On the Performance of an Implicit–Explicit Runge–Kutta Method in Models of Cardiac Electrical Activity

Raymond J. Spiteri* and Ryan C. Dean

***Abstract*—Mathematicalmodels of electric activity in cardiac tissue are becoming increasingly powerful tools in the study of cardiac arrhythmias. Considered here are mathematical models based on ordinary differential equations (ODEs) that describe the ionic currents at the myocardial cell level. Generating an efficient numerical solution of these ODEs is a challenging task, and, in fact, the physiological accuracy of tissue-scale models is often limited by the efficiency of the numerical solution process. In this paper, we examine the efficiency of the numerical solution of four cardiac electrophysiological models using implicit–explicit Runge–Kutta (IMEX-RK) splitting methods. We find that variable step-size implementations of IMEX-RK methods (ARK3 and ARK5) that take advantage of Jacobian structure clearly outperform the methods commonly used in practice.**

***Index Terms*—Efficient numerical methods, implicit–explicit Runge–Kutta (IMEX-RK) methods, ordinary differential equations (ODEs), Rush–Larsen method, simulation of electrophysiological models, splitting methods.**

I. Introduction

COMPUTER simulation is becoming an important tool in cardiovascular research. Mathematical models of the heart can be used to simulate heart conditions and the effects of certain drugs designed to treat them. Presently, the development of a drug often costs hundreds of millions of dollars [1]. One aim of computer simulation is to reduce this cost, e.g., by reducing the number of physical experiments needed in designing a drug.

Electrophysiological models of the heart describe how electricity flows through the heart, controlling its contraction. In this paper, we consider four mathematical models for the flow of ionic currents at the myocardial cell level. The models consist of systems of ordinary differential equations (ODEs). Cardiac electrophysiological models are often based on the Nobel prize-winning work of Hodgkin and Huxley [2] in the 1950s that modeled neural tissue mathematically as a circuit. Modern cardiac electrophysiological models adapt the work of Hodgkin and Huxley to describe electrical activity in the heart and include data gathered from experiments to form models with increasing physiological accuracy.

A major barrier to obtaining the most useful data from tissue-scale electrophysiological models of the heart is the challenge of performing the simulations efficiently. Often, the physiological accuracy of the mathematical model must be sacrificed for a simulation to become feasible; see, e.g., [3]–[6]. The ODE systems describing the cellular dynamics in these models are nonlinear and stiff (see, e.g., [7]). The consequence of stiffness is that the speed of the solution process is limited by considerations of numerical stability instead of accuracy. Hence, the solution process may be made more efficient through the use of appropriate numerical algorithms.

In this paper, we investigate the efficiency of implicit–explicit Runge–Kutta (IMEX-RK) splitting methods (see, e.g., [8]) for the simulation of four cardiac electrophysiological models. Cardiac electrophysiological models contain linear and nonlinear terms as well as stiff and nonstiff terms, so IMEX methods are a natural choice for their numerical solution. An IMEX-RK method uses both an implicit RK (IRK) method and an explicit RK (ERK) method to approximate the solution to an ODE.

The use of splitting methods for cardiac simulations is quite common, particularly for simulations in which the ODEs for myocardial cell models are coupled with partial differential equations (PDEs), describing the flow of electricity through myocardial tissue. For example, a second-order accurate operator splitting method for the monodomain model of myocardiac tissue is studied in [9]. The coupled nonlinear PDE model is split into an uncoupled linear parabolic PDE and a system of nonlinear ODEs. The PDE is then solved with either the forward Euler method or an alternating direction implicit method, and the ODE system is solved with the Rush–Larsen method [10]; see also later. This splitting is also used with first- and second-order IMEX schemes in [11] for both the monodomain and bidomain models. An implicit method is used for the PDE(s), and an explicit method is used to integrate the ODEs. The use of both a second-order Crank–Nicolson method and a first-order semi-implicit method to solve the bidomain model is investigated in [12]. These methods were optimized by splitting the nonlinear cell model into two components, one with positive eigenvalues and another with negative eigenvalues. This work is extended in [13], where the cell model is split from the PDEs and solved with the backward Euler method and further optimized by deriving an explicit scheme to solve the ODEs. The work in [12] is extended in [14] to create a first-order method to simulate the flow of electricity in the heart and the surrounding

Manuscript received June 12, 2007; revised October 23, 2007. This work was supported in part by NSERC Canada and MITACS. *Asterisk indicates corresponding author.*

*R. J. Spiteri is with the Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan S7N 5A9, Canada (email: spiteri@cs.usask.ca).

R. C. Dean is with the Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan S7N 5A9, Canada (e-mail: r.dean@usask.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBME.2007.914677

torso. In [15], the ODEs are separated as in [9], and then, the approach from [14] is used to solve the PDEs in the heart and torso, resulting in a second-order method. The work of [12] is also extended in [16] to include additional complexities such as rotational anisotropy and blood layer boundary conditions. Other numerical methods used to solve the ODEs in such models include singly diagonally implicit RK (SDIRK) methods, fully implicit RK methods, and multistep methods based on backward differentiation formulas (BDFs); see later and also, e.g., [17].

In this paper, we consider four mathematical models of cardiac electrophysiology: the Luo–Rudy model of guinea pig ventricular tissue [18], the Courtemanche *et al*. model of human atrial tissue [19], the Winslow *et al*. model of canine ventricular tissue [20], and the Puglisi–Bers model of rabbit ventricular tissue [21]. We perform a thorough comparison of the performance of two specific IMEX-RK methods, denoted here as ARK3 and ARK5 [22], respectively, to numerical methods commonly used in practice for simulating these four models. We also consider problem-specific optimizations to the IMEX-RK methods described. The results presented have the obvious limitation that they are based on single-cell models, and thus, further studies are necessary to fully establish their applicability to tissue-scale simulations. However, the present work can provide guidance for the development of efficient splitting methods for the simulation of PDE models that use variable step-size ODE solvers for the cellular dynamics.

The rest of this paper is organized as follows. In Section II, we give a brief introduction to mathematical models of electrical activity in the heart. In Section III, we give a brief overview of the numerical methods used in this paper for the numerical solution of the ODEs arising from these models. In Section IV, we discuss the results of the numerical experiments. We find that a variable step-size implementation of an IMEX-RK method with a customized linear system solver clearly outperforms all the commonly used numerical methods of which we are aware. Finally, in Section V, we give some conclusions and directions for future research.

II. Electrical Activity of the Heart

Because of their intricacy, obtaining physiologically accurate mathematical models is a difficult task. A further challenge to obtaining physiological accuracy is that of performing the simulation efficiently. To move effectively beyond models for one cell, enough cells must be included in the model to realistically approximate the geometry and physiology of the heart. Because the heart has approximately 10^{10} cells [23], any realistic simulation will have enough cells (or clusters of cells) to dramatically magnify any inefficiencies in the numerical method. This has forced some researchers to reduce the physiological accuracy of their models to allow the simulation to be performed within an acceptable amount of time; see, e.g., [3]–[6]. The models are numerically *stiff*, and so standard (explicit) numerical methods are often unable to provide efficient simulations. If the efficiency of the simulation process can be significantly improved, then greater physiological accuracy, and subsequently, more useful data can be obtained.

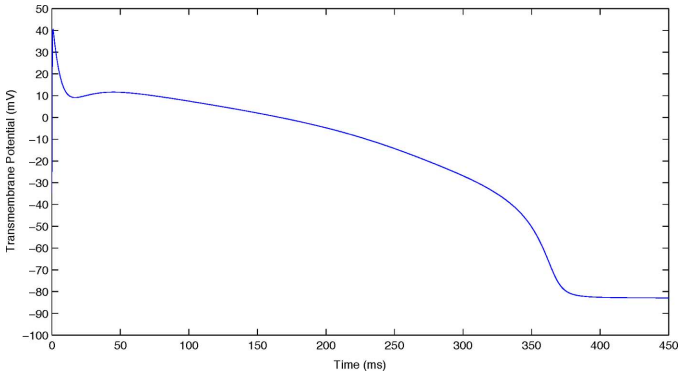

Fig. 1. Transmembrane potential over time in the Luo–Rudy model. This is a numerical solution produced by ARK5 for (1) coupled with seven other ODEs [18] via the $I_{\rm ion}$ current.

A. Luo–Rudy Model of Guinea Pig Ventricular Tissue

In 1991, Luo and Rudy developed a model of guinea pig ventricular action potentials based on a previous model from Beeler and Reuter [24]. The Luo–Rudy model [18] extended the Beeler–Reuter model to include fast inward sodium and outward potassium currents to make the model more physiologically accurate. The general approach of these models is based on Hodgkin–Huxley type formalism [2]; the Luo–Rudy model itself consists of eight nonlinear ODEs.

For an individual cardiac cell, we have that the transmembrane potential V_m satisfies [18]

$$\frac{dV_m}{dt} = -\frac{1}{C_m}(I_{\rm ion} + I_{st}) \tag{1}$$

where C_m is the membrane capacitance, $I_{\rm ion}$ is the total transmembrane ionic current, and I_{st} is the stimulus current. An example of the evolution of V_m over time for this model is given in Fig. 1.

The Luo–Rudy model contains six ionic currents that are determined by six gating variables [18]. The evolution of each gating variable y is governed by a nonlinear ODE involving rate parameters α_y and β_y in the general form

$$\frac{dy}{dt} = \frac{y_\infty - y}{\tau_y} \tag{2}$$

where

$$y_\infty = \frac{\alpha_y}{\alpha_y + \beta_y} \quad \text{and} \quad \tau_y = \frac{1}{\alpha_y + \beta_y}.$$

The remaining ODE in the Luo–Rudy model describes calcium concentration in the cell

$$\frac{d([{\rm Ca}]_i)}{dt} = -10^{-4} I_{\rm si} + 0.07(10^{-4} - [{\rm Ca}]_i) \tag{3}$$

where $[{\rm Ca}]_i$ is the intracellular calcium concentration and $I_{\rm si}$ is the slow inward calcium current [18]. The six gating equations of the form (2) are coupled with (1) and (3) to form the complete Luo–Rudy model. Full details of the model can be found in [18].

In 1994, Luo and Rudy published an improvement to this model, now known as the Luo–Rudy Phase II model [25], [26]. This model includes the actions of ionic pumps and changes in

ionic concentrations. It is a more physiologically accurate, yet more complicated model than (1)–(3), consisting of 14 ODEs. We do not consider the Luo–Rudy Phase II model in this paper.

B. Courtemanche et al. Model of Atrial Tissue

In 1998, Courtemanche *et al*. developed a model of human atrial action potentials [19]. It was developed in response to findings that show that there are important differences in human action potentials when compared to those of other mammals frequently used in models. Courtemanche *et al*. developed this model with human data supplemented with animal data when needed. The Courtemanche *et al*. model is an extension of the Luo–Rudy Phase II model. It consists of 21 ODEs. Full details of the model can be found in [19].

C. Winslow et al. Model of Canine Ventricular Tissue

In 1999, Winslow *et al*. developed a model of canine ventricular tissue [20]. This model is based on a guinea pig model that was an extension of the Luo–Rudy Phase II model. The Winslow *et al*. model was developed using experimental data to modify the guinea pig model so that it would simulate canine ventricular tissue. The Winslow *et al*. model is particularly detailed when describing the dynamics of Ca^{2+}, which is an important consideration in heart failure. It consists of 32 ODEs, making it the most complex of the models in this paper. Full details of the model can be found in [20].

D. Puglisi–Bers Model of Rabbit Ventricular Tissue

In 2001, Puglisi and Bers developed a model of rabbit ventricular tissue [21]. Although rabbit ventricular tissue is used frequently in experiment, no mathematical model had been previously developed for it. This model was adapted from the Luo–Rudy model to include data from the literature and from the joint laboratory of Puglisi and Bers. This model was designed to be a learning aid for students as well as a tool for researchers to reproduce experimental data via computer simulation. Thus, physiological accuracy was of paramount importance. The Puglisi–Bers model gives particular detail to calcium handling in order to accurately simulate heart failure. This model contains 17 ODEs. It is also referred to as the *LabHeart* [21] model. Full details of the model can be found in [21].

III. Numerical Methods

In this paper, we focus on the solution of the *initial-value problem* (IVP) for ODEs, defined as

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}\left(t, \mathbf{y}\right), \qquad \mathbf{y}(0) = \mathbf{y}_0 \tag{4}$$

on the interval $t \in [0, t_f]$. We now describe the numerical methods compared here. We also describe aspects of creating an efficient numerical simulation. Except for the fully implicit methods studied, all numerical experiments were run using `odeToJava` [27], a Java-based problem-solving environment for IVPs.[1] Details on how the fully implicit methods were run are given later.

A. Forward Euler Method

Arguably, the simplest numerical method for approximating the solution of (4) is *Euler's method*, or *Forward Euler* (FE). One step of FE from $(t_{n-1}, \mathbf{y}_{n-1})$ to $(t_n, \mathbf{y}_n)$ is given by

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \Delta t_n \mathbf{f}\left(t_{n-1}, \mathbf{y}_{n-1}\right)$$
$$t_n = t_{n-1} + \Delta t_n$$

where $\mathbf{y}_n \approx \mathbf{y}(t_n)$. The time step $\Delta t_n := t_n - t_{n-1}$ need not be constant for each n. It is well known that FE is first-order accurate [28], which is generally too low for efficiency in practice. Nonetheless, because of its simplicity, it is often used to simulate IVPs (4) associated with models of cardiac activity, and hence it serves as a useful benchmark against which to measure other numerical methods.

B. Explicit Runge–Kutta Methods

The FE method can be viewed as the simplest of a more general class of numerical methods for solving IVPs known as *explicit Runge–Kutta* (ERK) methods; see, e.g., [7]. The ERK methods aim to improve on the FE method by increasing the accuracy of the numerical solution by means of additional $\boldsymbol{f}$ evaluations (or *stages*) within a given time step. A general s-stage RK method has the form

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \Delta t_n \sum_{i=1}^{s} b_i \mathbf{k}_i$$

where for $i = 1, 2, \ldots, s$

$$\mathbf{k}_i = \mathbf{f}\left(t_{n-1} + \Delta t_n\, c_i, \mathbf{y}_{n-1} + \Delta t_n \sum_{j=1}^{s} a_{ij}\mathbf{k}_j\right)$$

and can be summarized via the *Butcher tableau* [7]

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array}.$$

A Runge–Kutta method is *explicit* if $\mathbf{A}$ is strictly lower triangular; otherwise, it is *implicit*. With ERK methods, the stages can be computed successively and their contributions combined to produce a high-order approximation at the end of the step. With IRK methods, a (generally) nonlinear system of equations must be solved at every time step for all stages simultaneously. However, because of their superior stability properties, IRK methods are well suited for stiff problems [7].

Perhaps the most popular high-order ERK method is the classical RK method, which is a four-stage, fourth-order ERK method, and which we denote by ERK4.

Another popular ERK method is the Dormand–Prince 5(4) (DP) pair, which is of order 5 and is coupled with a method of order 4 for error estimation and step-size control [28]. This method is the basis of the popular Matlab routine `ode45`.

[1] Presently, `odeToJava` supports only ERK and linearly implicit IMEX-RK methods.

C. Fully Implicit Methods

We include three fully implicit methods in our study. The first is arguably the simplest fully implicit method, the *backward Euler* (BE) method, a one-stage, first-order IRK method with Butcher tableau.

The second is the SDIRK method SDIRK4 of [29, p. 107]. This is a five-stage, L-stable method of order four with an embedded method of order 3.

The third method is the three-stage Radau IIA method RADAU5 of [29, p. 78]. It is a fully implicit RK method of order 5 and has the property of *stiff decay* [29].

We note that we only consider IRK and not BDF methods in the present study. In the context of splitting methods for mono- or bidomain models, the ODEs and PDEs are only advanced over small intervals in time before updated solution information is exchanged between systems. Results from [17] indicate that BDF methods were not competitive with IRK methods in terms of CPU time in such situations. Moreover, BDF methods require storage of the solution from past steps. This makes them less attractive than RK methods because this extra storage requirement may consequently reduce the overall resolution available for the simulation.

D. Implicit—Explicit Methods

When the right-hand side of an ODE can be written as the sum of two terms

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}_I\left(t, \mathbf{y}\right) + \mathbf{f}_E\left(t, \mathbf{y}\right) \tag{5}$$

it is often natural to consider approximating the contributions of $\mathbf{f}_E(t, \mathbf{y})$ and $\mathbf{f}_I(t, \mathbf{y})$ using different numerical methods. Such methods are known as *additive* methods. In general, when the right-hand side of an ODE can be written as the sum of n terms, these methods are called n-additive methods. When the constituent numerical methods are RK methods, then they are known as n-additive RK methods. Furthermore, if $d\mathbf{y}/dt = \mathbf{f}_E(t, \mathbf{y})$ is such that it is best approximated with an explicit method and $d\mathbf{y}/dt = \mathbf{f}_I(t, \mathbf{y})$ is such that it is best approximated with an implicit method, we may use an IMEX method in an attempt to approximate the solution to this ODE efficiently [8]. An example of when an IMEX method would be useful is when $\mathbf{f}_E(t, \mathbf{y})$ consists of nonstiff and/or nonlinear terms and $\mathbf{f}_I(t, \mathbf{y})$ consists of stiff and/or linear terms. Again, when the constituent implicit and explicit methods are RK methods, we have an IMEX-RK method.

For example [30], the combination of the FE and BE methods gives the IMEX-RK method

$$\mathbf{y}_n = \mathbf{y}_{n-1} + \Delta t_n \left(\mathbf{f}_E\left(\mathbf{y}_{n-1}\right) + \mathbf{f}_I\left(\mathbf{y}_n\right)\right).$$

More generally, an s-stage IRK method with coefficients $\mathbf{A}, \mathbf{c}, \mathbf{b}$ is combined with an $(s+1)$-stage ERK method with coefficients $\hat{\mathbf{A}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}$. As is conventional, we assume that $\hat{\mathbf{c}} = (0,\ \mathbf{c})^T$, and the IRK method is taken to be an SDIRK method [22], [30]. An SDIRK method has a lower triangular $\mathbf{A}$ with all diagonal elements a_{ii} nonzero and equal. This form of $\mathbf{A}$ creates efficiencies in the solution to the nonlinear equations for the stages at each step, i.e., allowing each stage to be solved for sequentially with the same Newton iteration matrix.

In this paper, we consider the IMEX-RK methods, ARK3(2)4L[2]SA and ARK5(3)8L[2]SA, from [22], which we denote by ARK3 and ARK5, respectively. ARK3 is an IMEX-RK method having four stages and order 3 with an embedded method of order 2 for error estimation and automatic step-size control; ARK5 has eight stages, order 5, and an embedded method of order 4. The Butcher tableaux of ARK3 and ARK5 are listed in Appendix C of [22].

We split the ODE (4) on each time step $[t_{n-1}, t_n]$ with the (default) dynamic splitting in `odeToJava`, namely, $\mathbf{f}_I(t, \mathbf{y}) := \mathbf{J}(t_{n-1}, \mathbf{y}_{n-1})\mathbf{y}(t)$ and $\mathbf{f}_E(t, \mathbf{y}) := \mathbf{f}(t, \mathbf{y}) - \mathbf{J}(t_{n-1}, \mathbf{y}_{n-1})\mathbf{y}(t)$, where $\mathbf{J} := \partial\mathbf{f}/\partial\mathbf{y}$. We note that this splitting is such that only the linear term is treated implicitly, and hence, there is no need for a Newton iteration when solving the implicit equations. This makes the method similar to *Rosenbrock methods*, see, e.g., [8], which are also linearly implicit methods but for ODEs (4) that are not split.

E. Rush–Larsen Method

An alternative to using classical methods such as FE that is popular in the cardiac simulation literature is the *Rush–Larsen* (RL) method. The RL method advances the solution to the gating equations (2) using

$$y_n = y_\infty + (y_{n-1} - y_\infty)e^{-\Delta t_n/\tau_y}$$

which represents the exact solution of (2) *assuming that all variables besides y are constant*. FE is then used to advance the solution of the remaining equations. Using this method, the Luo–Rudy model, for example, is no longer stiff [31]; i.e., the time step size can be chosen based on accuracy considerations. However, this method is only first-order accurate, and thus, suffers from the usual drawbacks of low-order methods.

F. Accuracy and Stability

An IVP is often called *stiff* if the choice of step-size Δt_n of a numerical method is determined by stability requirements rather than by accuracy requirements.[2] Generally, the step-size required for a stiff problem is much smaller than accuracy requirements dictate. In such cases, the numerical solution is typically much more accurate than required by the user.

When a numerical method is able to produce a stable approximation, we are then interested in the accuracy of the approximation. When the exact solution is not known, we may be able to generate a *reference solution* by using a variable step-size solver with low error tolerances until two approximations are produced that agree to a desired number of significant digits. In this study, we generate a reference solution by using a high-order, variable step-size implicit solver, and lowering the error tolerances for successive approximations until two approximations are identical for at least ten significant digits at N equally spaced output points $t_i = it_f/N$, $i = 1, 2, \ldots, N$,

[2]Arguably, there is no universally accepted definition of a stiff problem, but this description of stiffness suffices for our study.

TABLE I
INITIAL VALUES FOR $V_m = V_{\text{rest}}$

Model	V_{rest}
Luo–Rudy	-35.0
Courtemanche et al.	-81.2
Winslow et al.	-35.0
Puglisi–Bers	-85.5

with $N = 100$. We can then measure the error in the approximation, $\mathbf{y}$, relative to the reference solution, $\hat{\mathbf{y}}$. A popular way to quantify error in the literature on heart simulation is the *relative root mean squared (rrms) error* of the transmembrane potential [32]

$$\text{rrms} := \sqrt{\frac{\sum_{i=1}^{N}(V_{m,i} - \hat{V}_{m,i})^2}{\sum_{i=1}^{N}\hat{V}_{m,i}^2}}$$

where $V_{m,i}$ is the numerical approximation and $\hat{V}_{m,i}$ is the reference solution at time t_i as described earlier. Given the many approximations made in creating the model, an rrms error of 5% is generally considered acceptable.

As a more familiar measure of error, we also quantify error via the *global error*, which we define as

$$e_{\text{global}} := \max_i |V_{m,i} - \hat{V}_{m,i}|, \qquad i = 1, 2, \ldots, N.$$

IV. RESULTS

Using `odeToJava`, we performed numerical experiments with the FE, ERK4, RL, DP, BE, SDIRK4, RADAU5, ARK3, and ARK5 methods and compared their performance. Except for V_m, the initial values of the variables were taken to be those that correspond to the heart in its resting state. For the Luo–Rudy and Winslow *et al.* models, the initial values of V_m were chosen to produce the effect of an explicit stimulus current. That is, the initial values used for V_m are above the thresholds for the cells to fire. For the Courtemanche *et al.* and Puglisi–Bers models, the initial values for V_m were taken as their resting values, and an explicit stimulus was applied as given by (1). In particular, for the Courtemanche *et al.* model, $I_{st} = -2000$ pA/pF from $t = 0$ to $t = 2$ ms, and 0 elsewhere; for the Puglisi–Bers model, $I_{st} = -10$ μA/μF from $t = 1$ to $t = 5$ ms, and 0 elsewhere. The specific initial values used for V_m are listed in Table I. See [18]–[21] for complete listings of the remaining initial values. These values may also be accessed at `http://www.cellml.org/models/` under the directories `luo_rudy_1991_version04`, `courtemanche_ramirez_nattel_1998_version02`, `winslow_rice_jafri_marban_ororke_1999_version01`, and `puglisi_bers_2001_version01`, respectively.

The models were solved over time intervals representing one cardiac cycle. Different time intervals were used due to specific physiological properties of the mammalian heart that each model represents. Accordingly, the Luo–Rudy model was solved on the interval [0,450] ms; the Courtemanche *et al.* model was solved on the interval [0,500] ms; the Winslow *et al.* model was solved on the interval [0,300] ms; and the Puglisi–Bers model was solved on the interval [0,330] ms.

All the numerical experiments were performed on an Athlon 64 3000+ 1.8 GHz processor with 1 GB RAM. CPU times reported are the minimum of five runs. We note that the runs for FE, ERK4, RL, DP, ARK3, and ARK5 were performed within the `odeToJava` framework. The runs for BE were performed using `ode15s` within Matlab with `MaxOrder` set to 1 and `BDF` to "`On`." To ensure a meaningful comparison, a conversion factor was determined to compare CPU time within Matlab to CPU time within `odeToJava`. This was done by computing the average of the ratios of a number of runs of `ode45` with the DP class within `odeToJava` for different tolerances. Similarly, the SDIRK4 and RADAU5 methods were run using their respective Fortran codes, and a conversion factor for CPU times was calculated by comparing runs of `DOPRI5.f` [33, p. 477] with the DP class in `odeToJava`. All CPU times reported later for BE, SDIRK4, and RADAU5 reflect this conversion.

A. Customized Linear System Solver

With the splitting employed, a linear system involving $\mathbf{J}(t, \mathbf{y})$ must be solved at each time step of an IMEX-RK integration. Accordingly, we are interested in the *sparsity pattern* of $\mathbf{J}(t, \mathbf{y})$ across the entire solution interval of an ODE. A sparsity pattern can be thought of as a map of a matrix, describing which entries of a matrix are always zero and which entries can be nonzero. The sparsity patterns of $\mathbf{J}(t, \mathbf{y})$ for each of the four models were generated; see Fig. 2 for the sparsity pattern of the Luo–Rudy model. If an element of $\mathbf{J}(t, \mathbf{y})$ is always zero, it may be possible to omit it during Gaussian Elimination [34], the method used for solving linear systems by our implicit solver. This means that we may further optimize the IMEX-RK results as compared to the previous section by customizing a Gaussian elimination code to take advantage of these sparsity patterns. Results of the ARK3 and ARK5 methods with customized Gaussian elimination routine are included later. In general, not only do we see that such a customization results in a performance improvement for each of the four models over a wide range of tolerances, but we also see that the customization results in a variable step-size implementation of either ARK3 or ARK5 being the most efficient numerical method for every model.

B. Constant Step-Size Tests

In Table II, we report the maximum step-size, $\Delta t_{\max}$, to three significant figures, for which FE, ERK4, and RL produce an approximation with less than 5% rrms error. We also report the corresponding CPU times required, the rrms error, and the global error.

For FE and ERK4, $\Delta t_{\max}$ is also the step-size that produces a stable solution, indicating that these methods generally view the problems as stiff. Thus, the resulting rrms errors can be well below the desired levels. We also see that FE takes approximately two–three times less CPU time than ERK4 on all the four models studied. Hence, as is well known, higher order does not lead to greater efficiency when nonstiff methods are applied to stiff problems with moderate accuracy requirements.

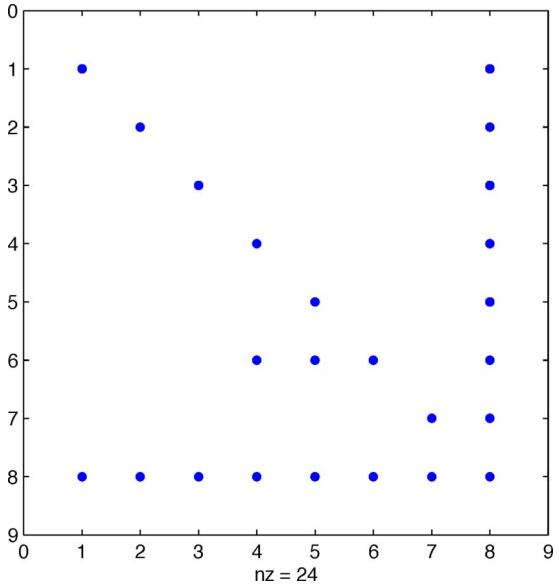

Fig. 2. Sparsity pattern for the Jacobian of the Luo–Rudy model.

TABLE II
RESULTS FOR THE FE, ERK4, AND RL METHODS

		$\Delta t_{\max}$	CPU Time	rrms Error	Global Error
L	FE	1.34E–2	2.24E–1	3.08E–2	2.81E+0
R	ERK4	1.86E–2	6.77E–1	3.87E–2	3.51E+0
	RL	2.50E–1	4.29E–2	4.79E–2	5.39E+0
C	FE	1.94E–2	8.05E–1	2.30E–3	1.94E+0
R	ERK4	2.68E–2	2.30E+0	4.73E–2	6.73E+0
T	RL	3.45E–1	7.89E–2	4.97E–2	3.75E+1
W	FE	1.07E–4	4.04E+1	7.78E–8	9.91E–2
I	ERK4	1.30E–4	1.31E+2	3.40E–2	2.61E–1
N	RL	2.80E–4	2.25E+1	4.86E–2	6.08E+0
P	FE	1.08E–2	4.54E–1	5.20E–3	3.54E–1
B	ERK4	1.48E–2	1.08E+0	1.30E–2	1.33E–1
	RL	4.30E–1	6.50E–2	4.83E–2	8.54E+0

We note that care must be exercised when determining $\Delta t_{\max}$ for the RL method because the rrms error produced is not a monotonically increasing function of the step size. In other words, there exists $\Delta t < \Delta t_{\max}$ for which the rrms error *exceeds* 5%. We also see that RL always takes less CPU time than FE, ranging from approximately two times less on the Winslow *et al*. model to approximately five–ten times less on the other three models.

C. Variable Step-Size Tests

Tables III and VI, respectively, report the results from DP, BE, SDIRK4, RADAU5, ARK3, and ARK5 with variable step-sizes applied to each of the four cardiac electrophysiological models. We run the models using standard error estimation and step-size control algorithms (see, e.g., [28]) for a range of absolute and relative tolerances. We set absolute tolerances and equal to relative tolerances, and define TOL to be their logarithm to base 10; e.g., TOL$= -3$ implies both absolute and relative tolerances were set to 10^{-3}. Integer values of TOL were run from -1 to -6 for all solvers and all the four models, but details are reported for only the runs with the best CPU time that met the 5% rrms error criterion.

For the Luo–Rudy (LR) model, ARK3 and ARK5 produce acceptable solutions in the least amount of CPU time. These methods are able to outperform the other methods in the study even without taking sparsity into account. In this case, ARK3 with sparsity is the most efficient method. It is also over 60% faster than RL, its next closest commonly used competitor.

For the Courtemanche *et al*. (CRT) model, ARK3 produces an acceptable solution in the least amount of CPU time. With sparsity, ARK3 is over 40% faster than RL, its next closest commonly used competitor.

For the Winslow *et al*. (WIN) model, ARK3 again produces an acceptable solution in the least amount of CPU time. With sparsity, ARK3 is over 25% faster than SDIRK4, its next closest commonly used competitor.

Finally, for the Puglisi–Bers (PB) model, we see that ARK3 and ARK5 produce acceptable solutions in the least amount of CPU time, with or without taking sparsity into account. In this case, ARK5 with sparsity produces an acceptable result almost ten times faster than RL, its next closest commonly used competitor.

D. Constant Step-Size IMEX

We also investigated the use of constant step-size ARK3 and ARK5 methods. We find that these constant step-size implementations significantly underperform the corresponding variable step-size implementations. They also significantly underperform the RL method on all of the models studied. We omit further details.

V. CONCLUSION

In this paper, we compared the performance of several numerical methods for approximating solutions to ODEs found in four popular mathematical models of cardiac electrical activity. In particular, we compared the performance two IMEX-RK methods (ARK3 and ARK5) to other commonly used numerical methods for these models, i.e., FE, ERK4, RL, DP, SDIRK4, and RADAU5.

For constant step sizes, the RL method is the most efficient of all the models studied here. It ranges from being approximately two times faster than the FE method for the Winslow *et al*. model to approximately five–ten times faster than FE for the other models.

For variable step sizes, the ARK methods outperformed the Dormand–Prince method for all the four models. We obtained qualitatively similar results from a comparison of the ARK methods with the Bogacki–Shampine 3(2) method, which is the underlying method behind Matlab's `ode23` routine; we do not comment on this further.

Overall, a variable step-size implementation of ARK3 or ARK5 with a customized linear system solver was the most efficient numerical method for all the four mathematical models of cardiac electrical activity considered here. The results in this paper indicate that it is generally advisable to use a numerical method with an inexpensive implicit component and variable

TABLE III
RESULTS FOR THE LUO–RUDY MODEL

	TOL	CPU Time (s)	Sparse CPU Time	rrms Error	Global Error	Average Δt	Max Δt	Min Δt
DP	–2	5.20E–1	–	1.85E–2	2.41E+0	9.26E–2	4.99E–1	7.80E–3
BE	–3	1.45E–1	–	1.61E–2	2.92E+0	1.08E+0	5.89E+0	1.85E–3
SDIRK4	–1	4.80E–2	–	1.77E–2	2.05E+0	1.25E+1	6.78E+1	1.00E–3
RADAU5	–1	7.30E–2	–	6.75E–4	1.78E–2	1.25E+1	9.58E+1	1.00E–3
ARK3	–2	4.00E–2	1.80E–2	5.33E–3	1.91E+0	7.25E+0	6.42E+1	1.00E–3
ARK5	–2	3.40E–2	2.40E–2	7.46E–4	2.42E+0	1.15E+1	4.85E+1	6.36E–3

TABLE IV
RESULTS FOR THE COURTEMANCHE *ET AL.* MODEL

	TOL	CPU Time (s)	Sparse CPU Time	rrms Error	Global Error	Average Δt	Max Δt	Min Δt
DP	–3	6.37E–1	–	5.26E–4	2.71E–1	1.66E–1	4.98E–1	2.84E–3
BE	–2	8.26E–2	–	2.40E–2	1.21E+1	2.34E+0	1.77E+1	2.82E–3
SDIRK4	–2	1.28E–1	–	7.51E–4	1.91E–1	9.43E+0	7.91E+1	1.00E–3
RADAU5	–1	2.40E–1	–	2.61E–3	2.79E+0	1.04E+1	5.00E+1	1.00E–3
ARK3	–2	6.90E–2	4.50E–2	7.87E–3	1.06E+0	7.14E+0	3.32E+1	1.00E–3
ARK5	–2	1.37E–1	6.00E–2	3.60E–2	2.28E+1	1.19E+1	5.43E+1	1.00E–3

TABLE V
RESULTS FOR THE WINSLOW *ET AL.* MODEL

	TOL	CPU Time (s)	Sparse CPU Time	rrms Error	Global Error	Average Δt	Max Δt	Min Δt
DP	–3	3.00E+1	–	7.92E–3	4.15E–1	4.88E–3	6.08E-1	1.57E–4
BE	–4	9.51E–1	–	3.71E–2	1.25E+1	1.32E–1	5.38E+0	8.58E–5
SDIRK4	–2	2.72E–1	–	5.03E–3	5.81E–1	3.75E+0	4.79E+1	1.00E–3
RADAU5	–3	6.13E–1	–	7.68E–3	9.84E–1	2.65E+0	2.70E+1	1.00E–3
ARK3	–3	2.68E–1	1.97E–1	1.46E–2	1.84E+0	1.69E+0	3.45E+1	1.00E–3
ARK5	–3	4.65E–1	2.92E–1	3.15E–2	3.96E+0	3.06E+0	3.45E+1	1.00E-4

TABLE VI
RESULTS FOR THE PUGLISI–BERS MODEL

	TOL	CPU Time (s)	Sparse CPU Time	rrms Error	Global Error	Average Δt	Max Δt	Min Δt
DP	–2	1.53E+0	–	4.43E–2	1.30E+0	4.09E–2	4.94E–1	4.71E–3
BE	–3	1.74E–1	–	5.69E–3	7.62E+0	6.13E–1	1.01E+1	1.76E–1
SDIRK4	–3	1.36E–1	–	8.48E–3	5.85E+0	4.92E+0	4.45E+1	1.00E–3
RADAU5	–2	1.53E–1	–	4.77E–2	5.53E+1	6.34E+0	7.18E+1	1.00E–3
ARK3	–4	6.90E–2	4.60E–2	2.38E–3	5.04E–1	4.12E+0	5.50E+1	1.00E–3
ARK5	–3	1.12E–2	7.00E–3	1.89E–2	1.63E+1	1.48E+0	7.20E+1	1.00E–4

step sizes and specialized techniques that take advantage of specific problem structure. When variable step the sizes are not possible, the RL method is the most efficient method.

It may be possible to further increase the efficiency of IMEX-RK methods over the methods studied here by modifying the definitions of $\mathbf{f}_E$ and $\mathbf{f}_I$ in (5) from the default ones in `odeToJava` or by constructing new IMEX-RK methods.

In this paper, we have only considered ODE models of one cell. Further investigations of models involving large numbers of cells coupled with PDEs in 2- or 3-D are necessary to fully establish the potential of the results presented here. In particular, linearly implicit IMEX operator-splitting methods that do not use constant step sizes may lead to substantial performance gains. We report elsewhere on research efforts in these directions.

ACKNOWLEDGMENT

The authors would like to thank M. MacLachlan for reference Matlab code for the Luo–Rudy, Courtemanche *et al*., and Winslow *et al*. models, and S. Flaim for reference Matlab code for the Puglisi–Bers model. They also thank the reviewers for their comments and R. Wang for his help with some of the Fortran codes.

REFERENCES

[1] J. A. DiMasi, R. W. Hansen, and H. G. Grabowski, "The price of innovation: New estimates of drug development costs," *J. Health Econ.*, vol. 22, no. 2, pp. 151–185, 2003.

[2] A. Hodgkin and A. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol. (Lond.)*, vol. 117, no. 4, pp. 500–544, 1952.

[3] K. Skouibine and W. Krassowska, "Increasing the computational efficiency of a bidomain model of defibrillation using a time-dependent activating function," *Ann. Biomed. Eng.*, vol. 28, no. 7, pp. 772–780, 2000.

[4] K. H. W. J. ten Tusscher and A. V. Panfilov, "Cell model for efficient simulation of wave propagation in human ventricular tissue under normal and pathological conditions," *Phys. Med. Biol.*, vol. 51, no. 23, pp. 6141–6156, 2006.

[5] L. K. N. Virag and J. M. Vesin, "A computer model of cardiac electrical activity for the simulation of arrhythmias," *Pacing Clin. Electrophysiol.*, vol. 21, no. 11, pp. 2366–2371, 1998.

[6] J. Sundnes, B. F. Nielsen, K.-A. Mardal, X. Cai, G. T. Lines, and A. Tveito, "On the computational complexity of the bidomain and the monodomain models of electrophysiology," *Ann. Biomed. Eng.*, vol. 34, no. 7, pp. 1088–1097, 2006.
[7] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations* (Cambridge Texts in Applied Mathematics Series). Cambridge, U.K.: Cambridge Univ. Press, 1996.
[8] W. Hundsdorfer and J. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations* (Springer Series in Computational Mathematics Series). vol. 33, Berlin, Germany: Springer-Verlag, 2003.
[9] Z. Qu and A. Garfinkel, "An advanced algorithm for solving partial differential equation in cardiac conduction," *IEEE Trans. Biomed. Eng.*, vol. 46, no. 9, pp. 1166–1168, Sep. 1999.
[10] S. Rush and H. Larsen, "A practical algorithm for solving dynamic membrane equations," *IEEE Trans. Biomed. Eng.*, vol. BME-25, no. 4, pp. 389–392, Jul. 1978.
[11] Y. Han and K. Ng, "Implicit-explicit methods for nonlinear bidomain modeling," *Ann. Biomed. Eng.*, vol. 28, Suppl. 1, pp. 34–34, 2000.
[12] J. Keener and K. Bogar, "A numerical method for the solution of the bidomain equations in cardiac tissue," *Chaos*, vol. 8, no. 1, pp. 234–241, 1998.
[13] J. Whiteley, "An efficient numerical technique for the solution of the monodomain and bidomain equations," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 11, pp. 2139–2147, Nov. 2006.
[14] J. Sundnes, G. Lines, K.-A. Mardal, and A. Tveito, "Multigrid block preconditioning for a coupled system of partial differential equations modeling the electrical activity in the heart," *Comput. Methods. Biomech. Biomed. Engin.*, vol. 5, no. 6, pp. 397–409, 2002.
[15] J. Sundnes, G. T. Lines, and A. Tveito, "An operator splitting method for solving the bidomain equations coupled to a volume conductor model for the torso," *Math. Biosci.*, vol. 194, no. 2, pp. 233–248, 2005.
[16] Y. Zhou, P. Jung, and C. Zhu, "Simulation of transmembrane potential propagation in three-dimensional bidomain cardiac tissue," in *Proc. 1st Int. Multi-Symp. Comput. Comput. Sci. (IMSCCS'06)*, vol. 1, Washington, DC: IEEE Computer Society, 2006, pp. 266–273.
[17] J. Sundnes, G. Lines, and A. Tveito, "ODE solvers for a stiff system arising in the modeling of the electrical activity of the heart," *Int. J. Nonlinear Sci.*, vol. 4, no. 1, pp. 67–80, 2002.
[18] C. Luo and Y. Rudy, "A model of ventricular cardiac action potential," *Circ. Res.*, vol. 68, no. 6, pp. 1501–1526, 1991.
[19] M. Courtemanche, R. J. Ramirez, and S. Nattel, "Ionic mechanisms underlying human atrial action potential properties: Insights from a mathematical model," *Amer. J. Physiol.*, vol. 275, no. 1, pp. H301–H321, 1998.
[20] R. L. Winslow, J. Rice, S. Jafri, E. Marbán, and B. O. Rourke, "Mechanisms of altered excitation-contraction coupling in canine tachycardia-induced heart failure, ii model studies," *Circ. Res.*, vol. 84, no. 5, pp. 571–586, 1999.
[21] J. L. Puglisi and D. M. Bers, "Labheart: An interactive computer model of rabbit ventricular myocyte ion channels and ca transport," *Amer. J. Physiol. Cell Physiol.*, vol. 281, no. 6, pp. C2049–C2060, 2001.
[22] C. A. Kennedy and M. H. Carpenter, "Additive Runge–Kutta schemes for convection-diffusion-reaction equations," *Appl. Numer. Math.*, vol. 44, no. 1–2, pp. 139–181, 2003.
[23] J. Sundnes, G. T. Lines, X. Cai, B. F. Nielsen, K.-A. Mardal, and A. Tveito, *Computing the Electrical Activity in the Heart*. New York: Springer-Verlag, 2006.
[24] G. Beeler and H. Reuter, "Reconstruction of the action potential of ventricular myocardial fibers," *J. Physiol. (Lond.)*, vol. 268, no. 1, pp. 177–210, 1977.
[25] C. Luo and Y. Rudy, "A dynamic model of the cardiac ventricular action potential. i. Simulations of ionic currents and concentration changes," *Circ. Res.*, vol. 74, no. 6, pp. 1071–1096, 1994.
[26] C. Luo and Y. Rudy, "A dynamic model of the cardiac ventricular action potential. ii. Afterdepolarizations, triggered activity, and potentiation," *Circ. Res.*, vol. 74, no. 6, pp. 1097–1113, 1994.
[27] "http://www.netlib.org/ode/odeToJava.tgz." [Online]. Available: http://www.netlib.org/ode/odeToJava.tgz
[28] L. F. Shampine, I. Gladwell, and S. Thompson, *Solving ODEs With MATLAB*. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[29] E. Hairer and G. Wanner, *Solving Ordinary Differential equations. II* (Springer Series in Computational Mathematics Series). vol. 14, Berlin, Germany: Springer-Verlag, 1991.
[30] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri, "Implicit-explicit Runge–Kutta methods for time-dependent partial differential equations," *Appl. Numer. Math.*, vol. 25, no. 2–3, pp. 151–167, 1997.
[31] R. J. Spiteri and M. C. MacLachlan, "An efficient non-standard finite difference scheme for an ionic model of cardiac action potentials," *J. Difference Equ. Appl.*, vol. 9, no. 12, pp. 1069–1081, 2003 (dedicated to Prof. R. E. Mickens on the occasion of his 60th birthday).
[32] B. Horacek, J. Warren, P. Stovicek, and C. Feldman, "Diagnostic accuracy of derived versus standard 12-lead electrocardiograms," *J. Electrocardio.*, vol. 33, pp. 155–160, 2000.
[33] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations. I* (Springer Series in Computational Mathematics Series). vol. 8, Berlin, Germany: Springer-Verlag, 1987.
[34] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods For Sparse Matrices* (Monographs on Numerical Analysis Series). 2nd ed., New York: Clarendon Press Oxford Univ. Press, 1989, Oxford Science Publications.

Raymond J. Spiteri received the Ph.D. degree in mathematics from the University of British Columbia, Vancouver, BC, Canada, in 1997.

He is currently an Associate Professor in the Department of Computer Science, University of Saskatchewan, Saskatchewan, Canada. His current research interests include numerical analysis, scientific computation, and problem-solving software environments for numerical time integration methods.

Ryan C. Dean is currently working toward the M.Sc. degree from the Department of Computer Science, University of Saskatchewan, Saskatchewan, Canada.

His current research interests include efficient numerical methods for simulation of electrical activity in human cardiac tissue.