Mathematical Biosciences 263 (2015) 1–17

Contents lists available at ScienceDirect

Mathematical Biosciences

journal homepage: www.elsevier.com/locate/mbs

Time-stepping techniques to enable the simulation of bursting behavior in a physiologically realistic computational islet

Samuel Khuvis, Matthias K. Gobbert, Bradford E. Peercy*

Department of Mathematics and Statistics, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA

ARTICLE INFO

Article history:
Received 20 May 2014
Revised 3 February 2015
Accepted 5 February 2015
Available online 14 February 2015

Keywords:
Computational islet
Beta cells
Stiff ordinary differential equations
Numerical differentiation formulas
Automatic differentiation
2010 MSC:
37M05
65L04
68W30
92-08
92B25

ABSTRACT

Physiologically realistic simulations of computational islets of beta cells require the long-time solution of several thousands of coupled ordinary differential equations (ODEs), resulting from the combination of several ODEs in each cell and realistic numbers of several hundreds of cells in an islet. For a reliable and accurate solution of complex nonlinear models up to the desired final times on the scale of several bursting periods, an appropriate ODE solver designed for stiff problems is eventually a necessity, since other solvers may not be able to handle the problem or are exceedingly inefficient. But stiff solvers are potentially significantly harder to use, since their algorithms require at least an approximation of the Jacobian matrix. For sophisticated models, systems of several complex ODEs in each cell, it is practically unworkable to differentiate these intricate nonlinear systems analytically and to manually program the resulting Jacobian matrix in computer code. This paper demonstrates that automatic differentiation can be used to obtain code for the Jacobian directly from code for the ODE system, which allows a full accounting for the sophisticated model equations. This technique is also feasible in source-code languages Fortran and C, and the conclusions apply to a wide range of systems of coupled, nonlinear reaction equations. However, when we combine an appropriately supplied Jacobian with slightly modified memory management in the ODE solver, simulations on the realistic scale of one thousand cells in the islet become possible that are several orders of magnitude faster than the original solver in the software Matlab, a language that is particularly user friendly for programming complicated model equations. We use the efficient simulator to analyze electrical bursting and show non-monotonic average burst period between fast and slow cells for increasing coupling strengths. We also find that interestingly, the arrangement of the connected fast and slow heterogeneous cells impacts the peak bursting period monotonically.

1. Introduction

The endocrine system of the pancreas contains clusters of cells called islets of Langerhans, which consist primarily of four different types of cells [15]. The most common type of cell is the β-cell, which is responsible for the secretion of insulin. Since diabetes is characterized by irregular levels of insulin, we are interested in developing a numerical framework that will allow us to calculate and better understand emergent dynamics that lead to the secretion of insulin. Many models of β-cells have been developed that range in focus from electrical bursting and exocytosis to glucose metabolism and have been recently reviewed in Ref. [1]. Furthermore, models of networks of β-cells have been developed [13,17,22]. We use a seven variable β-cell model by Bertram and Sherman [2] and a comparable three variable model by Sherman and Rinzel [8,21].

As an example of a physiological question, we analyze in this paper the effect of electrical coupling between heterogeneous cells in differing patterns of connectivity within a computational islet, represented by a three-dimensional $N \times N \times N$ cube of β-cells. On the physiologically realistic scale of $N^3 = 1000$ cells in an islet, N in $N \times N \times N$ ranges up to 10. By concatenating all unknown variables into a vector y of length $7N^3$ for the seven variable and $3N^3$ for the three variable model, the matrix form of the initial value problems for both islet models with N^3 cells can then be written as a system of coupled ordinary differential equations (ODEs)

$$\frac{dy}{dt} = f^{(\text{ode})}(t,y) = f(t,y) + Gy, \quad 0 < t \le t_f, \quad y(0) = y_0. \tag{1}$$

The two terms on the right-hand side distinguish explicitly the nonlinear reactions in $f(t,y)$ and the coupling between cells involving the matrix G. To capture several burst periods, the simulations for the three variable model are required from 0 ms to 200,000 ms and for the seven variable model the simulations from 0 ms to 500,000 ms.

The reaction equations in these models result in systems of ODEs that are referred to as stiff, since the reaction speeds can vary widely,

* Corresponding author. Tel.: +1 410 455 2436; fax: +1 410 455 1066.
E-mail address: bpeercy@umbc.edu (B.E. Peercy).

http://dx.doi.org/10.1016/j.mbs.2015.02.001
0025-5564/

say between voltage dynamics and gating or endoplasmic reticulum calcium dynamics. ODE solvers appropriate for stiff ODEs necessarily require the Jacobian matrix $J^{(\text{ode})}(t, y) = \nabla_y f^{(\text{ode})}(t, y)$ of the system of ODEs. Since the simulations are needed for large final times to fully capture at least several burst periods, the efficient performance of the ODE solver is crucial. This is the focus of this work.

Some other work that takes into account multiple β-cells to create an islet in three spatial dimensions includes Tsaneva-Atanasova and Sherman [24], who consider electrical coupling and coupling in calcium in a manner similar to our work in a $6 \times 6 \times 6$ islet implemented with Runge–Kutta time-stepping in Fortran 95. Other work by Sherman, Xu, and Stokes [22] considers a $5 \times 5 \times 5$ islet to address the impact of intercellular currents during an experiment where a single islet cell is voltage clamped to record local transmembrane currents. Our works [9] and [16] utilize the islet framework described here with the seven variable model and further include coupling in metabolic variables. Recent work by Pu et al. [17] includes the mention of computer performance. They consider a hexagonal lattice of β-cells with up to 1000 cells using LSODE in Fortran 90. They report that solving a 10 variable model with 1000 cells for 2000 s of model time takes 26 h [17, p. 7]. This information gives an indication of the expected computational load involved in simulations of the intended scale of a seven variable model on $N \times N \times N$ cells.

One of the crucial opportunities for a user to influence the performance of a stiff ODE solver is the choice, how to supply the Jacobian information. Choices range from supplying no information, which results in the code computing a numerical approximation of it, to providing a function that returns the values of the matrix $J^{(\text{ode})}(t, y)$ for inputs (t, y) whenever required by the ODE solver. This function is analogous in interface to the function for $f^{(\text{ode})}(t, y)$ that the user has to provide to the solver in any case to specify the ODE problem (1). But the function for the matrix $J^{(\text{ode})}(t, y)$ is much more difficult to supply, since an analytic formula for each derivative component needs to be calculated and then hand-coded by the user. Since this is a tedious and potentially extremely error prone process, automatic differentiation is a tool that was developed to take the Fortran or C code of a function and differentiate it symbolically. This is different than conventional symbolic differentiation in that both input and output of the process are functions in the same source code as the input function, so it is directly usable as code for a Jacobian function for an ODE solver for which one had to write code for the right-hand side function anyway. It is clear that Matlab offers the same opportunity for automatic differentiation as Fortran or C, but it has taken longer for automatic differentiation libraries to become available. We use the software package ADiMat for automatic differentiation in Matlab [4].

This paper demonstrates the potential advantage of using automatic differentiation in particular in Matlab: Matlab is a language that allows readily the correct implementation of fairly complicated nonlinear model equations, for instance for ODEs, and many users prefer it for this reason over other (particularly source code) languages. By automatic differentiation, the programming of a Jacobian is not a limiting factor any more. In particular, once the machinery of the automatic differentiation is set up, it can readily be used again to obtain a new Jacobian, if the model is changed in any way. This is easier than it would be to hand-calculate the Jacobian again.

To analyze the effect of coupling between cells in a computational islet, we consider an islet of β-cells with varying burst rates [14]. The distribution of these cells is not known, therefore we are investigating several possible distributions of slow bursting and fast bursting cells and capturing their emergent behavior, and we introduce a quantitative measure for the heterogeneity. It turns out that the arrangement of the connected fast and slow heterogeneous cells impacts the peak bursting period monotonically. We also observe that as the heterogeneity of an islet structure increases the peak coupling strength decreases. These simulations demonstrate that both the three variable and the seven variable model have analogous dynamics. This justifies the use of the three variable model as a stand-in for the more complex model in the numerical analyses.

The remainder of this paper is structured as follows: Section 2 describes the physiological background and both the three variable and seven variable models in detail; Section 2.4 specifically defines the model of the coupling strength in (1). Section 3 motivates the inclusion of all available numerical methods, specifies our modifications, and describes the use of ADiMat in more detail. Section 4 contains the full results of our physiological studies. Section 5 presents the numerical performance studies for both models that drive home the need for using an appropriate ODE solver designed for stiff problems and that Matlab can also be an extremely efficient computational tool. For instance, enabling physiologically realistic simulations for the duration of several burst periods on an islet with 1000 cells that takes on the order of 10 min instead of 10 h becomes possible. Finally, Section 6 discusses the detailed conclusions that can be drawn from all reported simulations.

2. Physiological models

In this section, we discuss the physiological background in Section 2.1, we specify the full details of the three variable and seven variable models in Sections 2.2 and 2.3, respectively, and we describe cell coupling in an islet in Section 2.4.

2.1. Physiological background

Diabetes mellitus is a disease 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 either an insulin deficiency or an insulin resistance which means that insulin does not properly interact with cells to signal glucose uptake. Type 1 diabetes corresponds to an insulin deficiency due to an autoimmune attack on insulin-producing β-cells, 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 (Centers for Disease Control and Prevention [5]), being able to model the cells and their interactions which play a large role in diabetes would be valuable.

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 islets contain α-cells, β-cells, δ-cells, and pancreatic polypeptide (PP) producing cells along with distributed capillaries, with β-cells 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.

The consensus model of stimulus-secretion coupling illustrates how a β-cell responds to glucose entering the cell. In the consensus model, after glucose enters the β-cell through the glucose transporter GLUT2 it is converted into pyruvate through glycolysis and then metabolized inside mitochondria. This process produces adenosine triphosphate (ATP) and cellular energy at the expense of adenosine diphosphate (ADP). The increase in ATP–ADP ratio results in the closing of K_{ATP} channels. This results in the depolarization of the β-cell which allows calcium to enter the cell. The calcium triggers autocatalytic release of more calcium from the endoplasmic reticulum and the exocytosis of insulin containing secretory granules. The insulin causes the blood glucose to return back to basal levels by signaling to cells throughout the body. As the glucose levels drop at the β-cell, ATP–ADP levels also tend to recover, allowing the K_{ATP} channels to open back up. The opening of these channels stops the depolarizing 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 of the order of tens of seconds, are categorized as fast β-cells. The cells which burst every few minutes, usually around every 4–6 min, are categorized as slow β-cells [20].

In an islet, cells are connected to each other through a complex of proteins called gap junctions. 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 or even as one entire islet [20]. 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 consider a pair of models which takes into the electrical activity in the cell to describe the bursting of a β-cell. We focus on the electrical output, comparable in both models, rather than other variables such as calcium not explicitly present in both models.

We begin now with a description of the three variable model, before returning to the seven variable model.

2.2. Details of the three variable model

Sherman and Rinzel developed a three variable model to simulate the behavior of two coupled β-cells [21]. This model uses the 3 state 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 a slow variable gating the potassium channel I_s.

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

$$\frac{dV}{dt} = \frac{-(I_K + I_{Ca} + I_s)}{C_m}, \tag{2}$$

$$\frac{dn}{dt} = \frac{\lambda(n_\infty(V) - n)}{\tau}, \tag{3}$$

$$\frac{ds}{dt} = \frac{s_\infty(V) + \beta - s}{\tau_s}. \tag{4}$$

The initial condition is $y_0 = (V_0, n_0, s_0)^T$ with $V_0 = -60$ mV, $n_0 = 0.0001$, $s_0 = 0.4$. This system of ODEs requires the following ionic currents that are used in the two cell model: I_{Ca}, I_K, I_s. These currents are defined as

$$I_K = \hat{g}_K n(V - V_K), \tag{5}$$

$$I_{Ca} = \hat{g}_{Ca} m_\infty(V)(V - V_{Ca}), \tag{6}$$

$$I_s = \hat{g}_s s(V - V_K), \tag{7}$$

where

$$\nu_\infty(V) = \frac{1}{1 + \exp\left[\frac{V_\nu - V}{\theta_\nu}\right]} \quad \text{for } \nu = m, n, s. \tag{8}$$

Table 1 displays the parameters of the system.

To write the system of ODEs (2)–(4) in the standard form $y'(t) = f(t, y)$ for the ODE solver, 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

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

We demonstrate now for one cell that for this model with only three equations and limited complexity, it is workable to insert the definitions for the currents I_K, I_{Ca}, I_s and for $\nu_\infty(V)$, given in detail in Eqs. (5)–(8), into $f(t, y)$ to enable the analytical calculation of the Jacobian. This approach yields concretely first the 3×1 vector

$$f(t, y) = \begin{bmatrix} \frac{-\hat{g}_K}{C_m}(V - V_K)n - \frac{\hat{g}_{Ca}}{C_m}(V - V_{Ca})m_\infty(V) - \frac{\hat{g}_s}{C_m}(V - V_K)s \\ \frac{\lambda}{\tau}(n_\infty(V) - n) \\ \frac{1}{\tau_s}(s_\infty(V) + \beta - s) \end{bmatrix}, \tag{9}$$

with $\nu_\infty(V)$ for $\nu = m, n, s$ in (8) and $y = (V, n, s)^T$. Then, the Jacobian of the ODE system for each cell is computed by analytically differentiating each component of $f(t, y)$ with respect to V, n, s to obtain the 3×3 matrix

$$J(t, y) = \nabla_y f(t, y) = \begin{bmatrix} J_{11} & \frac{-\hat{g}_K}{C_m}(V - V_K) & \frac{\hat{g}_s}{C_m}(V - V_K) \\ \frac{\lambda}{\tau}n'_\infty(V) & -\frac{\lambda}{\tau} & 0 \\ \frac{1}{\tau_s}s'_\infty(V) & 0 & -\frac{1}{\tau_s} \end{bmatrix} \tag{10}$$

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

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

$$= \frac{\exp\left(\frac{V_\nu - V}{\theta_\nu}\right)}{\theta_\nu\left[1 + \exp\left(\frac{V_\nu - V}{\theta_\nu}\right)\right]^2} \tag{11}$$

for $\nu = m, n, s$. One key to making the coding of the above workable without mistake is to program the functions $\nu_\infty(V)$ in (8) and $\nu'_\infty(V)$ in (11) separately and not insert them into (9) and (10). This approach often simplifies the writing of code, but the point is that for a more complicated system, such as the seven-variable model below, this method will not yield sufficient simplifications to allow for coding that is safely free of mistakes. Indeed, already the first step above of inserting the currents to get (9) is too complicated for hand-calculations.

To extend this two cell model to an $N \times N \times N$ computational islet, define now 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 ℓth element is the value of the parameter for cell ℓ using the indexing $\ell = i + N(j-1) + N^2(k-1)$ for $i, j, k = 1, \ldots, N$, and our solution vector y is the $3N^3 \times 1$ 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) for each cell,

Table 1
Constants and variables for the three variable model, where (f) or (s) denote which parameter value goes with a fast or slow cell, respectively.

Symbol	Name	Example or range	Units
t	Time	$0 \le t \le 200,000$	ms
$\hat{g}_{Ca}$	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
C_m	Capacitance	5300	fF
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	35,000	ms
λ	n-gate time constant modifier	0.9	1
β	Heterogeneity constant	0 (f) or 0.075 (s)	1

the second N^3 elements are the values of the right-hand side of (3) for each cell, and the third N^3 elements are the values of the right-hand side of (4) for each cell. Thus, $f(t, y)$ in our initial value problem (1) represents the nonlinear reactions, and the coupling between the cells is modeled explicitly as Gy with the coupling matrix G described in Section 2.4.

The Jacobian of the right-hand side of the ODE system (1) with $3N^3$ equations and a coupling term is then given by the $3N^3 \times 3N^3$ matrix

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

where each $N^3 \times N^3$ block of J is given by

$$J_{11} = \text{diag}\left(-\frac{\hat{g}_K}{C_m}\, n - \frac{\hat{g}_{Ca}}{C_m}\, m_\infty(V) - \frac{\hat{g}_{Ca}}{C_m}\,(V - V_{Ca})\, m'_\infty(V) - \frac{\hat{g}_s}{C_m}\, s\right), \tag{13}$$

$$J_{12} = \text{diag}\left(\frac{-\hat{g}_K}{C_m}(V - V_K)\right), \tag{14}$$

$$J_{13} = \text{diag}\left(\frac{\hat{g}_s}{C_m}(V - V_K)\right), \tag{15}$$

$$J_{21} = \text{diag}\left(\frac{\lambda}{\tau} n'_\infty(V)\right), \tag{16}$$

$$J_{22} = -\frac{\lambda}{\tau} I_{N^3\times N^3}, \tag{17}$$

$$J_{23} = 0_{N^3\times N^3}, \tag{18}$$

$$J_{31} = \frac{1}{\tau_s} s'_\infty(V) I_{N^3\times N^3}, \tag{19}$$

$$J_{32} = 0_{N^3\times N^3}, \tag{20}$$

$$J_{33} = -\frac{1}{\tau_s} I_{N^3\times N^3}. \tag{21}$$

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 diag(v), which constructs a diagonal matrix with the components of the $N^3 \times 1$ vector v along the diagonal.

It is clear that coding this $3N^3 \times 3N^3$ Jacobian of the $N \times N \times N$ cell model is more problematic than the single cell one demonstrated above. But Matlab offers a good opportunity for doing so effectively due to its vectorization and matrix capabilities. For instance, the diagonal matrices in $J^{(\text{ode})}$ can be created in Matlab by the `diag` command; however, this command sets up a dense matrix (that stores all zeros in the matrix), and the analogue command `spdiags` for sparse matrices should actually be used.

2.3. Details of the seven variable model

We simulate the glycolytic oscillator model for β-cells using a deterministic model developed by Smolen [23]. This model was then coupled by Bertram and Sherman [2] with Sherman's update [20] of the model developed by Chay and Keizer [6] and includes voltage data collected by Rorsman and Trube [18].

This model uses 7 state variables in one cell: V, n, [Ca], [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, I_K, that are open in the cell.

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

[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-bisphosphate in the cell.

The dynamics of these state variables are modeled by a system of 7 coupled ordinary differential equations given by

$$\frac{dV}{dt} = \frac{-(I_K + I_{Ca} + I_{K(Ca)} + I_{K(ATP)})}{C_m}, \tag{22}$$

$$\frac{dn}{dt} = \frac{(n_\infty(V) - n)}{\tau_n}, \tag{23}$$

$$\frac{d[\text{Ca}]}{dt} = f_{\text{cyt}}\,(J_{\text{mem}} + J_{\text{er}}), \tag{24}$$

$$\frac{d[\text{Ca}_{\text{er}}]}{dt} = -\sigma_V f_{\text{er}} J_{\text{er}}, \tag{25}$$

$$\frac{d[\text{ADP}]}{dt} = \frac{[\text{ATP}] - [\text{ADP}]\,\exp\left\{(r+\gamma)\left(1 - \frac{[\text{Ca}]}{r_1}\right)\right\}}{\tau_a}, \tag{26}$$

$$\frac{d[\text{G6P}]}{dt} = \kappa\,(R_{\text{GK}} - R_{\text{PFK}}), \tag{27}$$

$$\frac{d[\text{FBP}]}{dt} = \kappa\,(R_{\text{PFK}} - 0.5\,R_{\text{GPDH}}). \tag{28}$$

The initial condition is $y_0 = (V_0, n_0, [\text{Ca}]_0, [\text{Ca}_{\text{er}}]_0, [\text{ADP}]_0, [\text{G6P}]_0, [\text{FBP}]_0)^T$ with $V_0 = -60$ mV, $n_0 = 0.00016$, $[\text{Ca}]_0 = 0.08$ nM, $[\text{Ca}_{\text{er}}]_0 = 789$ nM, $[\text{ADP}]_0 = 170$ nM, $[\text{G6P}]_0 = 187$ nM, $[\text{FBP}]_0 = 16$ nM. These equations require the following ionic currents that are used in the individual β-cell model: $I_K, I_{Ca}, I_{K(Ca)}, I_{K(ATP)}$, where $I_{XX(YY)}$ is the current of ion XX and modulated by YY. These currents are defined by

$$\begin{aligned}
I_K &= \bar{g}_K\, n\,(V - V_K),\\
I_{Ca} &= \bar{g}_{Ca}\, m_\infty(V)\,(V - V_{Ca}),\\
I_{K(Ca)} &= g_{K\,Ca}\,(V - V_K),\\
g_{K\,Ca}([\text{Ca}]) &= \frac{\bar{g}_{K\,Ca}([\text{Ca}])}{1 + \left(\frac{K_d}{[\text{Ca}]}\right)^2},\\
I_{K(ATP)} &= g_{K\,ATP}\,(V - V_K),\\
g_{K\,ATP} &= \bar{g}_{K,ATP}\, o_\infty([\text{ADP}]),\\
m_\infty(V) &= \frac{1}{1 + \exp\left[\frac{-(20+V)}{12}\right]},\\
n_\infty(V) &= \frac{1}{1 + \exp\left[\frac{-(16+V)}{5}\right]},\\
o_\infty([\text{ADP}]) &= \frac{((8.4\times 10^{-5})\,[\text{ADP}]^2 + 0.0016\,[\text{ADP}] + 0.08)\bar{g}_{K,ATP}}{(0.05\,[\text{ATP}] + 0.0052\,[\text{ADP}] + 1)(1 + 0.0098\,[\text{ADP}])^2},\\
[\text{ATP}] &= 1500 - \frac{1}{2}[\text{ADP}]\\
&\quad + \frac{1}{2}(-3[\text{ADP}]^2 - 6000[\text{ADP}] + 9,000,000)^{\frac{1}{2}},\\
J_{\text{er}} &= 0.0002([\text{Ca}_{\text{er}}] - [\text{Ca}]) - 0.4[\text{Ca}],\\
J_{\text{mem}} &= (4.5\times 10^{-6}) I_{Ca} - 0.2[\text{Ca}],
\end{aligned}$$

where R_{GK}, $\bar{g}_{K\,Ca}$, and $\bar{g}_{K,ATP}$ are parameters relating to the bursting rate of the cell. These are the parameters altered for fast or slow cells. The parameters are chosen for robust burst period separation between fast and slow bursters. Modest changes that retain the slow fast separation will have only modest quantitative but not qualitative effects on the output and will not substantially affect the computational timing beyond that already captured in the strength of the coupling conductance. Table 2 displays the parameters of the system.

Table 2
Constants and variables for the seven variable model, where (f) or (s) denote which parameter value goes with a fast or slow cell, respectively.

Symbol	Name	Example or range	Units
t	Time	$0 \le t \le 500,000$	ms
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
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	0.4 (f) or 0.2 (s)	nM/ms
$\bar{g}_{KCa}$	Maximum K Ca channel conductance	25,000 (f) or 27,000 (s)	pS
$\bar{g}_{K,ATP}$	Maximum K ATP channel conductance	600 (f) or 100 (s)	pS

R_{PFK} is the rate of phosphofructokinase activity. It is given as a combinatorial function of [G6P] and [FBP] that can be found in [23]. Each cell in our cluster has a certain state based on the 7 state variables. We call this set of variables S.

To model the state variables in an islet with $N \times N \times N$ cells, we introduce now V, n, [Ca], [Ca_{er}], [ADP], [G6P], and [FBP] as vectors of length N^3 corresponding to the value of each variable for each of the cells in the islet. For each vector the ℓth element is the value of the parameter for cell ℓ, where $\ell = i + N(j-1) + N^2(k-1)$ for $i, j, k = 1, \ldots, N$, and our solution vector, y, is the $7N^3 \times 1$ vector

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

The vector relating $f(t, y)$, the right-hand side of (22)–(28), is similarly defined as

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

In matrix form, our initial value problem is again the ODE system (1), involving the nonlinear reactions $f(t, y)$ and the coupling between the cells is modeled as Gy with the coupling matrix G described in Section 2.4. Unlike the three variable model, the seven variable model is too complex to compute the analytical Jacobian by hand.

2.4. Cell coupling

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 coupling in our models, which means that only cells directly adjacent to each other have a gap junction between them. In our model, only voltage is coupled between neighboring cells.

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

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

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

Our simulation uses a $N^3 \times N^3$ matrix C, where the (a, b) entry of C is the coupling strength between cell a and cell b. Here, cell ℓ is defined by indexing $\ell = i + N(j-1) + N^2(k-1)$ for $i, j, k = 1, \ldots, N$. We define a matrix C to represent the voltage coupling between neighboring cells in the cubic lattice. This can be thought of as the adjacency matrix for nearest neighbors multiplied by the coupling strength. 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 strengths of the islet. This C matrix is used to define G in the ODE system (1) as

$$G = \begin{pmatrix} C & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}. \tag{29}$$

The coupling matrix 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. Each block, including each 0 block, is of size $N^3 \times N^3$ for the $N \times N \times N$ islet. Only the upper left block for the voltage variable is non-zero in G and contains the C matrix; among the variables, voltage coupling is the only coupling treated here.

The distribution of the cells with heterogeneous bursting properties is not known, therefore we are investigating several possible distributions of slow bursting and fast bursting cells and capturing their emergent behavior. Each islet distribution has a connection heterogeneity 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. We introduce a measure of connection heterogeneity based on the fraction of neighboring cells with burst frequency different from the cell's own burst frequency. This quantitative measure of connection heterogeneity provides a rational ordering of the different islet structures. The connection heterogeneity of a particular islet distribution is computed as

$$c_h = \frac{1}{N^3} \sum_{i=1}^{N^3} \frac{d_i}{v_i}, \tag{30}$$

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 consider five distributions of slow and fast cells. These islet distributions are depicted in Fig. 1. Black and white circles represent slow and fast cells and the lines between cells represent gap junctions. The five distributions we consider are:

a. 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 connection heterogeneity of 0.0928.
b. 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 connection heterogeneity of 0.1712.

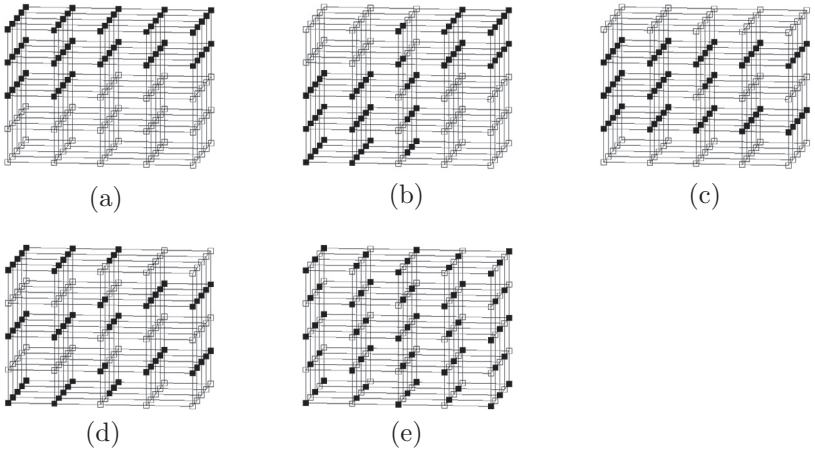


Fig. 1. Schematic of distributions of slow and fast cells for a $5 \times 5 \times 5$ islet: (a) grouped structure, (b) split grouped structure, (c) layered middle structure, (d) layered split structure, and (e) equally distributed structure.

c. 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 connection heterogeneity of 0.2693.
d. The *layered split structure* has N alternating layers where a layer with the first $\frac{N^2}{2}$ cells are slow and 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. It has a connection heterogeneity of 0.4261.
e. 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 connection heterogeneity of 1.0000.

3. Numerical methods

This section introduces the numerical methods used to solve both the seven variable and the three variable models described in Section 2.

We compare in our studies five approaches to supplying the Jacobian and two implementations of the NDF*k* family of methods, for a total of ten cases using the Matlab function `ode15s`. To provide a clear contrast of the performance of a stiff ODE solver to non-stiff solvers, we also include two other ODE solvers in our comparison, the non-stiff solvers in functions `ode45` and `ode113`.

3.1. Matlab's stiff solver ode15s

Systems of ODEs of the types (2)–(4) and (22)–(28) need to be handled as stiff equations, since the reactions modeled by the ODEs can have a wide range of speeds. The numerical differentiation formulas (NDF*k*), which improve upon the well-known backward differentiation formulas (BDF*k*), are a state-of-the-art ODE solver family for stiff problems. An introduction to these methods is available in Ref. [19] and a detailed description introduction in our context in Ref. [11]. There are two key ingredients to stiff ODE solvers: To achieve the stability required for solving stiff ODEs, the NDF*k* methods discretize the time derivative backwards in time and solve the arising nonlinear system by Newton's method. This requires the Jacobian matrix, that is, the gradient of the right-hand side of the ODE system (1), which involves the Jacobian of the systems (2)–(4) and (22)–(28).

One implementation of the NDF*k* methods is Matlab's `ode15s` function. `ode15s` selects both the step size Δt and method order k automatically to control the estimated ODE error.

The other implementation is a version of `ode15s` with modified memory handling. This version makes two improvements on the management of memory in `ode15s`. The first improvement is to not store the 3D array `dif3d` which contains all gradient approximations for all solution components at all time steps. The second improvement is to improve the allocation and reallocation of memory for vectors stored by the function by using larger chunk sizes. These modifications reduce the memory needed for the simulation significantly and enable the solution of larger problems. They also have 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. These issues were discovered and improvements were tested originally in Ref. [7] and studied more systematically in Ref. [11]. By deleting the array `dif3d`, we give up on some powerful post-processing functions available in Matlab; but most users do not take advantage of these anyhow.

The five approaches to supplying the Jacobian matrix are:

Numerical Jacobian: Since the NDF*k* uses Newton's method, it requires the Jacobian of the system of ODEs. If no Jacobian is provided by the user, the Jacobian is numerically approximated using divided differences for each element in the matrix, which is an expensive computation.

Dense Jacobian: Matlab's `ode15s` provides the option 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. Evaluating an analytic Jacobian is much cheaper computationally than computing its approximation numerically and also increases its accuracy.
As we can see from (12) to (21), the Jacobian is a block matrix, and each block is mathematically diagonal. We can use Matlab's `diag` function to create these diagonal matrices. However, this creates these matrices in dense storage mode, in which all data of the matrix – including elements with zero value – are stored.

Sparse Jacobian: We supply the `ode15s` function with the Jacobian in sparse storage mode in this implementation. This recognizes the mathematical fact that most matrix elements of diagonal matrices are zero. Since Matlab stores a sparse matrix more efficiently than a dense matrix by not storing the zero values, we can expect a speedup in the runtime of simulations. So, rather than using Matlab's `diag` command we use the `spdiags` command to create the sparse diagonal matrices necessary to form the Jacobian in this approach. This will show that correct usage of its features is vital for efficient calculations in Matlab.

Sparsity pattern of Jacobian: An alternative to supplying the Jacobian of the system of ODEs to `ode15s` is to provide the sparsity pattern of the Jacobian matrix, J. Matlab's `ode15s` allows for the option to provide the sparsity pattern of the Jacobian by a matrix with ones at each position where the Jacobian contains a non-zero element and zeros everywhere else. The Jacobian is then computed numerically as in the first method, however

it is stored in sparse storage mode and only the elements of the Jacobian which correspond to a non-zero element of the matrix S are computed.

Automatically differentiated (AD) Jacobian: We are able to use a software tool to generate a function for the analytic Jacobian for functions that are too complex to compute analytically. The ADiMat software tool [4] used in this paper uses the function for the ODE to generate code that computes the Jacobian of the right-hand side of the system. We ensure that the calculated Jacobian uses sparse storage mode by providing the sparsity pattern to ADiMat.

After installing this software according to its instructions, we create a file `jac.m` which computes the Jacobian of the system of ODEs. In this file, we define an `admOptions` struct which contains all the information necessary for the software to compute the Jacobian. We define the following fields: `Jpattern`, `independents`, and `dependents`. `Jpattern` contains the sparsity pattern of the Jacobian, with ones at each position where the Jacobian contains a non-zero element and zeros everywhere else. We then call the function `admDiffFor` to compute the Jacobian with the following code:

```
Jac = admDiffFor(@model, @cpr, t, y,p,N^3,opts);
```

The `Jac` matrix returned by this function is returned by the `jac` function.

The first argument, `@model`, is a function handle for the function containing the model. The second argument, `@cpr`, is function handle to a function used by the software to generate a sparse Jacobian. The third argument, `t` is the current time. The fourth argument, `y`, is the solution vector at that time. The fifth argument, `p`, is the distribution of slow and fast cells. The sixth argument, `N^3`, is the number of cells in the islet. The seventh argument, `opts`, is the `admOptions` struct.

This code creates a file, called `g_model.m`. This function should not be used directly, since sparsity is not preserved in this function. Rather, use the code above to generate the Jacobian.

By providing this `jac` function to `ode15s` we provide a sparse Jacobian to the ode solver.

One key advantage of this approach is that once the machinery of ADiMat is set up, it is available to compute a new Jacobian function whenever a change to the model is made.

3.2. Two non-stiff ODE solvers in Matlab

Matlab's `ode45` solver uses the explicit Runge–Kutta–Fehlberg 5(4) pair of Dormand and Prince [19]. In `ode45`, the step size Δt is selected automatically to control the estimated ODE error. The method is explicit and thus does not require a Jacobian. It is not suitable for stiff problems.

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 1–12, see Ref. [19]. While the Adams–Moulton method used as corrector is implicit, due to the use of functional iteration, no Jacobian is required for the algorithm. `ode113` selects both the step size Δt and method order k automatically to control the estimated ODE error. However, for the functional iterations to be stable, the time step needs to be restricted to be relatively small. This makes this family of methods unsuitable for stiff problems.

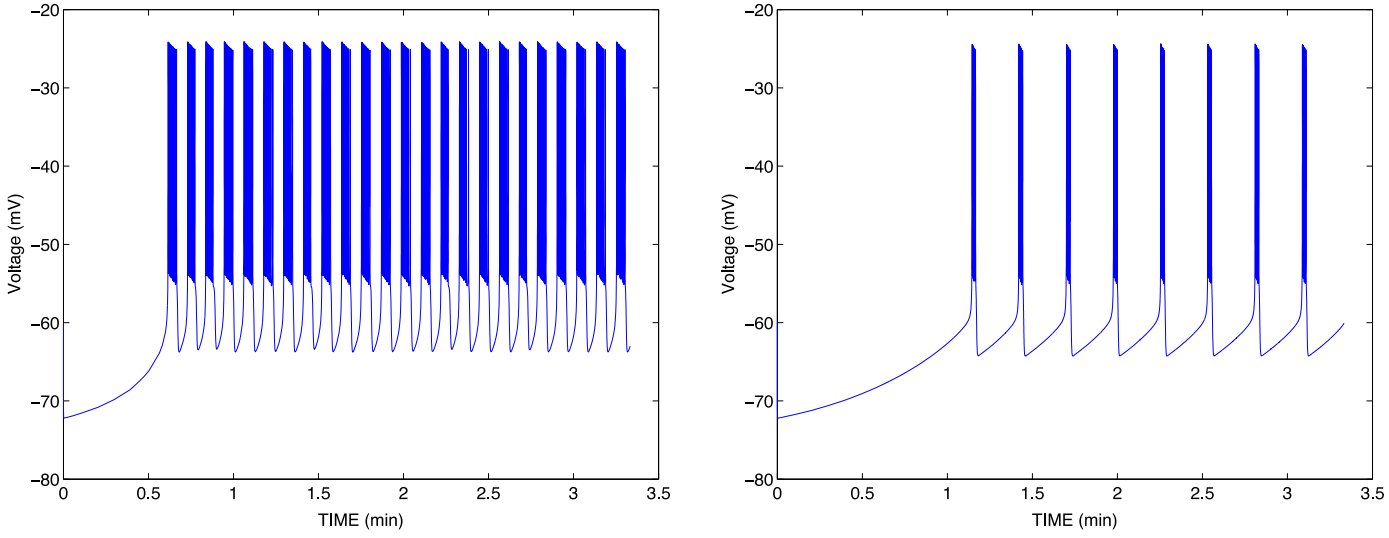
4. Physiological results

This section collects representative physiological results of simulations for the three variable model in Figs. 2–4 and Table 3 and for the seven variable model in Figs. 5–7 and Table 4. We present the same sets of results for both models to confirm their analogous dynamics, which provides the justification for using the three variable model as a test bed for numerical methods intended for the seven variable model and other models with similar dynamics.

All physiological studies use a $5 \times 5 \times 5$ computational islet with 50% slow cells and 50% fast cells. Results use the most efficient numerical method available for each model using a relative tolerance of 10^{-3} and an absolute tolerance of 10^{-5} on the ODE error.

The simulations for the three variable model are run from 0 ms to 200,000 ms and for the seven variable model the simulations from 0 ms to 500,000 ms. The coupling strength between cells in the islet is expressed in pS for the seven variable model, however the three variable model uses a unitless coupling strength as used in Ref. [8] that can be multiplied by $\frac{C_m}{\tau} = 265$ pS to obtain the coupling strength in pS to compare with the seven variable model, with C_m and τ from Table 1.

We study the effects of distributions on the average burst period of cells in the islet for the five different islet structures defined in Section 2.4: grouped structure with connection heterogeneity $c_h = 0.0928$, split grouped structure with $c_h = 0.1712$, layered middle



(a) fast cell (b) slow cell

Fig. 2. Bursting behavior of uncoupled cells for the three variable model.

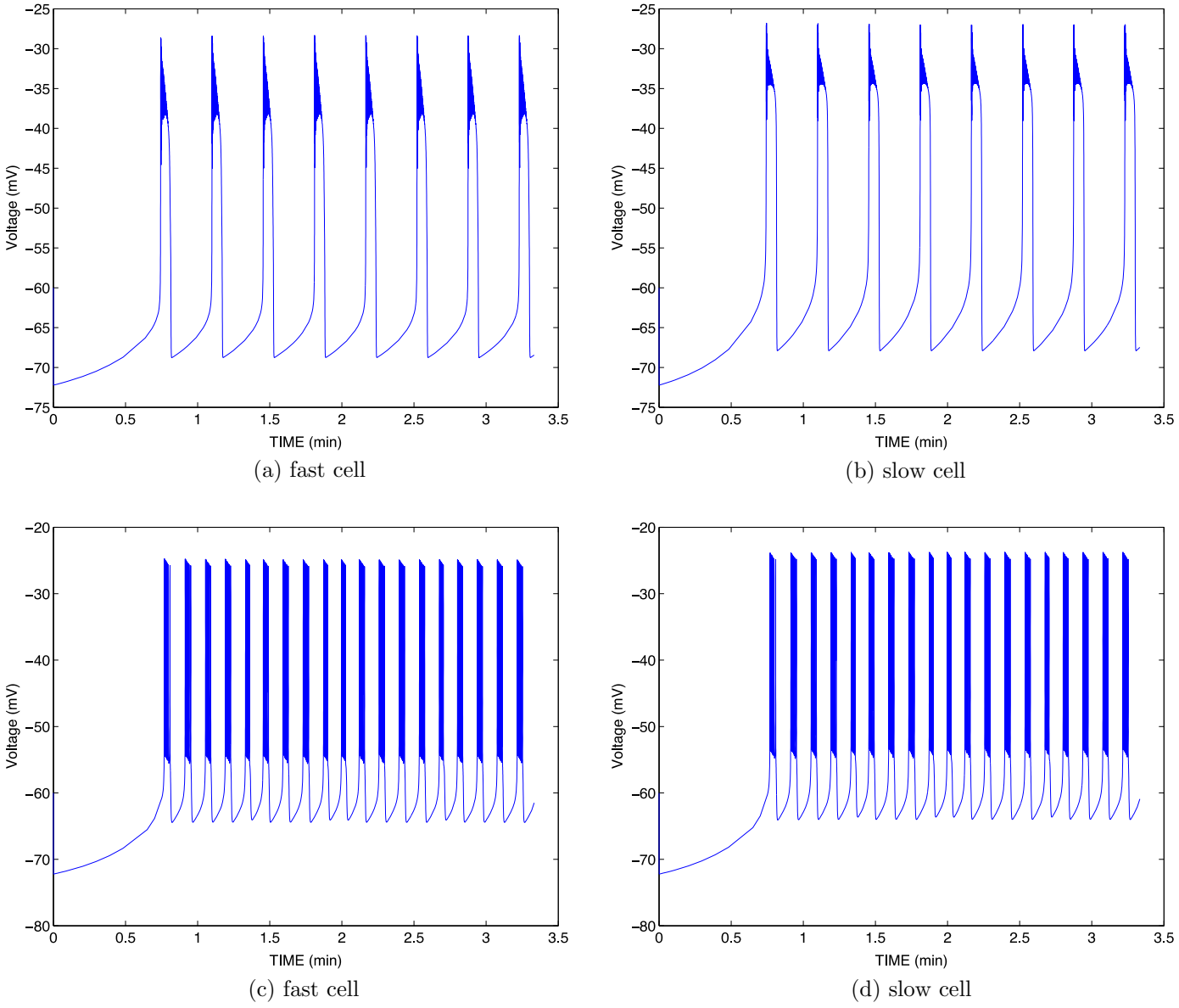


Fig. 3. Bursting behavior of cells in an islet with equally distributed structure ($c_h = 1$) for the three variable model. (a) and (b) Islet with coupling strength of 0.1. (c) and (d) Islet with coupling strength of 1.

structure with $c_h = 0.2693$, layered split structure with $c_h = 0.4261$, and the equally distributed structure with $c_h = 1.0000$.

Figs. 2 and 3, and 5 and 6 contain the bursting behavior of cells in an islet. The bursting behavior of a cell is shown by plotting the voltage of a cell in mV 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 very small coupling strength.

As we can see in Figs. 2 and 5, 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 behaviors for uncoupled cells are the same for different islet structures. As we can see, this is true for both the three variable model and seven variable model, though the burst periods and amplitude are different for each model.

However, as we can see by comparing plots (a) and (b) in each of Figs. 3 and 6, the burst period is already synchronized when there is a coupling strength of 0.1 between cells for the three variable model and a coupling strength of 100 pS for the seven variable model. We observe that although the burst period is the same at such a coupling strength, there is still a difference in the behavior of the bursting. For a stronger coupling strength of 1 for the three variable model and 1000 pS for the seven variable model, these differences between slow and fast cells are far less visible. We observe this same behavior for each of the five structures.

The five plots (a) through (e) in Figs. 4 and 7 contain plots of average burst period in minutes versus coupling strength for a representative fast cell and slow cell 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 three variable model and 1000 ms for the seven variable model 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 mV. 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.

As we would expect, the burst period is the same for the uncoupled cells. Insets have been added to Fig. 4(a) and (b) to clarify this point since the burst periods of slow and fast cells increase when there is a very small coupling strength for the grouped and split grouped islet structures. In all other cases we observe that the burst period and

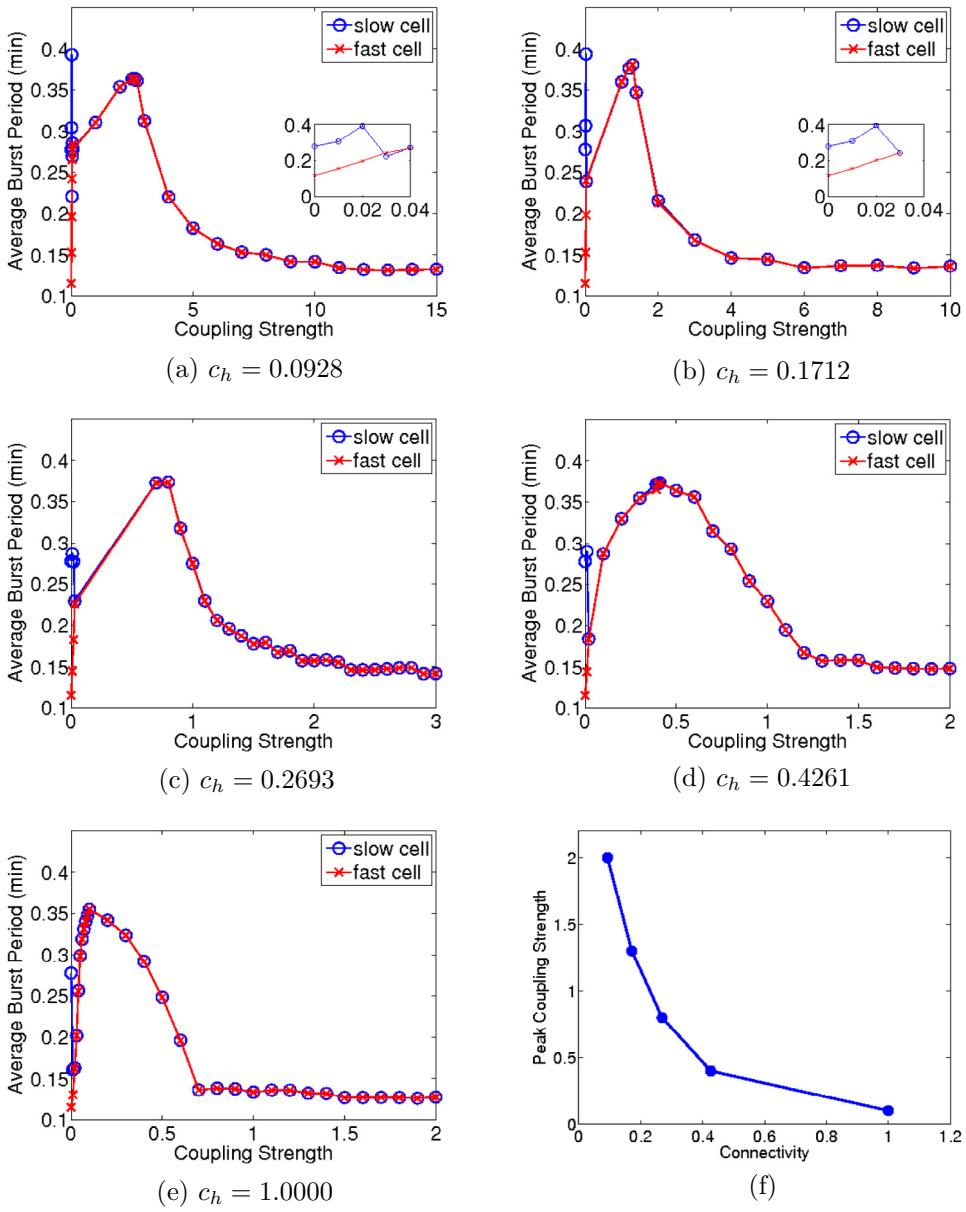


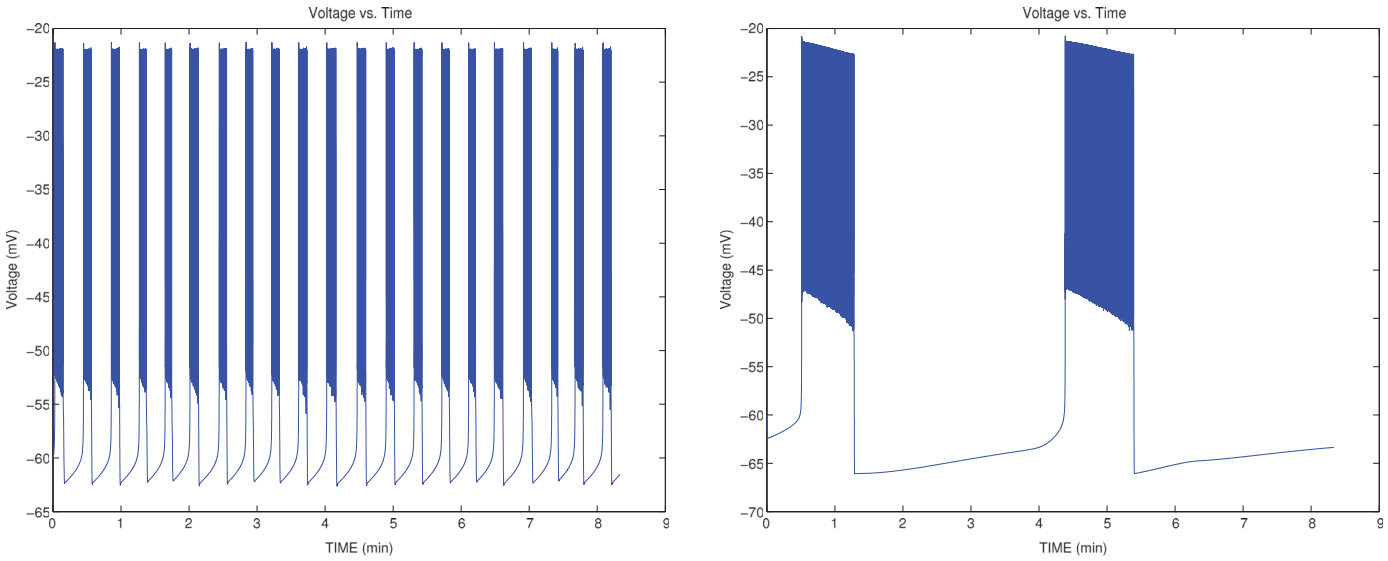
Fig. 4. Average burst period over a range of coupling strengths for cells for the three variable model: (a) grouped structure, (b) split grouped structure, (c) layered middle structure, (d) layered split structure, (e) equally distributed structure. (f) Peak coupling strength vs. connection heterogeneity.

Table 3
Synchronization, peak, and leveled coupling strengths for each islet structure for the three variable model.

Coupling strength	Grouped ($c_h = 0.0928$)	Split grouped ($c_h = 0.1712$)	Layered middle ($c_h = 0.2693$)	Layered split ($c_h = 0.4261$)	Equal ($c_h = 1.0000$)
Synchronization	0.06	0.03	0.03	0.02	0.02
Peak	2.60	1.30	0.80	0.40	0.10
Leveled	15.00	10.00	3.00	2.00	1.00

bursting behavior for the same coupling strength varies for different islet structures. As we previously noted, the burst period synchronizes at a very small coupling strength for all five islet structures. We observe that as the connection heterogeneity of the structure increases the synchronization coupling strength decreases. 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. Plots (f) in Figs. 4 and 7 contain plots of peak coupling strength versus connection heterogeneity of the islet for each model. 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. To compare the values of coupling strength quantitatively, the dimensionless coupling strength in Fig. 4(f) can be multiplied by 265 pS, as explained above. These plots demonstrate that as connection heterogeneity increases, the peak coupling strength decreases. This demonstrates the appropriateness of the definition for connection heterogeneity of an islet structure in Section 2.4.



(a) fast cell (b) slow cell

Fig. 5. Bursting behavior of uncoupled cells for the seven variable model.

Tables 3 and 4 collect the coupling strengths for each islet structure at which (i) the slow and fast cells synchronize, (ii) the peak burst period is achieved, and (iii) the burst period stops changing as coupling strength increases. These coupling strengths are determined using the five plots (a) through (e) of average burst period vs. coupling strength for each islet structure in Figs. 4 and 7. We define synchronization as when the burst periods of the slow and fast cells are within 1.25 s of each other. The peak coupling strength is given by the highest point after synchronization, and leveled refers to the coupling strength, after which the lines are essentially flat; we considered more data than are shown in the plots, hence the extent of the horizontal axes in the five plots (a) through (e) in Figs. 4 and 7 were actually set after determining the leveled coupling strength. Since these tables contain coupling strengths of particular physiological interest, we test the numerics in Section 5 for each model for these coupling strengths.

5. Numerical performance studies

This section contains the results of our numerical performance studies, first for the three variable model in Tables 5 and 6 and Fig. 8, and then for the seven variable model in Tables 7–9. Results were obtained using `ode15s`, `ode45`, and `ode113` with a relative tolerance of 10^{-3} and an absolute tolerance of 10^{-5} on the respective estimator of the ODE error. We begin by reviewing the results for the three variable model since we are able to analytically compute the Jacobian, allowing us to compare the performance results for this numerical method to those for the automatically differentiated Jacobian. We then demonstrate the results for the seven variable model, where the computation of the analytical Jacobian is too complex to write by hand.

Simulations were run on one node of the 86-node cluster tara in the UMBC High Performance Computing Facility (http://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. While Matlab is nominally serial, it does use multithreading on this kind of node with 8 computational cores available. And naturally, an advantage of a large cluster is that many runs of Matlab can take place simultaneously, which is a powerful enabler for studies with many cases, such as reported in the previous section.

5.1. Three variable model

Table 5 contains the runtimes for the coupling strengths from Table 3 for all five islet structures using the memory modified `ode15s` with sparse Jacobian for the three variable model. We consider these different coupling strengths to test the ODE solver across a range of conditions. For all five islet structures we observe that the peak coupling strength has the shortest runtime, followed by the leveled coupling strength, with the synchronization coupling strength having the longest runtime. We also observe that the runtime is approximately the same for analogous simulations for each of the five islet structures. Based on this observation, we only show results for the equally distributed islet structure in the following.

Matlab's `ode15s` returns useful statistics about its performance. One statistic of particular interest is the time stepping behavior of the ODE solver. We find that for analogous simulations the synchronization coupling strength took the most steps, followed by the leveled coupling strength, and then the peak coupling strength. As an example, Fig. 8(a) contains the time step size vs. time and Fig. 8(b) contains the method order vs. time for the equally distributed islet structure using the memory modified `ode15s` with sparse Jacobian. We observe that time step size decreases and method order increases as the cell bursts. This is exactly the desired behavior for an ODE solver with sophisticated time step and method order control based on an error estimator for the ODE error.

Table 6 contains the runtimes for the equally distributed islet structure with the synchronization coupling strength, peak coupling strength, and leveled coupling strength for all available numerical methods. We observe that in most cases the runtime is longer for both of the non-stiff solvers `ode45` and `ode113`, when compared to the runtimes of the more efficient `ode15s` methods, particularly for larger coupling strengths. The function `ode45` is competitive at peak coupling strength, which is the least stiff case with shorter runtimes than the other coupling strengths. But overall, these timing results demonstrate the need to use a stiff solver when running simulations for a range of coupling strengths.

We observe that simply providing a dense Jacobian results in a small improvement in runtime when compared to the numerical Jacobian. In the instance of $N = 10$ for the peak coupling strength, for instance, we observe an improvement in runtime from 4140 s to 3588 s. Even if we provide a sparse Jacobian, we only observe a

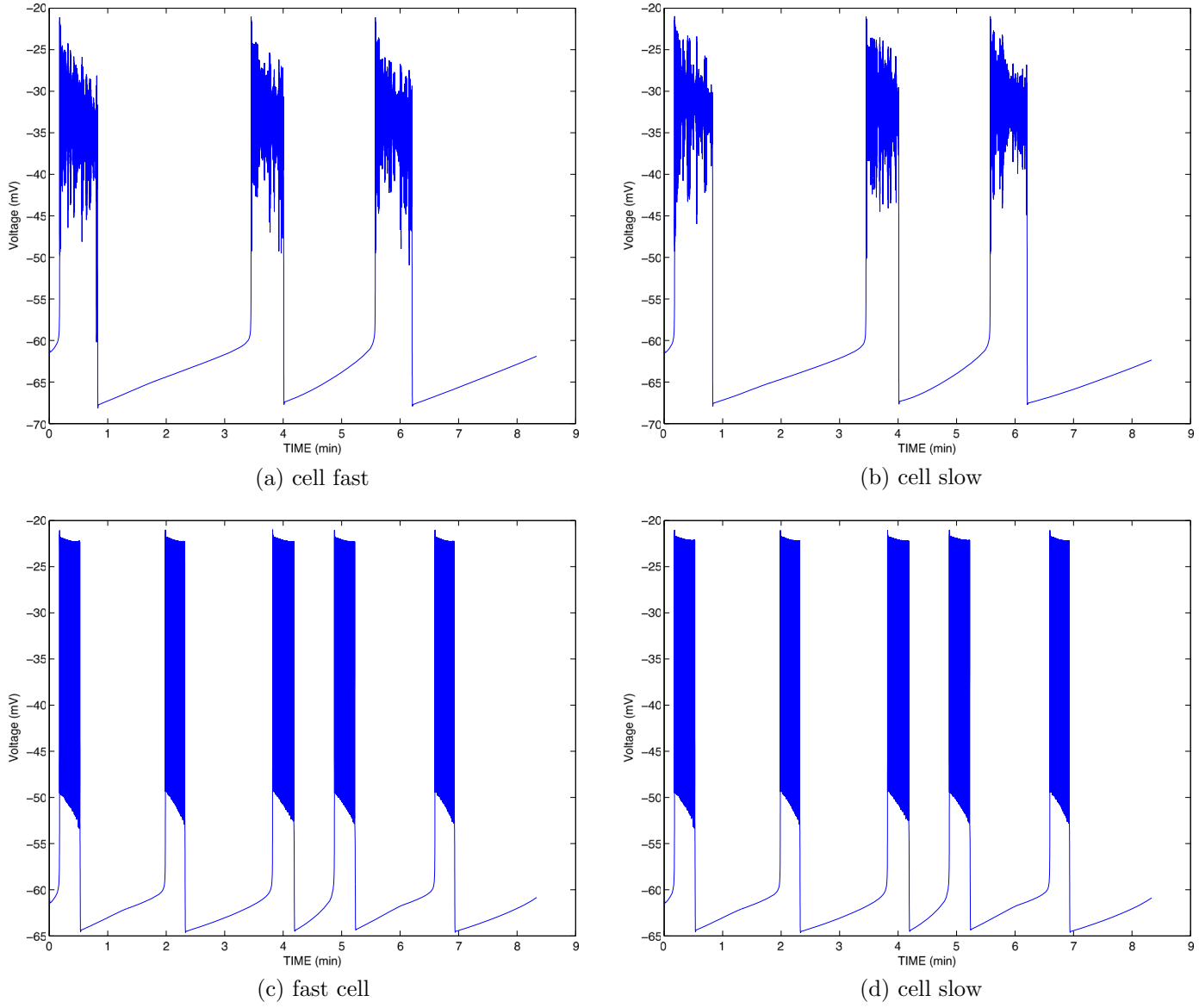


Fig. 6. Bursting behavior of cells in an islet with equally distributed structure ($c_h = 1$) for the seven variable model. (a) and (b) Islet with coupling strength of 100 pS. (c) and (d) Islet with coupling strength of 1000 pS.

small improvement in runtime to 2268 s. If we provide the sparsity pattern of the Jacobian, we observe a runtime of 2245 s, similar to when the sparse Jacobian is provided. We also observe that the original `ode15s` with automatically differentiated Jacobian has a runtime of 2296 s, which is on the same order of magnitude improvement as the original `ode15s` with sparse Jacobian. We can conclude that for the original `ode15s`, having a sparse Jacobian matrix is the key, which is accomplished by any of the latter three methods.

By using the memory modified `ode15s` with numerical Jacobian, we observe a improvement in runtime from the original 4140 s to 1813 s, which is a larger improvement than any of the methods described so far. We observe a small improvement in runtime by providing a dense Jacobian, with a runtime of 1276 s. By providing a sparse Jacobian, we observe a much more dramatic improvement in runtime to just 57 s. This is an improvement in runtime on the order of 100 times faster than the original `ode15s` with numerical Jacobian. We observe that providing the sparsity pattern of the Jacobian results in a runtime of 53 s. This is slightly faster than using the sparse Jacobian. The memory modified `ode15s` with automatically differentiated Jacobian has a runtime of 89 s, which is still on the same order of magnitude improvement as the memory modified `ode15s` with sparse Jacobian or sparsity pattern. We can conclude overall that using the memory modified `ode15s` with any of the three methods that use a sparse Jacobian provided the dramatic improvement of runtime.

We observe that `ode113` runs out of memory for larger N for some of the coupling strengths. A memory study confirms that as we increase N, the memory usage increases until it runs out of memory for the worst cases of large N using `ode113` for synchronization and leveled coupling strength. A study of the memory required to run our simulations with `ode15s` implementations found that the most memory for these simulations was 16.1 GB for the least efficient method of numerical Jacobian with original `ode15s`. Since `ode15s` stores a large amount of information for each step, the number of steps directly affects the memory usage. We found that providing an analytical Jacobian results in a small improvement in memory usage. The largest improvement in runtime is obtained by using the memory modified `ode15s` with sparse Jacobian. We observe that the most memory efficient implementations are obtained by providing the sparsity pattern of the Jacobian or by providing the AD Jacobian function to the memory modified `ode15s`.

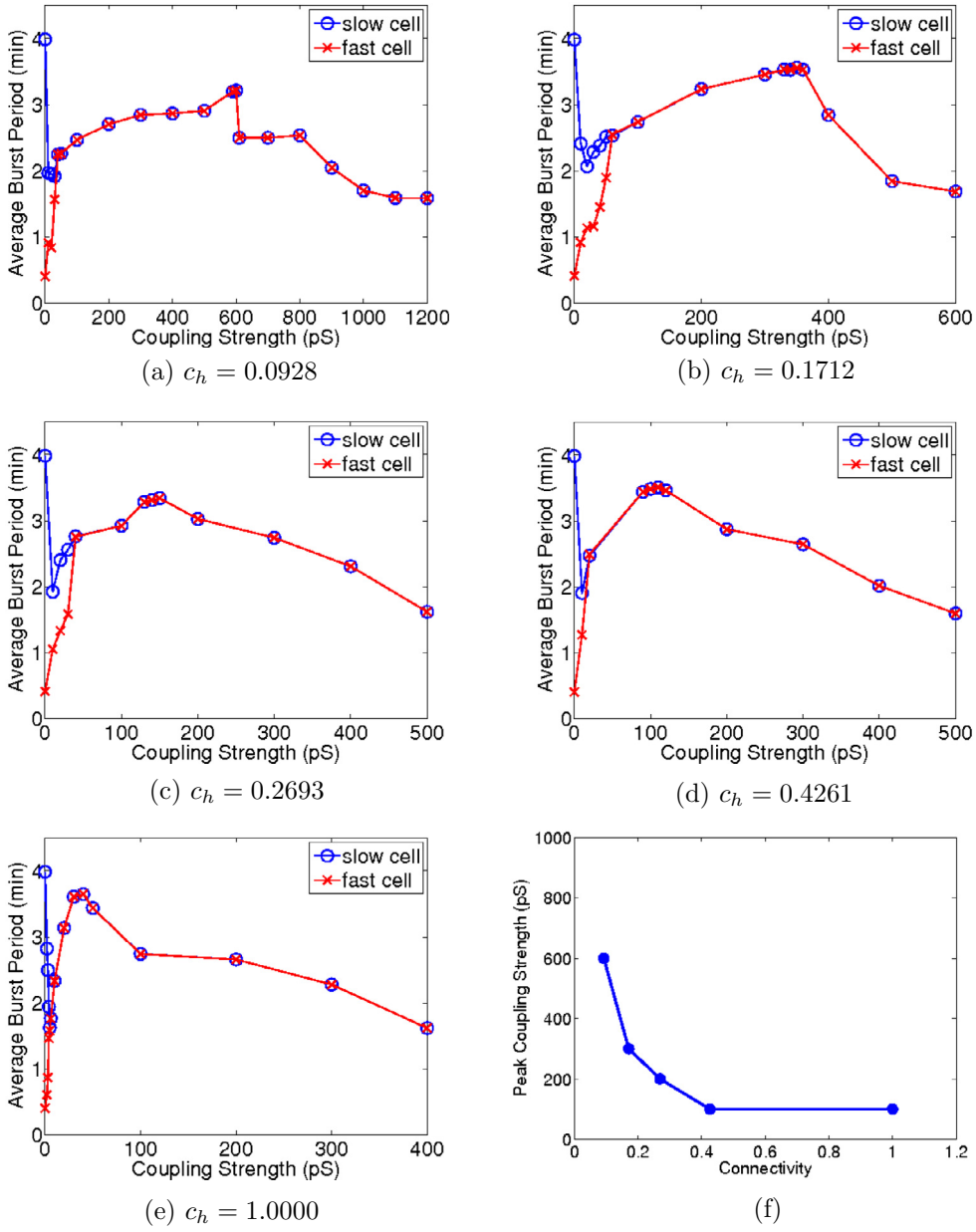


Fig. 7. Average burst period over a range of coupling strengths for cells for the seven variable model: (a) grouped structure, (b) split grouped structure, (c) layered middle structure, (d) layered split structure, (e) equally distributed structure. (f) Peak coupling strength vs. connection heterogeneity.

Table 4
Synchronization, peak, and leveled coupling strengths for each islet structure for the seven variable model.

Coupling strength	Grouped ($c_h = 0.0928$)	Split grouped ($c_h = 0.1712$)	Layered middle ($c_h = 0.2693$)	Layered split ($c_h = 0.4261$)	Equal ($c_h = 1.0000$)
Synchronization (pS)	100	60	40	20	6
Peak (pS)	600	340	200	140	40
Leveled (pS)	1100	600	500	500	400

We can conclude from these complete studies of all available numerical methods for the three variable model that the memory modified `ode15s` with sparse Jacobian, sparsity pattern of the Jacobian, and the AD provided Jacobian are essentially equally good and are at least two orders of magnitude faster than the other methods. On the scale of this improvement, using the AD provided Jacobian is not worse than hand-coding an analytically derived Jacobian, but the AD provided Jacobian is a lot easier and less error prone. Providing the sparsity pattern of the Jacobian is yet easier than providing the automatically differentiated Jacobian since it does not require the use and setup of ADiMat. Note, however, that since providing the sparsity pattern is not automated, providing the incorrect pattern may result in a drastic increase in runtime.

5.2. Seven variable model

Table 7 contains the runtimes for the coupling strengths from Table 4 for all five islet structures using the method of memory modified `ode15s` with AD provided Jacobian for the seven variable model. We observe that for analogous simulations the runtimes are

Table 5
Runtimes in seconds for the three variable model for all islet structures using memory modified `ode15s` and sparse Jacobian.

N	Grouped	Split grouped	Layered middle	Layered split	Equally distributed
(a) Synchronization coupling strength					
2	12	16	16	19	18
3	16	24	23	28	24
4	19	26	28	32	34
5	24	34	37	41	39
6	37	42	54	54	56
7	45	55	68	74	69
8	61	76	86	102	101
9	86	104	130	134	134
10	105	152	187	215	205
(b) Peak coupling strength					
2	7	7	5	8	4
3	9	8	12	9	4
4	10	10	8	9	7
5	12	13	12	10	9
6	10	11	17	14	13
7	15	14	27	22	18
8	24	22	35	29	28
9	39	39	46	44	29
10	64	51	63	59	57
(c) Leveled coupling strength					
2	7	7	5	7	8
3	8	8	11	8	8
4	10	10	12	10	10
5	13	12	12	12	13
6	18	18	15	17	17
7	26	24	29	24	25
8	34	33	37	33	37
9	49	49	48	44	50
10	68	71	64	67	75

approximately the same for each islet structure. We also observe that the runtimes for the simulations using synchronization coupling strength is greater than the runtimes for the other coupling strengths for all five islet structures. For islet structures with larger connection heterogeneities (layered split and equally distributed) the runtime for simulations using the leveled coupling strength is greater than the runtime for simulations using the peak coupling strength. However, for smaller connection heterogeneities simulations using the peak coupling strength has a greater runtime than simulations using the leveled coupling strength. We again proceed with the equally distributed islet structure as example in the following.

Table 8 contains the runtimes for the equally distributed islet structure using all available numerical methods. Since it is too complicated to analytically compute the Jacobian of the seven variable model, results for the dense and sparse Jacobians are not available for the seven variable model. First of all, Matlab's `ode45` returns an invalid solution, so it cannot be used for this model; even tightening the error tolerance on this solver does not resolve this problem. This is a cautionary tale about applying a non-stiff solver to a stiff problem. The error estimated in this well-known and reliable solver is not able to detect that it is producing garbage. For simulations that did not run out of memory, `ode113` has a greater runtime than any of the numerical methods using `ode15s`. This demonstrates dramatically the need for a stiff solver for this model. Among the stiff solvers, we observe that the original `ode15s` can run out of memory for larger N. For the cases where it does not run out of memory, providing the AD Jacobian often achieves a slightly faster runtime. Providing the sparsity pattern of the Jacobian to the original `ode15s` achieves yet faster runtimes for small N, though as N increases the runtime also generally increases in comparison to AD Jacobian. In fact, for the synchronization coupling strength for $N = 7$ the AD Jacobian has a runtime of approximately 2 days whereas the run where the sparsity pattern is provided takes longer than the maximum time of 5 days allowed on

Table 6
Runtimes in seconds for the three variable model for the equally distributed islet structure using all numerical methods, where O.M. signifies a simulation that ran out of memory.

	Original `ode15s`					Memory modified `ode15s`					`ode45`	`ode113`
N	Num.Jac.	Dense Jac.	Sparse Jac.	Pattern	AD	Num.Jac.	Dense Jac.	Sparse Jac.	Pattern	AD		
(a) Synchronization coupling strength												
2	28	19	19	22	38	24	18	18	21	36	13	16
3	59	51	47	57	60	37	27	24	28	37	19	92
4	261	229	223	235	230	68	41	34	37	46	39	793
5	972	737	719	822	690	113	68	39	44	50	81	2551
6	2596	2550	2289	2422	2208	277	184	56	60	73	182	8115
7	7759	7206	6720	7902	6724	586	408	69	75	91	355	26,224
8	13,622	12,767	12,148	13,463	12,419	1274	912	101	105	138	608	43,869
9	37,943	35,406	37,053	36,537	33,804	2262	1732	134	133	168	1064	68,612
10	68,644	62,374	60,667	64,614	61,993	5009	3774	205	185	271	1633	O.M.
(b) Peak coupling strength												
2	5	4	4	5	8	5	4	4	5	7	7	7
3	8	5	5	6	8	7	5	4	5	7	9	24
4	23	16	13	15	17	16	9	7	8	10	14	117
5	58	44	36	39	40	31	16	9	11	13	27	481
6	179	152	104	111	110	86	46	13	16	19	52	1394
7	508	449	323	347	333	197	129	18	20	27	102	4607
8	1028	865	601	627	617	456	293	28	29	44	157	7480
9	1570	1294	963	856	913	676	440	29	29	47	221	7958
10	4140	3588	2268	2245	2296	1813	1276	57	53	89	354	14,444
(c) Leveled coupling strength												
2	10	7	8	9	16	10	7	8	8	15	34	49
3	20	12	11	14	20	17	9	8	10	16	74	544
4	48	27	25	28	35	33	13	10	12	20	216	3324
5	117	84	70	72	78	64	25	13	15	24	580	14,263
6	295	204	158	195	171	131	65	17	20	32	1356	128,185
7	806	749	567	656	609	239	165	25	30	35	2846	O.M.
8	1545	1324	1015	1136	992	467	358	37	37	45	6965	O.M.
9	4432	3801	3085	3567	3590	936	746	50	53	66	8105	O.M.
10	7649	7049	4992	5967	5528	2061	1725	75	73	98	11,249	O.M.

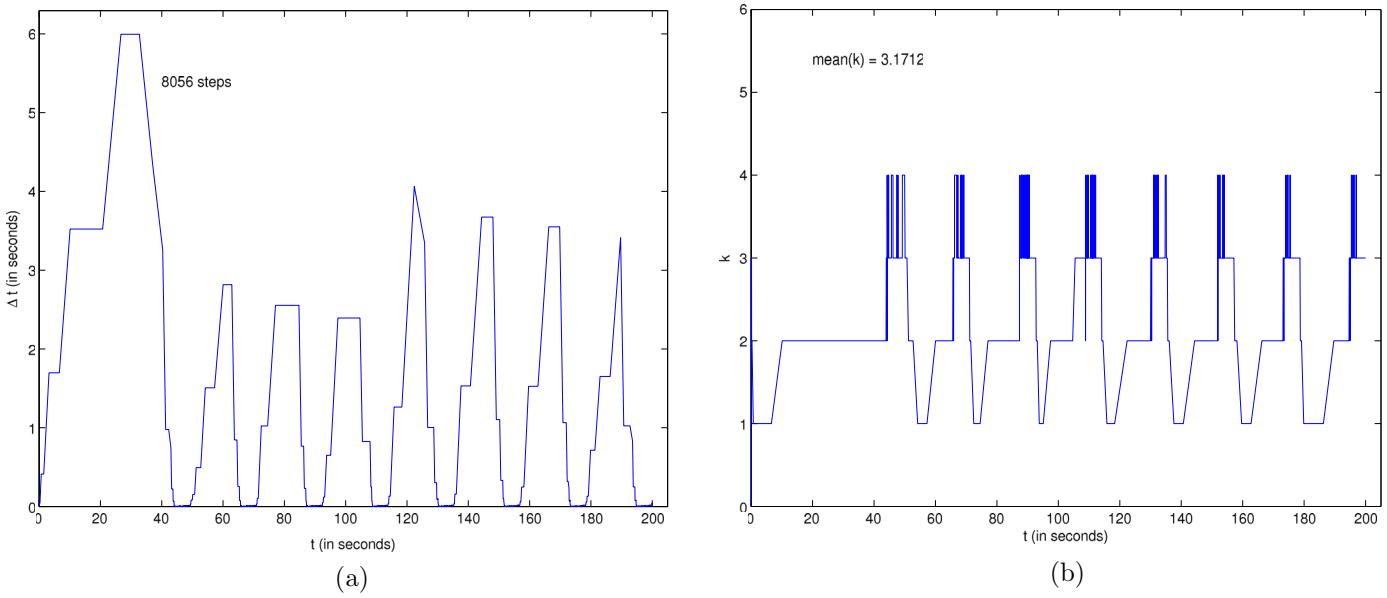


Fig. 8. Numerical performance data for the three variable model for a $10 \times 10 \times 10$ computational islet with peak coupling strength using memory modified `ode15s` with sparse Jacobian: (a) time step Δt vs. t and (b) method order k vs. t.

Table 7
Runtimes in seconds for the seven variable model for all islet structures using memory modified `ode15s` and AD Jacobian.

N	Grouped	Split grouped	Layered middle	Layered split	Equally distributed
(a) Synchronization coupling strength					
2	23	34	39	43	166
3	34	30	45	61	98
4	56	76	76	91	153
5	107	104	98	132	132
6	181	188	131	199	286
7	302	305	232	273	252
8	494	492	368	433	629
9	673	678	400	577	511
10	1075	995	597	816	1406
(b) Peak coupling strength					
2	81	86	22	56	24
3	67	57	70	41	41
4	63	67	56	56	42
5	55	67	73	59	61
6	91	66	66	82	98
7	134	94	195	148	135
8	324	249	210	242	230
9	354	345	306	378	321
10	993	576	552	508	487
(c) Leveled coupling strength					
2	94	92	48	98	92
3	91	82	107	90	104
4	98	82	71	107	126
5	99	76	76	75	180
6	95	114	139	114	139
7	116	124	174	162	214
8	184	160	296	318	322
9	278	217	405	342	398
10	686	466	551	560	545

our system. But clearly, neither providing the sparsity pattern of the Jacobian nor providing the AD Jacobian fixes the fundamental problem of the original `ode15s` running out of memory. This necessitates the use of the memory modified `ode15s`. Already with a numerical Jacobian, the modified `ode15s` is significantly faster, in addition to never running out of memory, also for cases, where the original `ode15s` does not run out of memory. In turn, both providing the sparsity pattern of the Jacobian and providing an AD Jacobian to the memory modified `ode15s` achieves dramatically faster runtimes, both essentially on the same order of magnitude. Thus, with the combination of an AD provided Jacobian or the sparsity pattern of the Jacobian with the memory modified `ode15s`, we can solve problems for all desired values of N and do so in very reasonable times even for the largest islet and across the full range of coupling strengths.

To conclude, Table 9 contains the observed memory usage for the same cases as studied in Table 8. We study the memory usage here, since many of our simulations run out of memory and to provide some indication what size of memory might be needed to perform simulations for the full seven variable model. As expected, the largest improvement in memory usage is observed by using the memory modified `ode15s` instead of the original one. The improvement is so large that unlike when using the original `ode15s`, we are now able to run all simulations without running out of memory. The memory usage is smallest for the memory modified `ode15s` with AD Jacobian or sparsity pattern of the Jacobian. This demonstrates that there is also a memory advantage to providing a Jacobian function or sparsity pattern to the ODE solver.

6. Discussion

The most important summaries of the numerical performance results in Section 5 are given in Table 6 for the three variable model and Table 8 for the seven variable model. These tables compare the runtimes of all numerical methods that are available for the respective models. Both tables drive home the need for using an appropriate ODE solver designed for stiff problems, since both popular non-stiff solvers (`ode45` and `ode113`) may not be able to handle the problem or are exceedingly inefficient eventually, when the problem becomes complex, such as the seven variable model that we are really interested in.

For the three variable model, where timing results for all numerical methods are available for values of N considered, the greatest improvement in runtime over a numerically approximated Jacobian results from having a sparse Jacobian matrix to use. This is true for any of the three methods that provide this. It is interesting that providing the sparsity pattern of the Jacobian, which gives a numerically approximated Jacobian, is faster than the true Jacobian functions, whether hand-coded sparse or AD provided. These comparisons among the methods apply both to the original `ode15s` and to the memory modified `ode15s`, but much more so to the latter, where the methods with

Table 8
Runtimes in seconds for the seven variable model for the equally distributed islet structure using all available numerical methods, where O.M. signifies a simulation that ran out of memory, E.T. signifies a simulation that ran for over 5 days, and I.S. represents a simulation that provided an invalid solution.

	Original `ode15s`			Memory modified `ode15s`			`ode45`	`ode113`
N	Num. Jac.	Pattern	AD	Num. Jac.	Pattern	AD		
(a) Synchronization coupling strength								
2	171	152	261	72	62	166	I.S.	302
3	1092	1083	1166	99	70	98	I.S.	3993
4	6830	7240	7071	283	108	153	I.S.	26,226
5	18,630	18,869	18,222	616	112	132	I.S.	78,473
6	68,546	72,427	66,111	2279	192	286	I.S.	O.M.
7	182,047	E.T.	190,144	3431	216	252	I.S.	O.M.
8	O.M.	O.M.	O.M.	11,698	381	629	I.S.	O.M.
9	O.M.	O.M.	O.M.	14,383	408	511	I.S.	O.M.
10	O.M.	O.M.	O.M.	45,500	694	1406	I.S.	O.M.
(b) Peak coupling strength								
2	24	23	31	16	17	24	I.S.	75
3	158	136	162	37	25	41	I.S.	1029
4	770	728	769	92	35	42	I.S.	5644
5	3365	2996	2992	341	53	61	I.S.	20,006
6	10,102	8447	9366	1015	83	98	I.S.	72,840
7	35,069	31,575	29,591	2486	121	135	I.S.	O.M.
8	58,785	58,057	51,590	6401	180	230	I.S.	O.M.
9	O.M.	E.T.	81,200	14,087	254	321	I.S.	O.M.
10	O.M.	O.M.	O.M.	29,243	356	487	I.S.	O.M.
(c) Leveled coupling strength								
2	43	36	108	32	24	92	I.S.	1066
3	217	169	240	77	30	104	I.S.	24,187
4	890	727	763	224	40	126	I.S.	168,222
5	2687	2293	2281	673	56	180	I.S.	O.M.
6	12,498	13,166	11,750	1428	118	139	I.S.	O.M.
7	39,893	34,844	36,075	3768	152	214	I.S.	O.M.
8	79,623	70,794	78,744	9563	221	322	I.S.	O.M.
9	O.M.	O.M.	O.M.	16,971	312	398	I.S.	O.M.
10	O.M.	O.M.	O.M.	38,551	438	545	I.S.	O.M.

Table 9
Observed memory usage in GB for the seven variable model for the equally distributed islet structure using all available numerical methods, where O.M. signifies a simulation that ran out of memory, E.T. signifies a simulation that ran for over 5 days, and I.S. represents a simulation that provided an invalid solution.

	Original `ode15s`			Memory modified `ode15s`			`ode45`	`ode113`
N	Num. Jac.	Pattern	AD	Num. Jac.	Pattern	AD		
(a) Synchronization coupling strength								
2	1.6	1.5	1.3	1.2	1.1	1.0	I.S.	1.8
3	1.9	2.6	2.6	1.3	1.3	1.1	I.S.	2.6
4	4.0	5.9	5.4	1.8	1.5	1.5	I.S.	10.0
5	8.8	8.4	8.4	2.4	2.6	1.7	I.S.	17.9
6	16.4	15.7	15.9	3.6	3.1	2.2	I.S.	O.M.
7	23.5	E.T.	23.0	4.6	4.1	4.2	I.S.	O.M.
8	O.M.	O.M.	O.M.	6.2	7.5	5.1	I.S.	O.M.
9	O.M.	O.M.	O.M.	8.3	6.1	6.0	I.S.	O.M.
10	O.M.	O.M.	O.M.	11.7	8.2	8.2	I.S.	O.M.
(b) Peak coupling strength								
2	1.3	1.2	1.2	1.2	1.1	1.1	I.S.	1.8
3	1.7	1.6	1.6	1.3	1.2	1.2	I.S.	2.6
4	2.7	2.4	1.8	1.6	1.3	1.3	I.S.	5.3
5	4.4	4.1	4.1	1.8	1.5	1.5	I.S.	9.8
6	7.3	6.7	6.9	2.2	1.8	1.8	I.S.	16.8
7	11.2	10.7	10.5	2.8	2.3	2.3	I.S.	O.M.
8	15.8	15.4	15.7	3.7	2.8	2.9	I.S.	O.M.
9	O.M.	E.T.	22.6	5.6	4.0	3.6	I.S.	O.M.
10	O.M.	O.M.	O.M.	8.0	6.5	5.2	I.S.	O.M.
(c) Leveled coupling strength								
2	1.3	1.2	1.2	1.2	1.0	1.0	I.S.	2.6
3	1.8	1.7	1.6	1.4	1.2	1.2	I.S.	8.9
4	2.1	2.4	2.4	1.5	1.3	1.3	I.S.	21.9
5	3.9	3.8	3.8	1.8	1.6	1.6	I.S.	O.M.
6	7.9	7.6	7.6	2.4	1.8	2.0	I.S.	O.M.
7	11.9	11.1	11.3	3.0	2.4	2.3	I.S.	O.M.
8	20.2	17.5	19.3	4.0	3.2	2.8	I.S.	O.M.
9	O.M.	O.M.	O.M.	5.6	3.6	4.2	I.S.	O.M.
10	O.M.	O.M.	O.M.	8.1	4.5	5.7	I.S.	O.M.

a sparse Jacobian are all two orders of magnitude faster than with using a numerical Jacobian. However, the overriding observation from Table 6 has to be that all cases of the memory modified `ode15s` are faster than any using the original `ode15s`, with runtimes of hours being eventually reduced to a few minutes. This points to the crucial importance of memory management to solve large problems, which is one important purpose of studying the seven variable model.

For the seven variable model, the results make it clear that the memory modified `ode15s` is necessary to solve problems for large islets with 1000 or more cells and that this modified solver also improves the runtime significantly in all cases. Additional dramatic improvement in runtimes can be observed when providing the sparsity pattern of the Jacobian or the AD provided Jacobian in the memory modified `ode15s`. For instance for 1000 cells, Table 8 shows that with these tools, simulation times of the order of a 10 min (600 s) suffice to obtain solutions with the same fidelity as with simulations taking 10 h (36,000 s). Thus, simulations for sophisticated models, for which it is unworkable to manually program the Jacobian, are possible in Matlab, a language that additionally has the benefit to allow for reliable programming of complicated model equations.

For comparison, Pu et al. [17, p. 7] report a runtime of 26 h for a simulation with 1000 cells up to 2000 s. Our simulations in Table 8 have a final time of 500,000 ms or 500 s, so note one fourth of 26 h as 6.5 h or 23,400 s, when comparing to the corresponding $N = 10$ results in Table 8. The results of [17] using LSODE programmed in Fortran 90 need to be compared then to the memory modified `ode15s` with numerical Jacobian, since LSODE does not store results at all time steps. The Matlab results in Table 8 can then be observed to be of the order of a factor of two slower than the LSODE results in Ref. [17]. This demonstrates that the algorithm in Matlab's `ode15s` can be an efficient simulation tool, even with a numerical Jacobian, i.e., without the user providing a Jacobian function, provided its memory management is improved. When combining the memory modified `ode15s` with the AD Jacobian, it clearly outperforms LSODE in Fortran 90, by being faster of the order of 20 times. This in turn indicates the wide applicability of our conclusions concerning an AD provided Jacobian as a method, since also LSODE could perform faster, if the user provided it with a Jacobian function. This would be possible in the same way as described here for Matlab by using automatic differentiation tools for source-code languages Fortran or C.

The most important summaries of the physiological results in Section 4 are given in Fig. 4 for the three variable model and in Fig. 7 for the seven variable model. The nonmonotonic average burst period between fast and slow cells for increasing coupling agrees with the coupled cell work of Sherman and Rinzel [21]. Figs. 4(f) and 7(f) show that interestingly, the arrangement of the connected fast and slow heterogeneous cells impacts the peak bursting period monotonically. For increasing connection heterogeneity from low, segregated, to high, connected only to opposite cell type, the peak bursting period decreases.

Recent work has shown an interesting result that modified KATP channels can be compensated for by gap junctional coupling knockdown [12]. We can make a similar comparison with a change in connection heterogeneity compensating with an altered gap junctional coupling strength. For example, in beta cell fission or neogenesis new beta cells incorporate into existing islet structures [10]. To balance a change in the connectivity, as measured by the connection heterogeneity, gap junctional coupling may be able to or may even need to adjust to manage any altered average burst period. The local gap junctional connectivity in this case may be critical, which can be adjusted in the model. Our work suggests qualitatively if not quantitatively how changes in connection heterogeneity and coupling strength can be compensatory.

While we selected fast and slow heterogeneity parameters to study the numerical framework and coupling strengths and arrangements, we would expect similar emergent behavior in the entire islet for a range of individual β-cell bursting periods between fast and slow and with sufficiently strong coupling. For weaker coupling, we might expect a loss of regular bursting pattern. Furthermore, the arrangement of the heterogeneous cells along with individual cell characteristics and coupling strength may play a key role in defining the emergent islet behavior, especially when extended to other cell types such as α- and δ-cells.

Acknowledgments

We thank the two anonymous referees for their thorough reviews and invaluable input.

The hardware used in the 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 support from the University of Maryland, Baltimore County (UMBC). See http://www.umbc.edu/hpcf for more information on HPCF and the projects using its resources.

References

[1] I. Ajmera, M. Swat, C. Laibe, N.L. Novere, V. Chelliah, The impact of mathematical modeling on the understanding of diabetes and related complications, CPT: Pharmacomet. Syst. Pharmacol. 2 (2013) e54.
[2] R. Bertram, A. Sherman, A calcium-based phantom bursting model for pancreatic islets, Bull. Math. Biol. 66 (2004) 1313–1344.
[3] R. Bertram, A. Sherman, L.S. Satin, Metabolic and electrical oscillations: partners in controlling pulsatile insulin secretion, Am. J. Physiol. Endocrinol. Metab. 293 (4) (2007) E890–E900.
[4] C.H. Bischof, H.M. Bücker, B. Lang, A. Rasch, A. Vehreschild, Combining source transformation and operator overloading techniques to compute derivatives for MATLAB programs, in: Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2002), IEEE Computer Society, Los Alamitos, CA, USA, 2002.
[5] Centers for Disease Control and Prevention, National diabetes fact sheet: national estimates and general information on diabetes and prediabetes in the United States, 2011. http://www.cdc.gov/diabetes/pubs/pdf/ndfs_2011.pdf (accessed 06.11.14).
[6] T.R. Chay, J. Keizer, Minimal model for membrane oscillations in the pancreatic beta-cell, Biophys. J. 42(2) (1983) 181–190.
[7] S. Conde, T. Lebair, C. Raastad, V. Smith, K. Stern, D. Trott, M.K. Gobbert, B.E. Peercy, A. 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.
[8] G. de Vries, A. Sherman, H.R. Zhu, Diffusively coupled bursters: effects of cell heterogeneity, Bull. Math. Biol. 60 (1998) 1167–1200.
[9] G. Gearhart, S. Jiang, T.J. May, J. Pan, S. Khuvis, M.K. Gobbert, B.E. Peercy, A. Sherman, Dynamics of computational islet simulations: Islets with majority mutated open K_{ATP} channels retain bursting, Lett. Biomath. (2014) [Invited paper for the inaugural issue of this journal, published originally in the Proceedings of the Sixth Symposium on BEER 2013].
[10] J. Jo, G. Kilimnik, A. Kim, C. Guo, V. Periwal, M. Hara, Formation of pancreatic islets involves coordinated expansion of small islets and fission of large interconnected islet-like structures, Biophys. J. 101(3) (2011) 565–574.
[11] S. Khuvis, Efficiency improvements in numerical methods for studying connectivity in a model of a pancreatic islet of heterogeneous beta cells (M.S. thesis), University of Maryland, Baltimore County, 2013.
[12] L.M. Nguyen, M. Pozzoli, T.H. Hraha, R.K.P. Benninger, Decreasing Cx36 gap junction coupling compensates for overactive KATP channels to restore insulin secretion and prevent hyperglycemia in a mouse model of neonatal diabetes, Diabetes 63 (5) (2014) 1685–1697.
[13] A. Nittala, S. Ghosh, X. Wang, Investigating the role of islet cytoarchitecture in its oscillation using a new beta-cell cluster model, PLoS ONE 2(10) (2007) e983.
[14] C.S. Nunemaker, J.F. Dishinger, S.B. Dula, R. Wu, M.J. Merrins, K.R. Reid, A. Sherman, R.T. Kennedy, L.S. Satin, Glucose metabolism, islet architecture, and genetic homogeneity in imprinting of [Ca2+]i and insulin rhythms in mouse islets, PLoS ONE 4 (12) (2009) e8428.
[15] C.S. Nunemaker, L.S. Satin, A tale of two rhythms: a comparative review of the pulsatile endocrine systems regulating insulin and GnRH secretion, Cellsci. Rev. 2 (1) (2005) 92–126.
[16] J. Pan, G. Gearhart, S. Jiang, T.J. May, Loss of metabolic oscillations in a multicellular computational islet of the pancreas, UMBC Rev.: J. Undergrad. Res. 15 (2014) 31–53.
[17] Y. Pu, S. Lee, D. Samuels, L. Watson, Y. Cao, The effect of unhealthy beta-cells on insulin secretion in pancreatic islets, BMC Med. Genom. 6 (Suppl. 3) (2013) S6.
[18] P. Rorsman, G. Trube, Calcium and delayed potassium currents in mouse pancreatic beta-cells under voltage-clamp conditions, J. Physiol. 374 (1986) 531–550.

[19] L.F. Shampine, M.W. Reichelt, The MATLAB ODE suite, SIAM J. Sci. Comput. 18 (1) (1997) 1–22.
[20] A. Sherman, Contributions of modeling to understanding stimulus-secretion coupling in pancreatic beta-cells, Am. J. Physiol. Endocrinol. Metab. 271(2) (1996) E362–372.
[21] A. Sherman, J. Rinzel, Rhythmogenic effects of weak electrotonic coupling in neuronal models, Proc. Natl. Acad. Sci. U.S.A. 89 (1992) 2471–2474.
[22] A. Sherman, L. Xu, C. Stokes, Estimating and eliminating junctional current in coupled cell populations by leak subtraction. A computational study, J. Membr. Biol. 143 (1) (1995) 79–87.
[23] P. Smolen, A model for glycolytic oscillations based on skeletal muscle phosphofructokinase kinetics, J. Theor. Biol. 174 (2) (1995) 137–148.
[24] K. Tsaneva-Atanasova, A. Sherman, Accounting for near-normal glucose sensitivity in Kir6.2[AAA] transgenic mice, Biophys. J. 97 (9) (2009) 2409–2418.