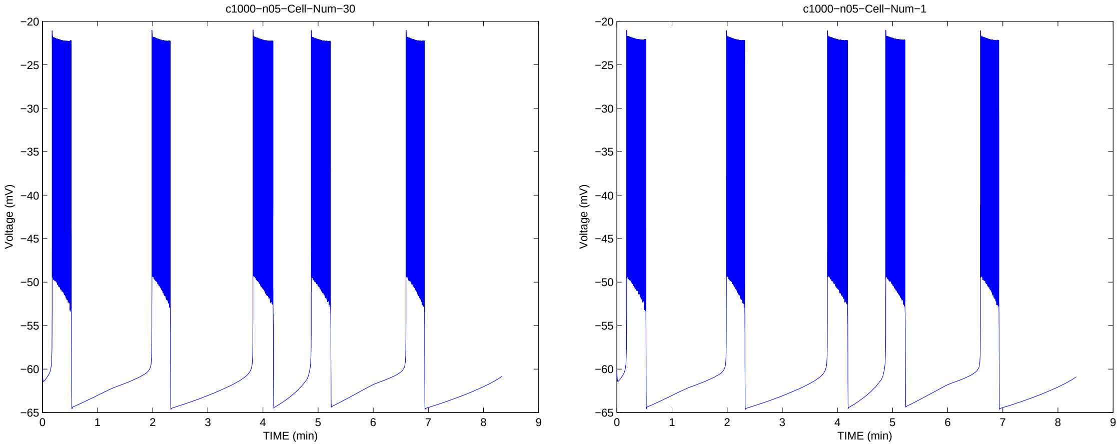


(c) fast cell (d) slow cell

Figure 4.1.5 Bursting behavior of cells in an islet with layered split structure ($c = 0.4261$) for the seven variable model. (a) and (b) islet with coupling strength of 100 pS. (c) and (d) islet with coupling strength of 1000 pS.

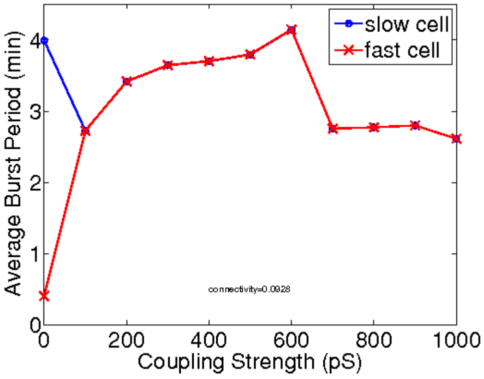


(a) fast cell (b) slow cell

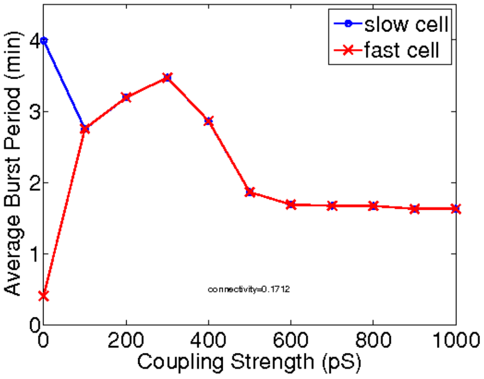


(c) fast cell (d) slow cell

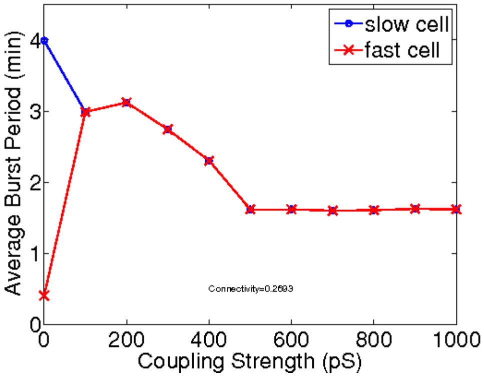
Figure 4.1.6 Bursting behavior of cells in an islet with equally distributed structure ($c = 1$) for the seven variable model. (a) and (b) islet with coupling strength of 100 pS. (c) and (d) islet with coupling strength of 1000 pS.



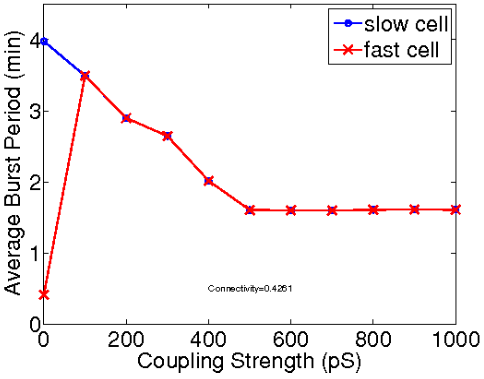
(a) $c = 0.0928$



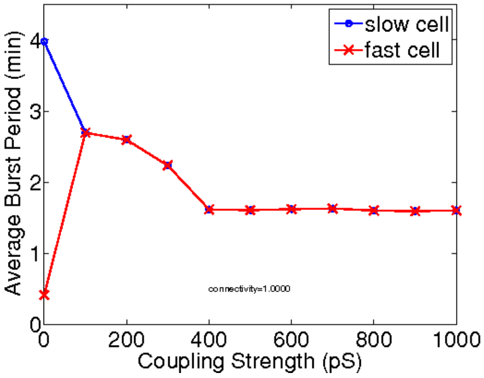
(b) $c = 0.1712$



(c) $c = 0.2693$



(d) $c = 0.4261$



(e) $c = 1.0000$

Figure 4.1.7 Average burst period over a range of coupling strengths for cells for the seven variable model: (a) grouped structure, (b) split grouped structure, (c) layered middle structure, (d) layered split structure, (e) equally distributed structure.

4.1.2 Three Variable Model

The following results are for the three variable model described in Section 2.4. The simulations were run from 0 ms to 200000 ms or about 3.3 minutes.

Note that in the case of the three variable model a unitless coupling strength as used in [6] can be multiplied by $\frac{C_m}{\tau} = 265$ pS to obtain the coupling strength in pS to compare with the seven variable model. For the grouped structure and split grouped structure simulations were run for coupling strength from 0 to 15 with increments of 1. For the layered middle structure simulations are run for coupling strengths from 0 to 3 with increments of 0.1. For the layered split structure simulations are run for coupling strength 0 to 2 with increments of 0.1. For the equally distributed structure simulations are run for coupling strengths from 0 to 1 with increments of 0.1.

Figures 4.1.8–4.1.13 contain the bursting behavior of cells in an islet. The bursting behavior of a cell is shown by plotting the voltage in millivolts of a cell versus time in minutes. For each islet we see the behavior of a representative slow cell and a fast cell since we observe that cells of the same type synchronize their bursting even for a coupling strength of 0.1.

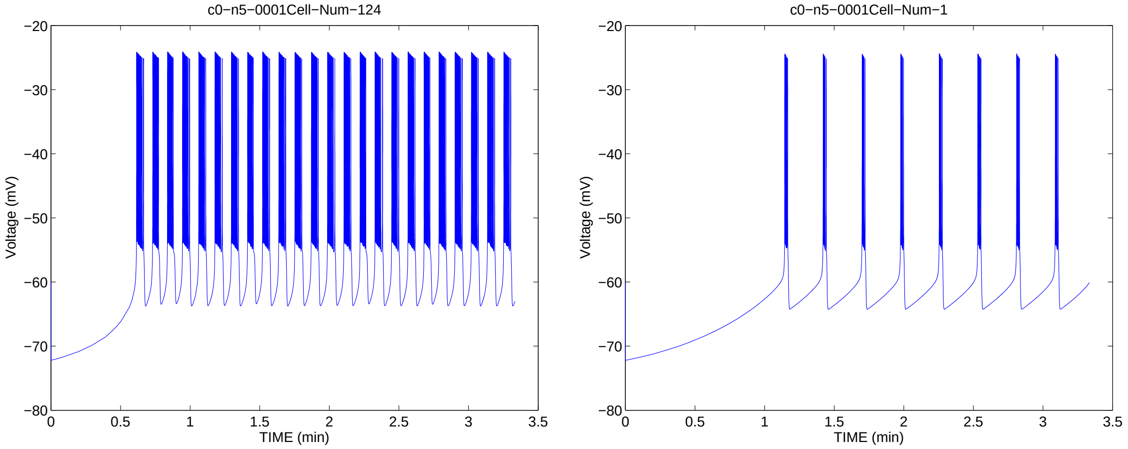
As we can see in Figure 4.1.8, when the cells are uncoupled the behavior of a fast cell is significantly different to the behavior of a slow cell. We also note that the bursting behavior for uncoupled cells are the same for different islet structures. This was true for the seven variable model, however, we observe different burst periods and amplitudes for each model.

However, as we can see by comparing (a) and (b) in each of Figures 4.1.9–4.1.13, the interburst period is already synchronized when there is a coupling strength of 0.1 between cells. We observe that although the interburst period is the same at such a coupling strength there is still a difference in the behavior of the bursting. For instance, if we compare Figure 4.1.9(a) with Figure 4.1.9(b) we observe an initial short burst for the slow cell that does not occur for the fast cell. Also, we note that the bursts do not go above −30 mV for the slow cell while for the fast cell the bursts do go above −30 mV. For a stronger coupling strength of 1 these differences between slow and fast cells are far less visible. In fact, for the equally distributed structure the slow and fast cells appear to have the same bursting behavior for a coupling strength of 1 as we can see by comparing Figure 4.1.13(c) and Figure 4.1.13(d).

Figure 4.1.14 contains a plot of average burst period in minutes versus coupling strength in picosiemens for each islet structure. To compute the average burst period of a cell we first find the times at which the voltage of the cell exceeds −60 mV. We then determine the time at which a burst ends by checking if it takes more than 100 ms for the voltage to be greater than −60 mV again. The beginning of each burst is defined as the next time at which the voltage is above -60 ms. The burst period is then defined as the average of the differences between the start of bursts and the differences between the end of bursts.

We also observe that the burst period and bursting behavior for the same coupling strength varies for different islet structures. Figure 4.1.14 contains a plot for each of the five islet structures which compares the average burst period of a representative fast cell and slow cell for a range of coupling strengths. As we previously noted, the burst period is already synchronized at a coupling strength of 0.1 for all five islet structures. For each islet structure the average burst period remains constant above a certain coupling strength. This coupling strength varies for each islet structure. The average burst period for very strong coupling strengths also varies for different islet structures.

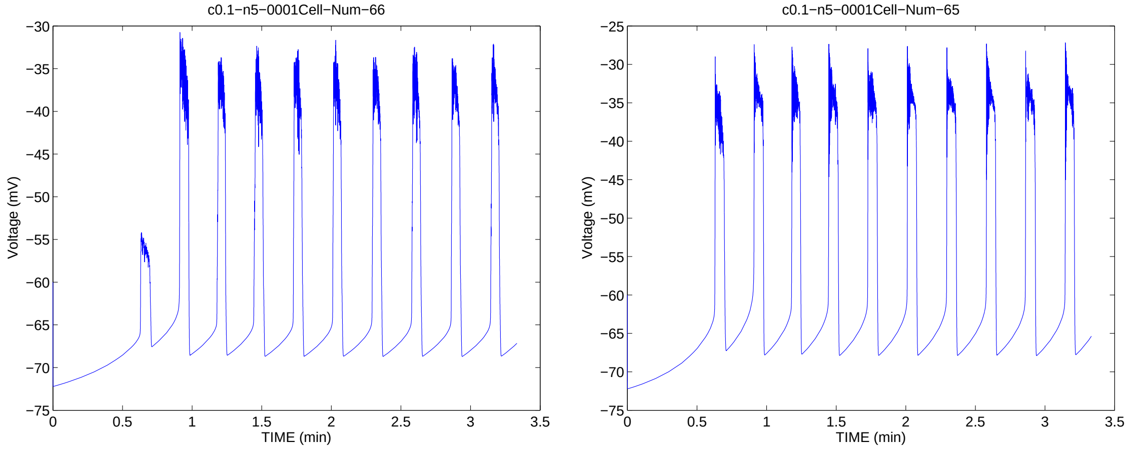
Another difference between the bursting behavior of cells for different islet structures is in the intermediate coupling strengths. For all of the islet structures except the equally distributed we observe a non-monotonicity after synchronization for interme-



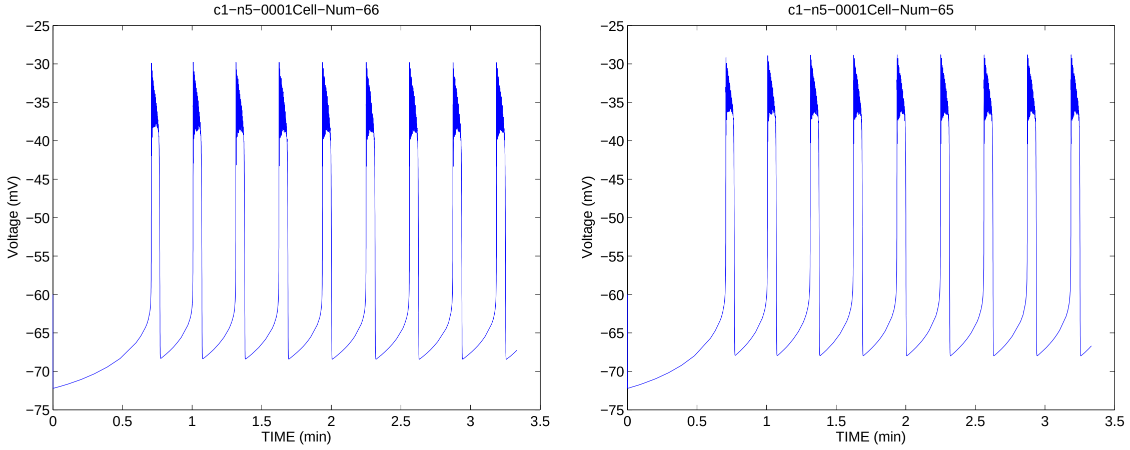
(a) fast cell (b) slow cell

Figure 4.1.8 Bursting behavior of uncoupled cells for the three variable model.

diate coupling strengths. In fact, for the grouped structure, split grouped structure, layered middle structure, and layered middle structure the burst period is longer for intermediate coupling strengths than for an uncoupled slow cell. Figure 1.1.1(b) in Chapter 1 contains a plot of peak coupling strength versus connectivity of the islet. The peak coupling strength is defined as the coupling strength at which the average burst period is slowest after the synchronization of slow and fast cells. This plot demonstrates the effect of the connectivity of an islet structure on the peak coupling strength. As we observed for seven variable model, in the three variable model as the connectivity of the islet increases the peak coupling strength decreases.

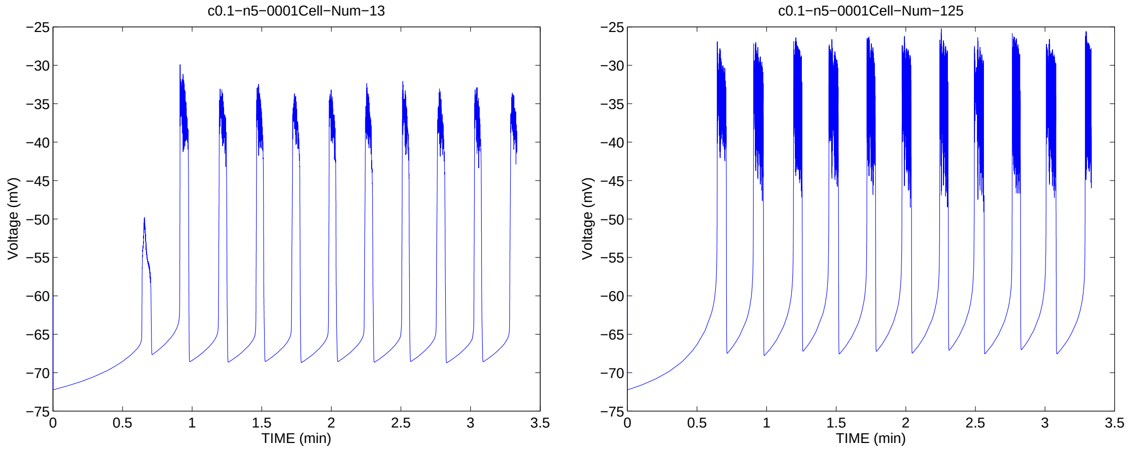


(a) fast cell (b) slow cell

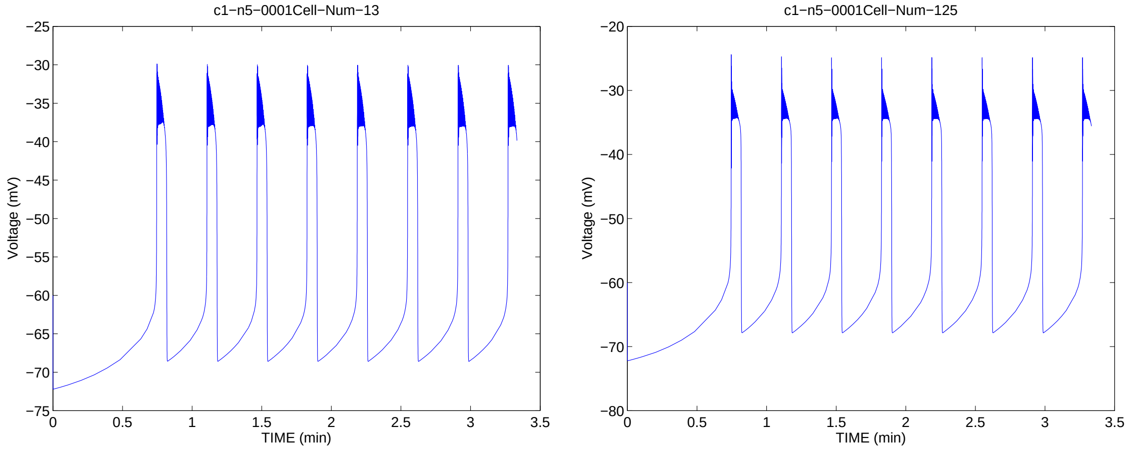


(c) fast cell (d) slow cell

Figure 4.1.9 Bursting behavior of cells in an islet with grouped structure ($c = 0.0928$) for the three variable model. (a) and (b) islet with coupling strength of 0.1. (c) and (d) islet with coupling strength of 1.

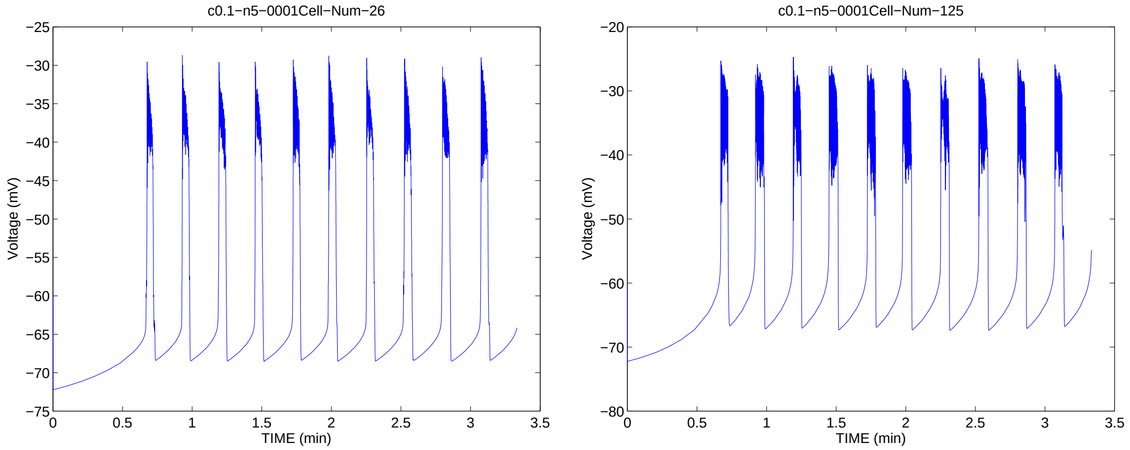


(a) fast cell (b) slow cell

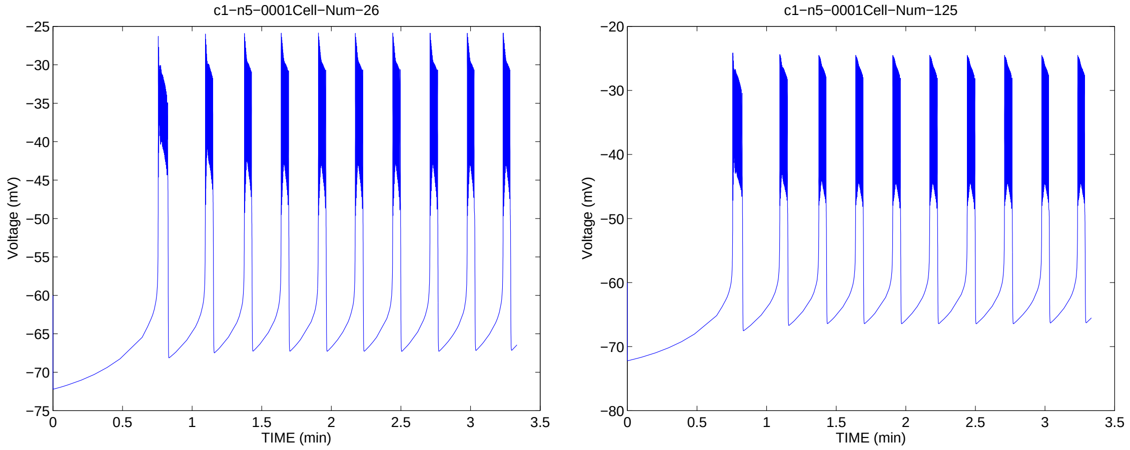


(c) fast cell (d) slow cell

Figure 4.1.10 Bursting behavior of cells in an islet with split grouped structure ($c = 0.1712$) for the three variable model. (a) and(b) islet with coupling strength of 0.1. (c) and (d) islet with coupling strength of 1.

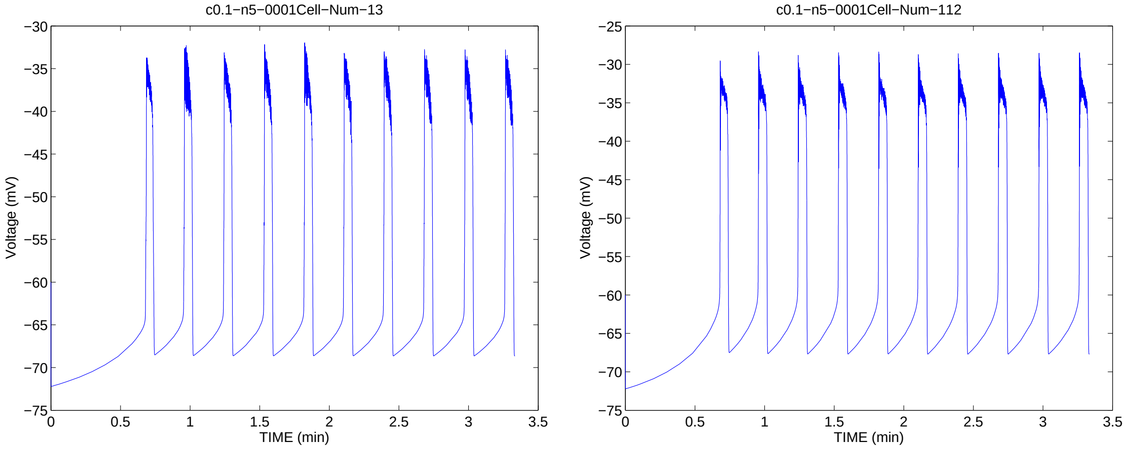


(a) fast cell (b) slow cell

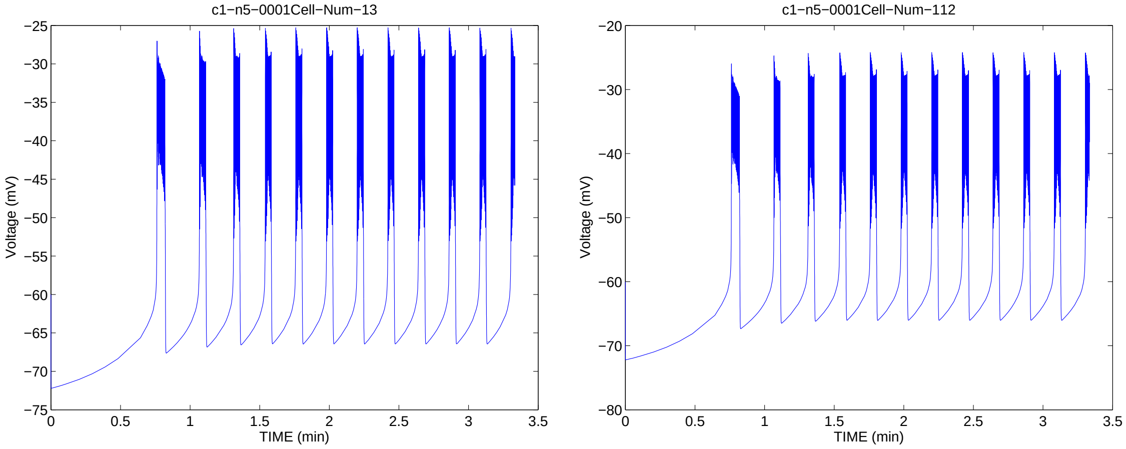


(c) fast cell (d) slow cell

Figure 4.1.11 Bursting behavior of cells in an islet with layered middle structure ($c = 0.2693$) for the three variable model. (a) and (b) islet with coupling strength of 0.1. (c) and (d) islet with coupling strength of 1.

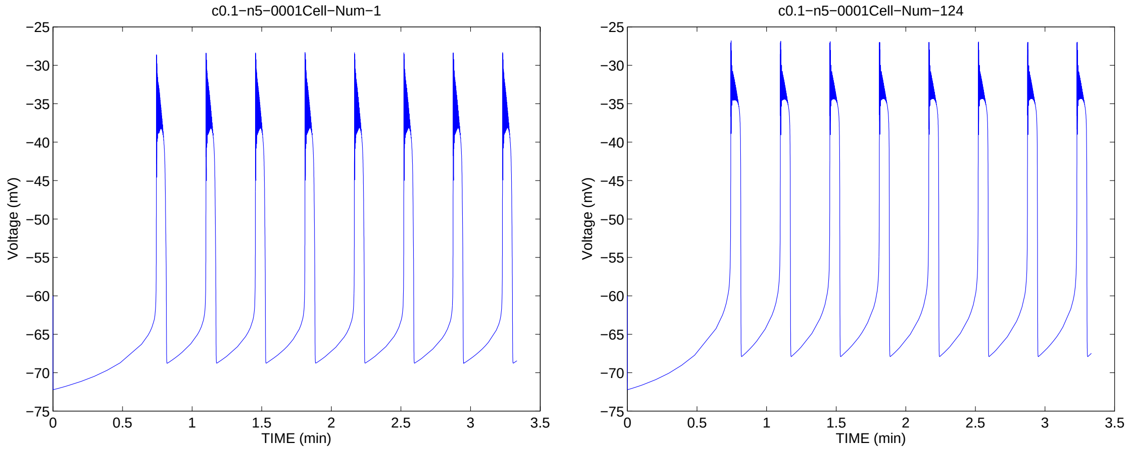


(a) fast cell (b) slow cell

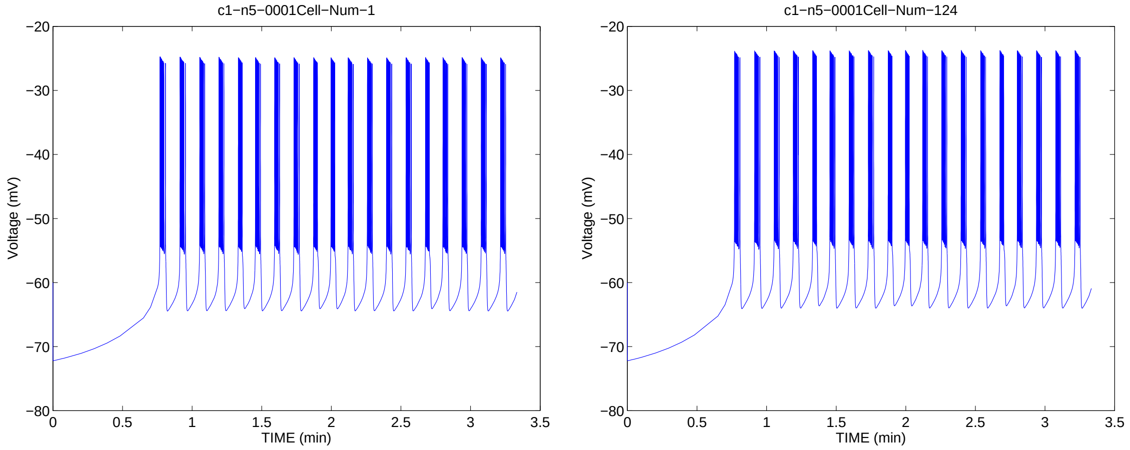


(c) fast cell (d) slow cell

Figure 4.1.12 Bursting behavior of cells in an islet with layered split structure ($c = 0.4261$) for the three variable model. (a) and (b) islet with coupling strength of 0.1. (c) and (d) islet with coupling strength of 1.

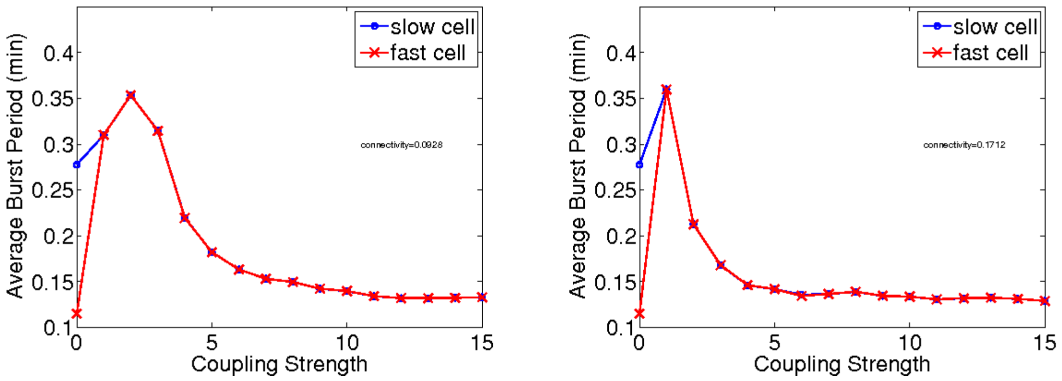


(a) fast cell (b) slow cell

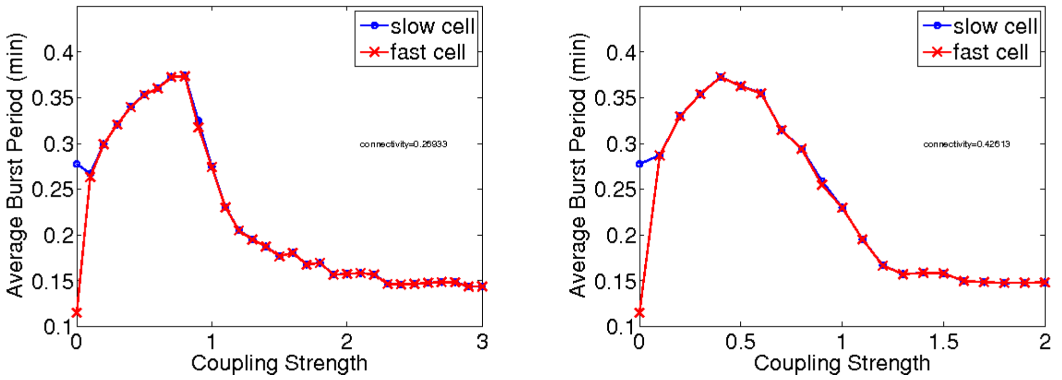


(c) fast cell (d) slow cell

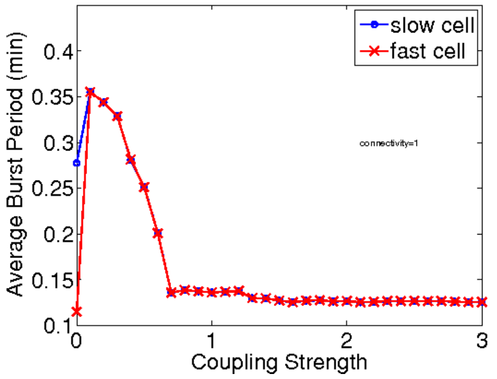
Figure 4.1.13 Bursting behavior of cells in an islet with equally distributed structure ($c = 1$) for the three variable model. (a) and (b) islet with coupling strength of 0.1. (c) and (d) islet with coupling strength of 1.



(a) $c = 0.0928$ (b) $c = 0.1712$



(c) $c = 0.2693$ (d) $c = 0.4261$



(e) $c = 1.0000$

Figure 4.1.14 Average burst period over a range of coupling strengths for cells for the three variable model: (a) grouped structure, (b) split grouped structure, (c) layered middle structure, (d) layered split structure, (e) equally distributed structure.

4.2 Numerical Performance Results

The following subsections contain the numerical results for the three variable model. There are subsections for each of the following versions of Matlab's `ode15s`: original `ode15s` with numerical Jacobian, the memory modified `ode15s` with numerical Jacobian, the memory modified `ode15s` with dense Jacobian, and the memory modified `ode15s` with sparse Jacobian. In each of these subsections, we have results for all five islet structure for uncoupled cells and islets with coupling strengths of 0.1 and 1. For the grouped structure and split grouped structure we run simulations from 0 to 15 with increments of 1. For the layered middle structure we run simulations from 0 to 3 with increments of 0.1. For the layered split structure we run simulations from to 1 with increments of 0.1. For the equally distributed structure we run simulations from 0 to 1 with increments of 0.1. The following numerical results are provided for each implementation:

time: runtime in seconds to complete one simulation,

nsteps: number of successful (i.e., accepted) time steps,

nfailed: number of failed (i.e., rejected) time steps,

nfevals: number of times that the right-hand side of the ODE was evaluated,

npds: number of times that partial derivatives, i.e., the Jacobian matrix, was computed,

ndecomps: number of LU decompositions,

nsolves: number of linear solves.

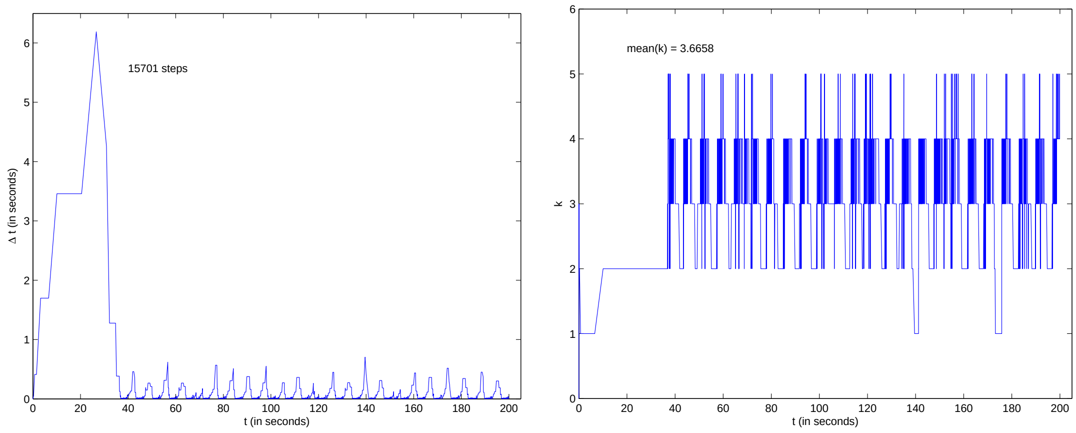
4.2.1 Original ode15s with Numerical Jacobian

Table 4.2.1 contains the performance results for an islet with uncoupled cells using Original `ode15s` with numerical Jacobian, as specified in Section 3.1. We observe that there is very little variation in nsteps, nfailed, npds, ndecomps, and nsolves, as N increases. This means that the reaction terms drive the system.

The number of function evaluations, however, increases as the size of the system increases. We expect this since most of the calls to the ODE function are during the evaluation of the Jacobian matrix, since divided differences, which call the ODE function, are used to compute each element of the Jacobian. So, for a larger system there will be more function evaluations. As we would expect, the runtime increases as N increases.

As discussed in Chapter 3, the order of the Numerical Differentiation Formula, k, and the time step size of the method, Δt, change while `ode15s` solves the system of ODEs. Figure 4.2.1(a) contains the time step size over time. As we can see, the time step size initially grows for the first few time steps, reaching a time step size of over 6 seconds, and then decreases to less than a second for the rest of the simulation. Figure 4.2.1(b) contains the order of the method over time. The method order varies throughout the simulation, but the mean order is 3.6658.

In Table 4.2.2 only the times are provided for all of the other simulations in this implementation. Just as with uncoupled cells, simulations for islets with coupling strengths of 0.1 and 1 also have an increase in runtime as N increases for all five islet structures. For a coupling strength of 0.1 we observe that the runtime decreases as the connectivity of the islet structure increases. However, for the stronger coupling strength of 1 the differences in runtime between different islet structures is not evident.



(a) (b)

Figure 4.2.1 Numerical performance data for uncoupled cells in a 10 × 10 × 10 computational islet using original ode15s with numerical Jacobian: (a) time step Δt vs. t, (b) method order k vs. t.

N	time (seconds)	nsteps	nfailed	nfevals	npds	ndecomps	nsolves
2	18	15,700	2,866	46,931	627	4,927	31,255
3	43	15,717	2,880	83,325	635	4,938	31,254
4	114	15,700	2,866	152,267	627	4,927	31,255
5	295	15,700	2,866	267,008	627	4,927	31,255
6	765	15,700	2,866	438,179	627	4,927	31,255
7	2,140	15,752	2,864	688,545	638	4,939	31,404
8	3,498	15,700	2,866	994,955	627	4,927	31,255
9	8,689	15,752	2,864	1,427,349	638	4,939	31,404
10	13,815	15,700	2,866	1,912,883	627	4,927	31,255

Table 4.2.1 Performance results for uncoupled cells using original ode15s with numerical Jacobian.

(a) Coupling Strength 0.1

N	Grouped	Split Grouped	Layered Middle	Layered Split	Equally Distributed
2	11	11	9	12	5
3	26	21	24	16	8
4	74	78	82	59	24
5	244	256	194	166	57
6	693	631	533	453	175
7	2,274	1,885	1,816	1,354	508
8	4,008	2,843	2,843	2,302	995
9	11,103	9,036	7,520	6,189	1,512
10	15,967	13,538	12,558	9,861	4,096

(b) Coupling Strength 1

N	Grouped	Split Grouped	Layered Middle	Layered Split	Equally Distributed
2	8	10	5	10	10
3	16	15	23	15	20
4	21	36	40	44	48
5	68	62	123	114	117
6	261	121	236	302	295
7	1,034	472	1,118	865	806
8	1,984	735	1,452	1,265	1,545
9	5,871	3,834	3,530	4,089	4,432
10	10,226	6,216	5,655	6,549	7,649

Table 4.2.2 Runtimes for original ode15s with numerical Jacobian.

(a) Coupling Strength 0.1

N	time (seconds)	nsteps	nfailed	nfevals	npds	ndecomps	nsolves
2	5	4,969	765	14,290	132	1,469	10,989
3	8	4,394	591	20,104	124	1,205	9,935
4	24	5,808	674	36,327	120	1,440	13,166
5	57	6,078	596	56,548	114	1,400	13,683
6	175	6,967	712	97,942	126	1,648	16,167
7	508	7,190	740	147,189	127	1,666	16,378
8	995	7,944	868	256,332	155	1,885	18,096
9	1,512	5,851	569	304,290	133	1,336	13,285
10	4,096	8,072	814	489,772	157	1,810	18,614

(b) Coupling Strength 1

N	time (seconds)	nsteps	nfailed	nfevals	npds	ndecomps	nsolves
2	10	8,623	1,678	26,717	333	2,834	18,391
3	20	8,802	1,636	43,001	299	2,820	18,482
4	48	8,454	1,646	78,612	315	2,713	17,816
5	117	8,604	1,624	129,869	297	2,764	18,196
6	295	9,185	1,575	168,132	229	2,703	19,510
7	806	9,911	1,253	143,332	118	2,349	21,791
8	1,545	11,290	1,174	150,836	82	2,327	24,801
9	4,432	12,119	1,050	210,191	84	2,307	26,398
10	7,649	12,807	1,062	297,870	90	2,331	27,779

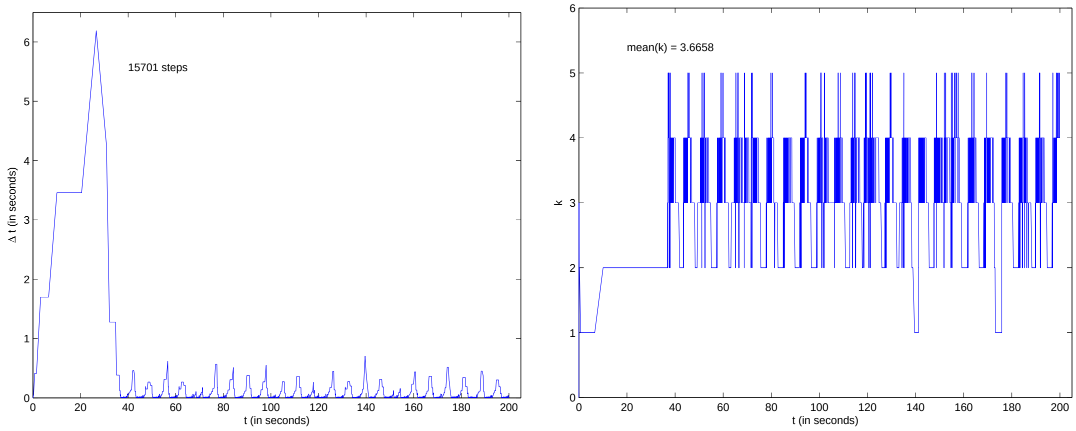
Table 4.2.3 Performance results for islet with equally distributed structure using original ode15s with numerical Jacobian.

4.2.2 Memory Modified Ode15s with Numerical Jacobian

Table 4.2.4 contains the performance results for an islet with uncoupled cells using memory modified `ode15s` with numerical Jacobian, as specified in 3.2. Since the only differences between this version of `ode15s` and the original `ode15s` is in the allocation of memory, the only difference in the performance results in the runtime. Just as in the previous subsection, the runtime increases as N increases. However, as we would expect, the runtimes are much faster for this memory modified `ode15s` than the original `ode15s` since it manages memory more efficiently.

Since the numerical results are identical, the time step size over time and method order over time should be the same for the memory modified `ode15s` and the original `ode15s`. As we can see by comparing Figure 4.2.2(a) with Figure 4.2.1(a) and Figure 4.2.2(b) with Figure 4.2.1(b), time step size and method order are indeed the same for both of these implementations.

Again, Table 4.2.5 contains only the runtimes for each of the remaining simulations. Just as for the uncoupled cells, the islets with 0.1 and 1 coupling between cells increases in runtime as N increases. Unlike the original `ode15s`, for the memory modified `ode15s` there does not appear to be any relationship between runtime and connectivity, even for the weaker coupling strength of 0.1.



(a) (b)

Figure 4.2.2 Numerical performance data for uncoupled cells in a 10 × 10 × 10 computational islet using memory modified ode15s with numerical Jacobian: (a) time step Δt vs. t, (b) method order k vs. t.

N	time (seconds)	nsteps	nfailed	nfevals	npds	ndecomps	nsolves
2	17	15,700	2,866	46,931	627	4,927	31,255
3	32	15,717	2,880	83,325	635	4,938	31,254
4	60	15,700	2,866	152,267	627	4,927	31,255
5	126	15,700	2,866	267,008	627	4,927	31,255
6	282	15,700	2,866	438,179	627	4,927	31,255
7	654	15,752	2,864	688,545	638	4,939	31,404
8	1,264	15,700	2,866	994,955	627	4,927	31,255
9	2,511	15,752	2,864	1,427,349	638	4,939	31,404
10	5,046	15,700	2,866	1,912,883	627	4,927	31,255

Table 4.2.4 Performance results for uncoupled cells using memory modified ode15s with numerical Jacobian.

(a) Coupling Strength 0.1					
N	Grouped	Split Grouped	Layered Middle	Layered Split	Equally Distributed
2	10	5	9	5	5
3	19	8	16	8	8
4	27	17	28	16	16
5	55	31	46	31	31
6	127	85	118	85	86
7	307	198	285	199	199
8	638	458	591	460	458
9	1,211	675	1,037	689	676
10	2,479	1,848	2,321	1,863	1,848

(b) Coupling Strength 1					
N	Grouped	Split Grouped	Layered Middle	Layered Split	Equally Distributed
2	8	9	5	9	10
3	12	17	16	16	17
4	14	32	19	32	33
5	32	64	35	65	64
6	88	131	78	134	131
7	249	239	258	240	239
8	561	461	443	466	467
9	1,149	933	853	940	936
10	2,585	2,057	1,762	2,060	2,061

Table 4.2.5 Runtimes for memory modified ode15s with numerical Jacobian.

(a) Coupling Strength 0.1

N	time (seconds)	nsteps	nfailed	nfevals	npds	ndecomps	nsolves
2	5	4,969	765	14,290	132	1,469	10,989
3	8	4,394	591	20,104	124	1,205	9,935
4	16	5,808	674	36,327	120	1,440	13,166
5	31	6,078	596	56,548	114	1,400	13,683
6	86	6,967	712	97,942	126	1,648	16,167
7	199	7,190	740	147,189	127	1,666	16,378
8	458	7,944	868	256,332	155	1,885	18,096
9	676	5,851	569	304,290	133	1,336	13,285
10	1,848	8,072	814	489,772	157	1,810	18,614

(b) Coupling Strength 1

N	time (seconds)	nsteps	nfailed	nfevals	npds	ndecomps	nsolves
2	10	8,623	1,678	26,717	333	2,834	18,391
3	17	8,802	1,636	43,001	299	2,820	18,482
4	33	8,454	1,646	78,612	315	2,713	17,816
5	64	8,604	1,624	129,869	297	2,764	18,196
6	131	9,185	1,575	168,132	229	2,703	19,510
7	239	9,911	1,253	143,332	118	2,349	21,791
8	467	11,290	1,174	150,836	82	2,327	24,801
9	936	12,119	1,050	210,191	84	2,307	26,398
10	2,061	12,807	1,062	297,870	90	2,331	27,779

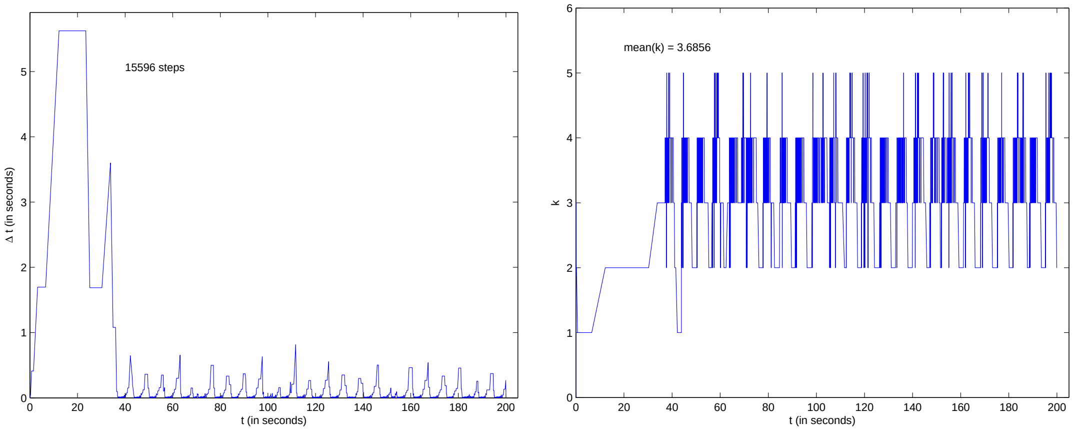
Table 4.2.6 Performance results for islet with equally distributed structure using memory modified ode15s with numerical Jacobian.

4.2.3 Memory Modified Ode15s with Dense Jacobian

Table 4.2.7 contains the performance result for an islet with uncoupled cells using memory modified `ode15s` with dense Jacobian, as specified in 3.3. When compared to the performance results for the memory modified `ode15s` with numerical Jacobian, we see slight improvements in nsteps, nfailed, ndecomps, and nsolves along with a huge improvement in nfevals; these equal exactly the nsolves now (aside from two extra evaluations), reflecting exactly one function evaluation per linear solve inside the Newton method. Since for this implementation we provide an analytic Jacobian, `ode15s` does not need to numerically approximate the Jacobian using divided differences. Since each divided difference makes several calls to the ODE function, providing the Jacobian significantly reduces the number of calls to the ODE function. Since the analytic Jacobian is more exact than a numerically approximated Jacobian, slight improvements in nsteps, nfailed, ndecomps, and nsolves are to be expected.

Figure 4.2.3(a) contains the time step size over time and Figure 4.2.3(b) contains the method order over time. The performance results are better when a dense Jacobian is provided, in particular, the number of steps is smaller when a Jacobian is provided. So, we would expect the time step sizes to be larger. If we compare Figure 4.2.3(a) with Figure 4.2.2(a), we see that the time step sizes are generally larger when the Jacobian is provided. We also note that the mean method order is larger when dense Jacobian is provided since the mean with numerical Jacobian is 3.6658 and with dense Jacobian it is 3.6856.

Table 4.2.8 contains only the runtime for all of the other simulations in this implementation. The runtime increases as N increases for coupling strengths of 0.1 and 1, just as it does for uncoupled cells. Just as for the memory modified `ode15s` with numerical Jacobian, there is no relationship between runtime and connectivity of islet structure.



(a) (b)

Figure 4.2.3 Numerical performance data for uncoupled cells in a 10 × 10 × 10 computational islet using memory modified ode15s with dense Jacobian: (a) time step Δt vs. t, (b) method order k vs. t.

N	time (seconds)	nsteps	nfailed	nfevals	npds	ndecomps	nsolves
2	13	15,595	2,850	30,936	626	4,873	30,934
3	16	15,552	2,858	30,887	647	4,877	30,885
4	23	15,595	2,850	30,936	626	4,873	30,934
5	43	15,595	2,850	30,936	626	4,873	30,934
6	116	15,595	2,850	30,936	626	4,873	30,934
7	303	15,546	2,922	31,136	657	4,951	31,134
8	630	15,595	2,850	30,936	626	4,873	30,934
9	1,394	15,546	2,922	31,136	657	4,951	31,134
10	3,023	15,595	2,850	30,936	626	4,873	30,934

Table 4.2.7 Performance results for uncoupled cells using memory modified ode15s with dense Jacobian.

(a) Coupling Strength 0.1

N	Grouped	Split Grouped	Layered Middle	Layered Split	Equally Distributed
2	9	4	4	4	4
3	15	5	5	5	5
4	21	9	9	9	9
5	37	16	15	16	16
6	100	48	44	51	48
7	258	134	121	127	119
8	565	292	291	283	288
9	1,079	441	440	440	440
10	2,349	1,303	1,306	1,317	1,304

(b) Coupling Strength 1

N	Grouped	Split Grouped	Layered Middle	Layered Split	Equally Distributed
2	7	7	7	7	7
3	11	9	9	9	9
4	7	13	13	13	13
5	17	26	23	25	25
6	58	64	57	63	70
7	184	179	164	166	168
8	469	344	347	362	354
9	939	743	745	742	741
10	2,121	1,603	1,604	1,620	1,604

Table 4.2.8 Runtimes for memory modified ode15s and dense Jacobian.

(a) Coupling Strength 0.1

N	time (seconds)	nsteps	nfailed	nfevals	npds	ndecomps	nsolves
2	4	4,953	788	11,085	140	1,490	11,083
3	5	4,354	600	9,931	135	1,219	9,929
4	9	5,728	659	12,980	122	1,417	12,978
5	16	6,100	611	13,776	123	1,412	13,774
6	48	6,972	746	16,293	135	1,672	16,291
7	119	7,178	744	16,424	135	1,680	16,422
8	288	7,910	863	18,295	160	1,883	18,293
9	440	5,856	581	13,491	138	1,360	13,489
10	1,304	8,098	829	18,402	158	1,854	18,400

(b) Coupling Strength 1

N	time (seconds)	nsteps	nfailed	nfevals	npds	ndecomps	nsolves
2	7	8,768	1,641	18,498	292	2,814	18,496
3	9	8,673	1,597	18,068	275	2,742	18,066
4	13	8,676	1,652	18,151	317	2,754	18,149
5	25	8,591	1,637	18,151	289	2,769	18,149
6	70	8,597	1,564	17,924	302	2,671	17,922
7	168	9,455	1,307	20,780	150	2,380	20,778
8	354	10,342	1,166	22,780	120	2,274	22,778
9	741	11,126	1,067	24,200	116	2,223	24,198
10	1,604	11,668	986	25,183	89	2,178	25,181

Table 4.2.9 Performance results for islet with equally distributed structure using memory modified ode15s with dense Jacobian.

4.2.4 Memory Modified Ode15s with Sparse Jacobian

Table 4.2.10 contains the performance results for an islet with uncoupled cells using memory modified `ode15s` with sparse Jacobian, as specified in Section 3.4. We observe that most of the performance results are similar for the memory modified `ode15s` with dense and sparse Jacobians. Since the only difference between running with dense Jacobian and sparse Jacobian is how the Jacobian is stored in memory, we would expect the only difference to be in the run time. As we can see from the tables in each subsection, for $N = 10$ the runtime is 3,023 seconds for dense Jacobian and 86 seconds for sparse Jacobian. This is a drastic improvement in runtime, in fact it is the greatest improvement for any of the implementations.

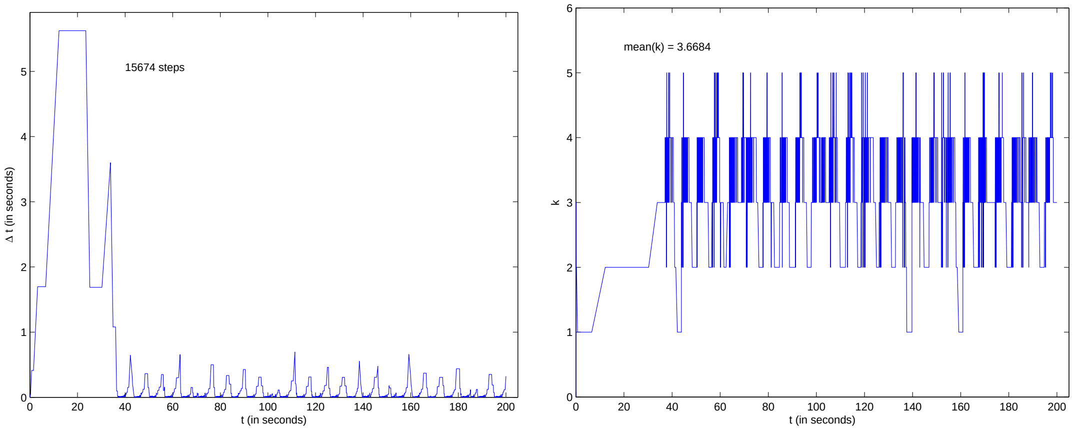
Since the only difference between providing dense Jacobian and sparse Jacobian is storage, we would expect the plots for time step size over time and method order over time to be the same for both implementations. If we compare Figure 4.2.4(a) with Figure 4.2.3(a) and Figure 4.2.4(b) with Figure 4.2.3(b), it appears to be similar, at least at the beginning of the simulation. However, we observe that the number of time steps is less for the dense Jacobian than for the sparse Jacobian and that the mean method order is less for the dense Jacobian than for the sparse Jacobian.

To better compare the two implementations, Figure 4.2.4(a) contains the time step size over time for both implementations. As we can see, the time step size remains the same for a long time, 90 seconds before there are changes between the two implementations. Equivalently, we can see the difference between the method order over time of the two implementations in Figure 4.2.4(b). Just as for the time step size over time, we see that method order over time begins to differ between the two implementations at 90 seconds. This difference in performance results is due to differences in how `ode15s` handles dense matrices and sparse matrices. Particularly in how it computes the LU decomposition of the M^{iter} matrix. The following lines of code appear in `ode15s`:

```
if issparse(Mt)
  [L,U,P,Q,R] = lu(Mt);
else
  [L,U,p] = lu(Mt,'vector');
end
```

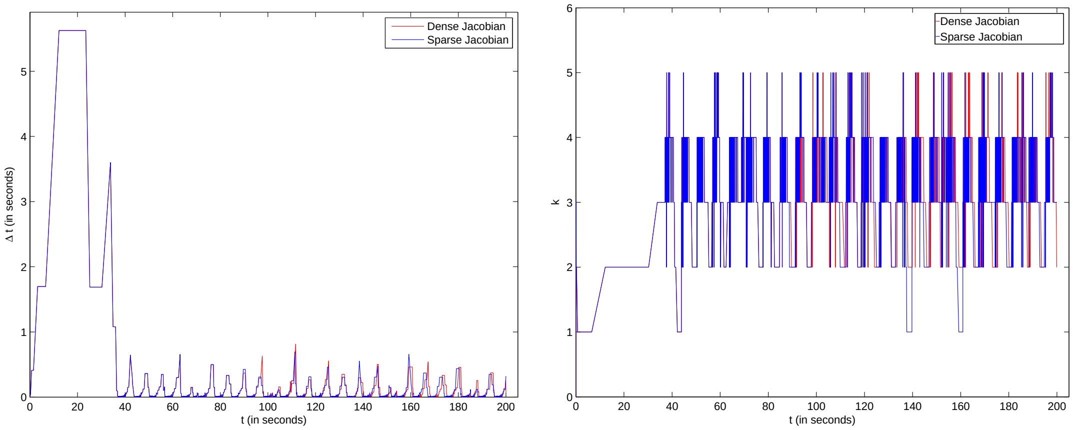
So, if the matrix is dense the `lu` function is invoked with particular arguments for dense matrices and according to Matlab's documentation uses LAPACK to find the LU decomposition. If the matrix is sparse, `lu` is invoked for sparse matrices and according to Matlab's documentation uses UMFPACK to find the LU decomposition. Matlab's documentation also states when the backslash operator is called on a dense matrix LAPACK is used and when it is called on a sparse matrix UMFPACK is used. These differences in the implementation of the code for sparse and dense matrices cause the differences in the performance results.

As in the previous implementations, only the runtimes are provided for the rest of the simulations. Table 4.2.11 contains the runtimes for these simulations. Just as for previous run types, we observe that runtimes increase as N increases for all coupling strengths. We do not observe any relationship between connectivity and runtime.



(a) (b)

Figure 4.2.4 Numerical performance data for uncoupled cells in a 10 × 10 × 10 computational islet using memory modified ode15s with sparse Jacobian: (a) time step Δt vs. t, (b) method order k vs. t.



(a) (b)

Figure 4.2.5 Numerical performance data for uncoupled cells in a 10 × 10 × 10 computational islet using memory modified ode15s with dense and sparse Jacobian: (a) time step Δt vs. t, (b) method order k vs. t.

N	time (seconds)	nsteps	nfailed	nfevals	npds	ndecomps	nsolves
2	14	15,673	2,882	31,264	626	4,925	31,262
3	14	15,673	2,882	31,264	626	4,925	31,262
4	17	15,673	2,882	31,264	626	4,925	31,262
5	20	15,673	2,882	31,264	626	4,925	31,262
6	26	15,673	2,882	31,264	626	4,925	31,262
7	34	15,726	2,889	31,231	637	4,941	31,229
8	46	15,673	2,882	31,264	626	4,925	31,262
9	61	15,726	2,889	31,231	637	4,941	31,229
10	84	15,673	2,882	31,264	626	4,925	31,262

Table 4.2.10 Performance results for uncoupled cells using memory modified ode15s with sparse Jacobian.

(a) Coupling Strength 0.1

N	Grouped	Split Grouped	Layered Middle	Layered Split	Equally Distributed
2	10	9	9	9	5
3	13	11	12	9	4
4	16	16	16	13	7
5	22	22	18	17	9
6	30	28	24	22	14
7	40	37	35	30	18
8	57	50	48	43	28
9	76	69	62	56	30
10	108	97	98	87	59

(b) Coupling Strength 1

N	Grouped	Split Grouped	Layered Middle	Layered Split	Equally Distributed
2	8	8	5	8	8
3	9	9	12	8	8
4	6	11	11	10	10
5	10	9	14	14	13
6	17	10	17	19	17
7	29	18	28	26	24
8	45	24	35	36	35
9	63	51	47	52	50
10	94	76	65	74	70

Table 4.2.11 Runtimes for memory modified ode15s and sparse Jacobian.

(a) Coupling Strength 0.1

N	time (seconds)	nsteps	nfailed	nfevals	npds	ndecomps	nsolves
2	5	4,953	788	11,085	140	1,490	11,083
3	4	4,354	600	9,931	135	1,219	9,929
4	7	5,728	659	12,980	122	1,417	12,978
5	9	6,100	611	13,776	123	1,412	13,774
6	14	6,972	746	16,293	135	1,672	16,291
7	18	7,178	744	16,424	135	1,680	16,422
8	28	7,910	863	18,295	160	1,883	18,293
9	30	5,845	571	13,472	142	1,346	13,470
10	59	8,098	829	18,402	158	1,854	18,400

(b) Coupling Strength 1

N	time (seconds)	nsteps	nfailed	nfevals	npds	ndecomps	nsolves
2	8	8,794	1,649	18,607	293	2,830	18,605
3	8	8,648	1,614	18,060	297	2,759	18,058
4	10	8,600	1,659	18,084	310	2,749	18,082
5	13	8,596	1,665	18,243	303	2,804	18,241
6	17	8,697	1,603	18,114	295	2,714	18,112
7	24	9,254	1,281	20,344	161	2,348	20,342
8	35	10,542	1,165	23,287	106	2,294	23,285
9	50	11,279	1,122	24,517	116	2,274	24,515
10	70	11,716	994	25,243	90	2,201	25,241

Table 4.2.12 Performance results for islet with equally distributed structure using memory modified ode15s with sparse Jacobian.

4.2.5 Ode45

Results in this section are implemented using Matlab's `ode45`. `ode45` uses an explicit Runge-Kutta(4,5) pair of Dormand and Prince [13]. Since it is an explicit method, `ode45` does not require a Jacobian.

Table 4.2.13 contains the performance results for an islet with uncoupled cells using Matlab's `ode45`. We observe that the performance results are better using `ode45` than when using original `ode15s` with numerical Jacobian, memory modified `ode15s` with numerical Jacobian, and memory modified `ode15s` with dense Jacobian. Since this simulation is for an uncoupled islet, the matrix G in (2.4.7) is 0 so this is essentially a non-stiff system.

Table 4.2.14 contains the performance results for an islet with equally distributed structure for coupling strength of 0.1 and 1 using `ode45`. As we can see, increasing the coupling strength of the islet causes the runtime, `nsteps`, `nfailed`, and `nfevals` to increase. Increasing coupling strength increases the magnitude of the matrix G in (2.4.7) causing the system to become more stiff. Since `ode45` is designed to solve non-stiff problems, we would expect the performance results to be worse as the coupling strength increases.

N	time (seconds)	nsteps	nfailed	nfevals
2	12	6,209	1,041	43,501
3	13	6,209	1,041	43,501
4	21	6,209	1,041	43,501
5	33	6,209	1,041	43,501
6	58	6,209	1,041	43,501
7	111	6,209	1,041	43,501
8	159	6,209	1,041	43,501
9	267	6,209	1,041	43,501
10	366	6,209	1,041	43,501

Table 4.2.13 Performance study for uncoupled cells using ode45.

(a) Coupling Strength 0.1				
N	time (seconds)	nsteps	nfailed	nfevals
2	7	4,012	311	25,939
3	9	4,537	260	28,783
4	14	5,132	272	32,425
5	27	5,382	249	33,787
6	52	5,674	210	35,305
7	102	5,805	270	36,451
8	157	5,991	286	37,663
9	221	5,466	215	34,087
10	354	5,964	262	37,357
(b) Coupling Strength 1				
N	time (seconds)	nsteps	nfailed	nfevals
2	34	18,610	1,329	119,635
3	74	27,371	1,419	172,741
4	216	31,065	1,840	197,431
5	580	32,893	1,903	208,777
6	1,356	33,922	1,893	214,891
7	2,846	34,647	1,795	218,653
8	6,965	35,094	2,020	222,685
9	8,105	35,394	2,038	224,593
10	11,249	35,561	2,105	225,997

Table 4.2.14 Performance study for islet with equally distributed structure using ode45.

4.2.6 Ode113

Results in this section are implemented using Matlab's `ode113`. The `ode113` function uses a PECE implementation of Adams-Bashforth-Moulton methods, as specified in Section 3.6. Since it uses functional iterations, `ode113` does not require a Jacobian.

Table 4.2.15 contains the performance results for an islet with uncoupled cells using Matlab's `ode113`. We observe that the performance results are worse for `ode113` than for when using the original `ode15s` with numerical Jacobian, memory modified `ode15s` with numerical Jacobian, memory modified `ode15s` with dense Jacobian, memory modified `ode15s` with sparse Jacobian, and `ode45`. Since this simulation is for an uncoupled islet, the matrix G in (2.4.7) so this is essentially a non-stiff system.

Table 4.2.16 contains the performance results for an islet with equally distributed structure for coupling strength of 0.1 and 1 using `ode113`. As we can see, increasing the coupling strength causes runtime, `nsteps`, `nfailed`, and `nfevals` to increase. The effect is far greater for `ode113` than for `ode15s` and `ode45`. In fact, for $N = 9$ and 10 we run out of memory while running the simulation. That is, `ode113` cannot solve our problem.

N	time (seconds)	nsteps	nfailed	nfevals
2	14	15,637	1,348	32,623
3	44	15,637	1,348	32,623
4	179	15,637	1,348	32,623
5	690	15,637	1,348	32,623
6	1,767	15,637	1,348	32,623
7	5,552	15,637	1,348	32,623
8	8,229	15,637	1,348	32,623
9	11,705	15,637	1,348	32,623
10	16,036	15,637	1,348	32,623

Table 4.2.15 Performance study for uncoupled cells using ode113.

(a) Coupling Strength 0.1

N	time (seconds)	nsteps	nfailed	nfevals
2	7	9,501	696	19,699
3	24	10,287	969	21,544
4	117	11,772	1,047	24,592
5	481	12,589	1,053	26,232
6	1,394	13,412	1,110	27,935
7	4,607	13,736	1,122	28,595
8	7,480	14,350	1,118	29,819
9	7,958	12,547	1,114	26,209
10	14,444	14,126	1,132	29,385

(b) Coupling Strength 1

N	time (seconds)	nsteps	nfailed	nfevals
2	49	40,198	4,689	85,086
3	544	58,856	7,110	124,823
4	3,324	66,287	8,360	140,935
5	14,263	69,829	8,963	148,622
6	40,160	72,659	8,891	154,210
7	128,185	74,446	9,382	158,275
8	Out of Memory	—	—	—
9	Out of Memory	—	—	—
10	Out of Memory	—	—	—

Table 4.2.16 Performance study for islet with equally distributed structure using ode113.

CHAPTER 5

CONCLUSIONS

In this investigation, we used two models for the metabolic and electrical activities of β-cells, a seven variable model and a three variable model. Both of these models consist of a system of coupled ordinary differential equations. We modeled the behavior of an islet of Langerhans by simulating the dynamics of a computational islet represented by a three-dimensional cube of β-cells that are electrically coupled through complexes of proteins called gap junctions.

As a physiological example, we present an investigation of five different distributions of slow and fast bursting cells, called islet structure. We introduce a measure of connectivity based on the fraction of neighboring cells with speed different from the cell's own speed. This quantitative measure of connectivity provides a rational ordering of the different islet structures under consideration. For each islet structure, we run simulations for a range of coupling strengths and find that the burst period intially increases as coupling strength increases before eventually decreasing. We also observe that as connectivity increases the coupling strength at which the burst period is the greatest decreases. We observe this relationship between connectivity and burst period in both the seven variable model and three variable model.

The investigation of islet structures over a range of coupling strengths is an example of a parameter study that requires large numbers of runs of the simulation code. We used Matlab's `ode15s` function to implement each model. Matlab's `ode15s` is used rather than another ODE solver such as `ode45` or `ode113` since our system of coupled ordinary differential equations is a stiff system and `ode15s` is an efficient ODE method for such systems.

In this research, we work to improve the runtime of Matlab's `ode15s`. Since the three variable model has the same dynamics as the seven variable model, we use the three variable model to work to improve runtime.

In order to improve runtime, we first use a modified version of `ode15s` which makes improvements to memory allocation. We then provide `ode15s` with the Jacobian of the ODE system. This Jacobian is first provided as a dense matrix and later as a sparse matrix. For the case of a $10 \times 10 \times 10$ computational islet with no coupling between cells, runtimes go from nearly 4 hours (13,815 seconds) using original `ode15s` with numerical Jacobian to under 1.5 hours (5,046 seconds) using memory modified `ode15s` with numerical Jacobian to under 1 hour (3,023 seconds) using memory modified `ode15s` with dense Jacobian and finally to just over a minute (84 seconds) using memory modified `ode15s` with sparse Jacobian. The runtime for `ode45` is worse than the runtime for our `ode15s` with sparse Jacobian and `ode113` is worse than the runtime of any version of `ode15s`. We also observe that for both `ode113` and `ode45` increasing the coupling strength causes worse performance results.

These improvements in the code for the three variable model can be implemented in the code for the seven variable model as well as other potential models for β-cell behavior. The improvements in runtime will also allow for physiological studies for islets of larger sizes and longer simulation times.

BIBLIOGRAPHY

[1] Kendall E. Atkinson. *An Introduction to Numerical Analysis.* John Wiley & Sons, second edition, 1989.

[2] Richard Bertram and Arthur Sherman. A calcium-based phantom bursting model for pancreatic islets. *Bull. Math. Biol.*, vol. 66, pp. 1313–1344, 2004.

[3] Richard Bertram, Arthur Sherman, and Leslie S. Satin. Metabolic and electrical oscillations: partners in controlling pulsatile insulin secretion. *Am. J. Physiol. Endocrinol. Metab.*, vol. 293, no. 4, pp. E890–900, 2007.

[4] Teresa Ree Chay and Joel Keizer. Minimal model for membrane oscillations in the pancreatic beta-cell. *Biophys. J.*, vol. 42, no. 2, pp. 181–190, 1983.

[5] Sidafa Conde, Teresa Lebair, Christopher Raastad, Virginia Smith, Kyle Stern, David Trott, Matthias K. Gobbert, Bradford E. Peercy, and Arthur Sherman. Enabling physiologically representative simulations of pancreatic beta cells. Technical Report HPCF–2010–21, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2010.

[6] Gerda de Vries, Arthur Sherman, and Hsui-Rong Zhu. Diffusively coupled bursters: Effects of cell heterogeneity. *Bull. Math. Biol.*, vol. 60, pp. 1167–1200, 1998.

[7] J. R. Dormand and P. J. Prince. A family of embedded Runge-Kutta formulae. *J. Comp. Appl. Math.*, vol. 6, pp. 19–26, 1980.

[8] Centers for Disease Control and Prevention. National diabetes fact sheet 2011.

`http://www.cdc.gov/diabetes/pubs/pdf/ndfs_2011.pdf`, 2011. accessed October 2, 2012.

[9] Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, vol. 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, second edition, 1993.

[10] Craig S. Nunemaker and Leslie S. Satin. A tale of two rhythms: a comparative review of the pulsatile endocrine systems regulating insulin and gnrh secretion. *Cellscience*, vol. 2, no. 1, 2005.

[11] P. Rorsman and G. Trube. Calcium and delayed potassium currents in mouse pancreatic beta-cells under voltage-clamp conditions. *J. Physiol*, vol. 374, pp. 531–550, 1986.

[12] Lawrence F. Shampine and M. K. Gordon. *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*. W. H. Freeman, 1975.

[13] Lawrence F. Shampine and Mark W. Reichelt. The MATLAB ODE suite. *SIAM J. Sci. Comput.*, vol. 18, no. 1, pp. 1–22, 1997.

[14] A. Sherman. Contributions of modeling to understanding stimulus-secretion coupling in pancreatic beta-cells. *Am. J. Physiol. Endocrinol. Metab.*, vol. 271, no. 2, pp. E362–372, 1996.

[15] Arthur Sherman and John Rinzel. Rhythmogenic effects of weak electrotonic coupling in neuronal models. *Proceedings of the National Academy of Sciences of the United States of America*, vol. 89, pp. 2471–2474, 1992.

[16] Paul Smolen. A model for glycolytic oscillations based on skeletal muscle phosphofructokinase kinetics. *J. Theor. Biol.*, vol. 174, no. 2, pp. 137–148, 1995.