Received: 18 November 2021 | Revised: 16 November 2022 | Accepted: 21 December 2022

DOI: 10.1002/cnm.3677

BASIC RESEARCH

WILEY

An inner-outer subcycling algorithm for parallel cardiac electrophysiology simulations

Sebastian Laudenschlager[1] | **Xiao-Chuan Cai**[2]

[1]Department of Computer Science, University of Colorado Boulder, Boulder, Colorado, USA

[2]Department of Mathematics, University of Macau, Macau, China

Correspondence
Xiao-Chuan Cai, Department of Mathematics, University of Macau, Macau, China
Email: xccai@um.edu.mo, cai@colorado.edu

Funding information
National Science Foundation, Grant/Award Numbers: ACI-1532236, ACI-1532235

Abstract

This paper explores cardiac electrophysiological simulations of the monodomain equations and introduces a novel subcycling time integration algorithm to exploit the structure of the ionic model. The aim of this work is to improve upon the efficiency of parallel cardiac monodomain simulations by using our subcycling algorithm in the computation of the ionic model to handle the local sharp changes of the solution. This will reduce the turnaround time for the simulation of basic cardiac electrical function on both idealized and patient-specific geometry. Numerical experiments show that the proposed approach is accurate and also has close to linear parallel scalability on a computer with more than 1000 processor cores. Ultimately, the reduction in simulation time can be beneficial in clinical applications, where multiple simulations are often required to tune a model to match clinical measurements.

KEYWORDS

cardiac electrophysiology, finite element on unstructured meshes, parallel processing, patient-specific cardiac geometry, time integration with subcycling

1 | INTRODUCTION

The prevalence of cardiovascular disease[1] gives cardiac research a sense of great importance, because advancements in cardiac research could ultimately lead to a reduction in deaths related to cardiovascular disease. One crucial area of cardiac research is cardiac electrophysiology, which investigates the electrical propagation that drives a heartbeat in the human heart. In a clinical setting, this can be studied, for example, with magnetic resonance imaging (MRI) and electrocardiograms (ECG) in order to monitor and detect irregularities in the electrical activity. A different approach is to simulate the electrical activity in the heart using cardiac models.[2–5] Cardiac electrophysiology simulations at the tissue level can often be modeled by either the bidomain or the monodomain equations, and incorporate a cellular model to deal with the intracellular currents. In this paper, we utilize the monodomain equations. This is a tradeoff between complexity and accuracy, as the bidomain equations more accurately describe cardiac electrical activity, but are also more expensive to solve computationally.[6,7]

The monodomain equations are solved via the finite element method (FEM) on a variety of meshes, ranging from simple slabs for benchmarking to meshes generated from actual MRI data. In order to generate patient-specific meshes from MRI data, we developed a semi-automatic mesh generation pipeline, which utilizes user-guided segmentation and automatic model creation from segmentation. This pipeline uses a variety of open source software, and allows for flexibility in mesh generation, including mesh size and quality.

With the semi-implicit discretization, a large sparse linear system of equations needs to be solved at every time step, which is the most time consuming part of the simulation. We solve the system with a conjugate gradient method preconditioned by a parallel algebraic multigrid. The actual software is a modified version of the FEM variant of the Cardioid[8] package. It uses the FEM library MFEM[9] and the solver/preconditioning library Hypre.[10] Using MFEM allows us to abstract the details of FEM, and instead focus on the higher level details of the problem. Furthermore, the Hypre library gives us access to various high performance solvers and preconditioners, giving us flexibility to test various parameters to find the optimal combination for our problem. In addition to MFEM and Hypre, a variety of Python scripts were developed for post-processing simulation data, as well as for visualization and data analysis.

An important aspect of our solver is parallelization, meaning simply that we can use more processors in parallel to solve the problem much more efficiently. This is crucial because it reduces the turnaround time for cardiac electrophysiology simulations, which typically require very fine resolutions,[11,12] thus allowing more simulations to be run in a shorter time frame. Many strategies and algorithms have been explored in parallel cardiac electrophysiology simulations to improve the efficiency of solving the bidomain or monodomain equations,[13–15] including techniques like parallel preconditioning[16,17] and fully implicit solvers.[18] The parallelization used for our work shares many aspects of existing parallel cardiac electrophysiology codes such as Chaste,[19] e.g. using a parallel FEM solver with preconditioning. We tested the parallel scalability of the Cardioid code on both simplified and realistic geometries, and found excellent scalability up to 1000 processors, similar to existing approaches.[20,21] Beyond the scalability of the code, we also performed a verification study of our simulation results. To that end, we compared activation times with those of the Niederer benchmark[22] across a variety of mesh sizes and timesteps. The results indicate good spatial and temporal convergence, and compare well with other implementations of monodomain solvers.

Due to the inherent multiscale nature of cardiac electrophysiology simulations, numerous previous studies have been conducted to improve the computational efficiency of simulations through spatial and temporal adaptivity. Spatial adaptivity through Adaptive Mesh Refinement (AMR) has been used on both the bidomain[23] and monodomain equations,[24,25] with modest observed speedups as a result of balancing the speedup with the cost of performing the mesh refinement. Temporal adaptivity has also been studied in the context of cardiac electrophysiology simulations, including simple global timestep adaptivity,[26] applying full adaptivity to both the reactive and diffusive terms,[27] and even uniformization methods based on Markov chains.[28,29] These time adaptive methods are globally applied, meaning a residual estimator is needed to determine when to refine the timestep and how much the refinement should be.

In this paper, we instead focus on locally refining the timestep based on the underlying dynamics of ionic model, similar to the locality found in spatial adaptivity. Thus, a residual estimator is not necessary since we know in advance that, for the human heart, the solution changes sharply during the depolarization phase. The main goal of this paper is to improve the turnaround time for a simulation by introducing a novel subcycling algorithm for timestepping in the ionic model. Instead of using the existing timestepping method, where a global fixed timestep is used for both the nonlinear parabolic partial differential equation (PDE) for the electrical conduction and the system of ordinary differential equations (ODE) for the cellular ionic currents, we use several iterations of smaller timesteps in the ionic model for the sharp depolarization phase. This allows us to use a larger timestep for the PDE, as well as for the ionic model ODEs during the repolarization phase. For different ionic components, the length of the sharp change periods are slightly different and our selection is based on the longest component. Through the use of subcycling methods, we can accurately utilize timesteps of 0.1 ms, instead of the small timestep sizes of 0.01–0.05 ms used in the benchmark tests.

The rest of this paper is organized as follows. In Section 2, we present the continuous and discretized models for tissue-level electrical propagation and the cellular ionic model, as well as the mesh generation pipeline. In Section 3, we present the subcycling method in the ionic model computation, followed by numerical results and associated discussion in Section 4. Finally, in Section 5, we wrap up with some concluding remarks.

2 | MATHEMATICAL MODELS

The cardiac electrical propagation described by the monodomain equations, based on the pioneering Hodgkin-Huxley model,[30] consists of two main components, the PDEs for electrical conduction and the ODEs for cellular ionic currents. We first present the monodomain equations, including the discretized form using FEM, followed by the ionic model.

2.1 | Monodomain equations

The monodomain equations describing the propagation of transmembrane potential $V(x, t)$ on $\Omega \in \mathbb{R}^3$ and $t \in [0, T]$ are given by

$$\begin{aligned} \chi C_m \left(\frac{\partial V}{\partial t} + I_{\text{ion}}(V) - I_{\text{stim}} \right) &= \nabla \times (\sigma \nabla V) \\ I_{\text{ion}}(V) &= \sum_{i \in \Gamma} I_i(V) \\ (\sigma \nabla V) \cdot \hat{n} &= 0 \quad \text{on} \quad \partial\Omega \\ V(x, 0) &= V_{\text{init}}, \end{aligned} \tag{1}$$

where χ is the surface area to volume ratio, C_m is the cell membrane capacitance, and σ is the conductivity tensor. The I_{ion} term represents the current due to ionic flow through cell membrane channels, while I_{stim} is an externally applied stimulus. The boundary condition given in this formulation is an implicit no-flux condition, while the initial voltage V_{init} is set to -83 mV. Note that Γ here represents the set of different subsystem currents that contribute to I_{ion}.

The ionic model used in this paper is the Ten Tusscher 2006 model,[31] which is commonly used in cardiac electrophysiology studies.[32,33] This model consists of 12 subsystem currents I_i where $i \in \Gamma$, which is given by the set

$$\Gamma = \{\text{Na, K1, to, Kr, Ks, CaL, NaCa, NaK, pCa, pK, bCa, bNa}\}.$$

These currents, in turn, are dependent on a set of 12 gating variables and six non-gating variables. The (independent) gating variables g^i are all given by Hodgkin-Huxley type ODEs of the form

$$\frac{dg^i}{dt} = \frac{g^i_\infty - g^i}{\tau_{g^i}} \quad \text{for } i \in G, \tag{2}$$

where G is the set of gating variables. Here g^i_∞ and τ_{g^i} are exponential functions of V. The non-gating variables, on the other hand, are not independent. They satisfy the following ODEs:

$$\begin{aligned} \frac{d\text{Na}}{dt} &= -\frac{I_{\text{Na}} + I_{\text{bNa}} + 3I_{\text{NaK}} + 3I_{\text{NaCa}}}{V_c F} \\ \frac{d\text{Ca}}{dt} &= \text{Ca}_{ibufc} \left[\frac{V_{\text{sr}}}{V_c} \left(I_{\text{leak}} - I_{\text{up}} \right) + I_{\text{xfer}} - \frac{I_{\text{bCa}} + I_{\text{pCa}} - 2I_{\text{NaCa}}}{2V_c F} \right] \\ \frac{d\text{Ca}_{\text{sr}}}{dt} &= \text{Ca}_{\text{srbufsr}} \left(I_{\text{up}} - I_{\text{leak}} - I_{\text{rel}} \right) \\ \frac{d\text{Ca}_{\text{ss}}}{dt} &= \text{Ca}_{\text{ssbufss}} \left(-\frac{I_{\text{CaL}}}{2V_{\text{ss}} F} + \frac{V_{\text{sr}}}{V_{\text{ss}}} I_{\text{rel}} - \frac{V_c}{V_{\text{ss}}} I_{\text{xfer}} \right) \\ \frac{dK}{dt} &= -\frac{I_{K1} + I_{\text{to}} + I_{\text{Kr}} + I_{\text{Ks}} - 2I_{\text{NaK}} + I_{\text{pK}} + I_{\text{stim}}}{V_c F} \\ \frac{d\overline{R}}{dt} &= -k_2 \text{Ca}_{\text{ss}} \overline{R} + k_4 \left(1 - \overline{R} \right). \end{aligned} \tag{3}$$

The right side of the equations in (3) are dependent on a variety of ionic currents and gating variables. We can write the non-gating ODEs more generally as

$$\frac{dw^j}{dt} = F^j(w, g, V) \quad \text{for} \quad j \in \text{NG}, \tag{4}$$

where NG is the set of non-gating variables, and F^j represents the right-hand side of the jth non-gating variable ODE.

The complexity of the ionic model stems from the sharp spikes found in the ionic model variables. In particular, they can be attributed to the gating variables, which can change rapidly during the depolarization phase. This, in turn, induces a rapid change in the ionic currents, which depend on the gating variables. For example, the fast sodium current I_{Na}, which depends on three gates, and the transient outward current I_{to}, which depends on two gates, both undergo a rapid change over a small time period, as shown in Figure 1. As we shall see later, these rapid changes limit the timestep sizes that can be used for the numerical solution of the non-gating variables, as they depend on these rapidly changing quantities.

2.2 | Spatial and temporal discretization

In order to discretize the PDE component that is the first equation in (1), in time, we will use the second-order implicit Crank-Nicolson[34] method. Using a timestep size of Δt, this yields

$$-\frac{\Delta t}{2\chi C_m}\nabla\cdot(\sigma\nabla V_n)+V_n=\frac{\Delta t}{2\chi C_m}\nabla\cdot(\sigma\nabla V_{n-1})+V_{n-1}+\Delta t I_{\text{total}}, \tag{5}$$

where V_n and V_{n-1} are the approximations to V at timesteps n and $n-1$, respectively, and $I_{\text{total}}=I_{\text{stim}}-I_{\text{ion}}(V_{n-1})$ groups the ionic current terms together, and the use of V_{n-1} in I_{ion} linearizes the problem.

For the spatial discretization, we discretize (5) in space using a finite element method with continuous piecewise linear tetrahedral elements. The resulting linear system of equations can then be written as

$$\left(M+\frac{\Delta tK}{2\chi C_m}\right)V_n^h=\left(M-\frac{\Delta tK}{2\chi C_m}\right)V_{n-1}^h+B_{n-1}, \tag{6}$$

where M and K are the mass and stiffness matrices, respectively, and $\widetilde{B}$ is the vector representing the contribution of the ionic model. Note that the notation V_n^h indicates the finite element approximation to V at timestep n. Thus, (6) is a linear system of equations, to be constructed and solved at every timestep. The coefficients are time-dependent and therefore the matrix changes from timestep to timestep, but the matrix on the left-hand side of (6) is symmetric and positive definite, and ill-conditioned when Δt is large. Moreover, the system becomes more difficult to solve as the geometry of Ω becomes more complex.

2.3 | Mesh generation

For generation of patient-specific cardiac meshes, we developed a new mesh-generation pipeline, which utilizes a mix of existing software and software developed in-house. The pipeline begins with the segmentation process, where

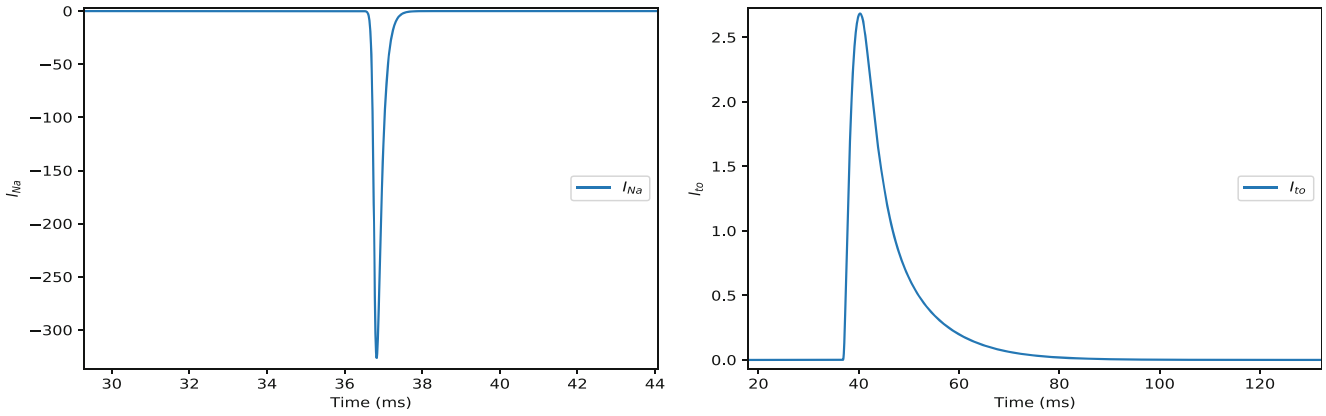

FIGURE 1 Fast sodium (I_{Na}) and transient outward (I_{to}) currents exhibit rapid changes over a small time period

sequences of Digital Imaging and Communications In Medicine (DICOM) images are segmented in a semi-automatic fashion in order to identify the left ventricle (LV), right ventricle (RV), and the epicardium. This process uses Slicer3D, an open source medical image processing software package,[35] and an extension called SlicerIGT.[36] The user selects a set of fiducials for each structure and for each image slice, and the resulting points are automatically interpolated to form a surface. The inter and intra-slice interpolation is achieved using Delaunay alpha triangulation,[37] which is useful over standard Delaunay triangulation because of the large variance in shape and size between structures, especially the RV due to its non-convex, crescent shape. This allows for flexible segmentation of the different cardiac structures of interest. In addition to identifying the three cardiac shapes, we identify another set of fiducials representing a cutting plane (on one image slice only) in order to denote the top of our region of interest. No interpolation is performed for the cutting plane. The whole first stage produces three VTK[38] files (one for each structure), and four fiducial files containing point information for the three structures and the cutting plane. Figure 2 shows sample interpolated deliniations during the segmentation process.

Once we have a surface mesh for each cardiac structure, we need to combine them into a single surface mesh, and cut it at the base of the heart. To that end, we developed a Python script to automatically handle this. It works by computing distance functions upon a regular grid, and using VTK's implicit cutting planes to carve away the non-heart regions. The distance functions are used to first remove everything above the cutting plane, then to remove the interior of the LV and RV, all while interpolating across the boundary of the cut. The effect of this procedure is to stitch together the three models and to hollow out the LV and RV cavities. See Figure 2 for a visualization of the three surfaces before the clipping process, and the resulting surface after.

The surface mesh after the clipping procedure is rather messy because of the simple clipping procedure. An example of this is shown in Figure 3, where the surface mesh on the left contains numerous poorly shaped triangular elements. In order to improve the surface mesh quality, we run the mesh through an open source software package called Approximate Centroidal Voronoi Diagrams (ACVD).[40] This package utilizes an optimized clustering algorithm in order

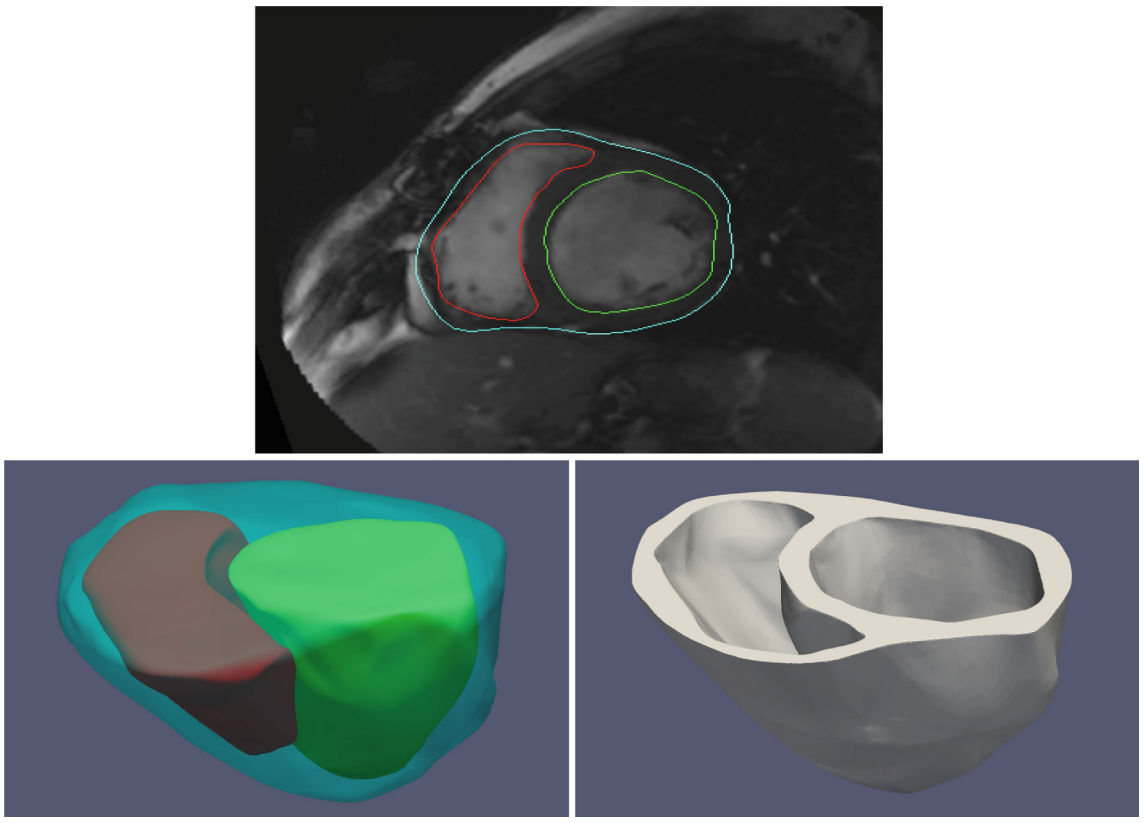

FIGURE 2 Top: The segmentation process via Slicer3D, showing the deliniations for LV (green), RV (red), and Epicardium (cyan) on a single image slice. magnetic resonance imaging data is from the Sunnybrook Cardiac Dataset.[39] Bottom: The three superimposed surfaces (left) before clipping (LV in green, RV in red, Epicardium in cyan), and the combined surface after clipping (right)

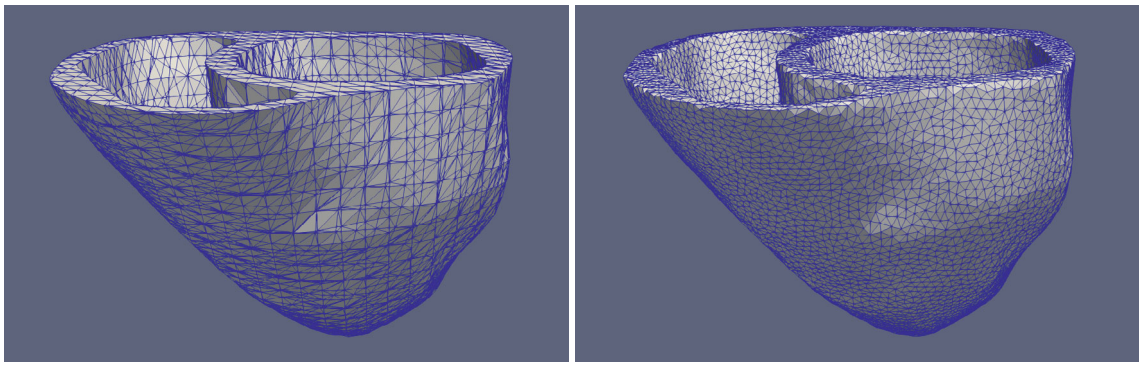

FIGURE 3 Using the Approximate Centroidal Voronoi Diagrams (ACVD) algorithm greatly improves the quality of the clipped surface mesh. Shown on the left is the surface mesh prior to applying ACVD, and shown on the right is the surface mesh after applying ACVD

to produce a higher quality surface mesh. The resulting surface mesh in Figure 3 on the right shows that, after running through ACVD, we get a much higher quality mesh.

The final step in the mesh generation pipeline is to convert the surface mesh into a 3D unstructured tetrahedral mesh. To this end, we utilize TetGen,[41] a versatile tetrahedral mesh generation package using constrained Delaunay tetrahedralizations. TetGen allows us to specify a myriad of mesh quality parameters, such as the radius-edge ratio or the volume constraint of a tetrahedron, in order to create a mesh of arbitrary quality or size. In addition, TetGen can output resulting meshes in several formats, including ones readable by VTK and Medit,[42] providing flexibility for multiple workflows.

2.4 | Solvers

In order to compute the ionic model contribution, we need to solve the gating and non-gating ODEs in (2) and (4). To that end, we use the Rush-Larsen method,[43] which solves the gating variables via an exponential integrator method and the non-gating variables via the Forward Euler method. The Rush-Larsen method is commonly used in cardiac electrophysiology simulations due to the stiffness of the ionic model ODEs,[44] so we use this method as a starting point. Using the Rush-Larsen method, the solution of the gating variable ODEs at timestep n are given by

$$g_n^i = g_{n-1}^i + \left(1 - e^{-\frac{\Delta t}{\tau_{g^i}}}\right)\tau_{g^i}\left(\frac{g_\infty^i - g_{n-1}^i}{\tau_{g^i}}\right), \tag{7}$$

where τ_{g^i} and g_∞^i are functions depending only on V. The solution of the non-gating variable ODEs is more complicated due to the lack of variable independency. Because of this, a single step of the Forward Euler method is used, which then gives the solution at timestep n as

$$w_n^j = w_{n-1}^j + \Delta t F^j\left(w_{n-1}^j, g_{n-1}, V_{n-1}\right). \tag{8}$$

The order in which the ionic model is solved for at a single timestep is: (1) compute the ionic currents and form I_{ion}, (2) update the non-gating variables, and (3) update the gating variables.

Due to the rapidly changing nature of the ionic variables during the depolarization process, we are limited to using a small timestep Δt, usually between 0.005 and 0.05 ms. Larger values of Δt are unable to accurately resolve the sharp spikes in the ionic model variables, which results in a significant delay in activation time.

Once the ionic model contribution is computed, we incorporate it into (6) and use an iterative method to solve the subsequent linear system. Because the system (6) is symmetric and positive definite, we choose a preconditioned conjugate gradient (CG) method[45] as our linear solver, with a relative stopping tolerance of 10^{-6}. The choice of preconditioner is the algebraic multigrid (AMG) method (V-cycle), specifically the BoomerAMG preconditioner in the

Hypre library, which has been shown to perform well in cardiac electrophysiology simulations.[16] The AMG parameters chosen include the l_1 hybrid symmetric Gauss–Seidel smoother (l_1-GS),[46] the Hybrid Modified Independent Set (HMIS) coarsening algorithm,[47] a distance-two modified extended interpolation algorithm,[48] and a strength threshold of $\theta = 0.25$.

3 | AN INNER-OUTER ITERATIVE METHOD WITH SUBCYCLING

The goal of this section is to describe a temporal subcycling algorithm in the computation of the ionic model, which is developed to allow the use of a larger timestep, thus reducing the computational overhead of a cardiac electrophysiology simulation. As described in Section 2.4, the current method of solving the ODE system of the ionic model is applying an exponential integrator to the gating variables, while the non-gating variables are solved with a single step of the Forward Euler method. Our focus will be on improving the performance of the solution of the non-gating variables.

A simple approach to such a performance improvement is to use a larger timestep, thereby requiring fewer timesteps for a given simulation. When using the Forward Euler method for the solution of the non-gating variable ODEs, however, the timestep size is limited because of accuracy and stability concerns. Thus, a global increase to the timestep size is not feasible. Since the outer (PDE) and inner (ODE) portions are solved separately, we can use different timestep sizes for each portion. Due to the outer portion being solved with a Crank–Nicolson method, we can increase the timestep size for the outer portion, while keeping the smaller timestep (with more iterations) for the inner portion. As we show in Section 4.3, we are able to increase the timestep size by a factor of 10, while only incurring a slight increase in cost associated with the computation of the ionic model, and retaining very similar accuracy to a reference solution.

In order to implement this subcycling idea, we developed a simple subcycling algorithm for the Forward Euler method, herafter known as Subcycling Forward Euler (SFE). This algorithm works by introducing a subcycling timestep, Δt_s, which is smaller than the original timestep by some factor $\zeta \in \mathbb{N}$ that is, $\Delta t_s = \frac{\Delta t}{\zeta}$. We then run the Forward Euler scheme a total of ζ times to match the outer PDE timestep. For this SFE algorithm, subcycling takes place over the entirety of the temporal domain. This ends up being unnecessary because of the short timeframe of the rapid depolarization process. Since the voltage is relatively smooth outside of the depolarization, we can optimize this algorithm to apply subcycling only to the timeframe in which the depolarization occurs, and use a globally larger timestep for the rest of the time.

Therefore, we implemented a hybrid subcycling version using Forward Euler (referred to as HFE hereafter), where we use a time threshold t_{thresh} to determine whether or not to apply subcycling. The intuition behind this method is that more accuracy is needed during the depolarization phase (with its associated sharp jumps), while the repolarization phase is smoother (see Section 4.2 for a more detailed discussion). For the HFE method, we use the SFE method for $t < t_{\text{thresh}}$, and a single Forward Euler step (as before) for $t \geq t_{\text{thresh}}$. Additionally, we implemented two more algorithms which utilize a combination of a first-order (SFE) and either a second-order or fourth-order Runge–Kutta method, referred to as HRK2 and HRK4, respectively. Specifically, we retain the SFE method for $t < t_{\text{thresh}}$, then switch to one step of either the second-order or fourth-order Runge Kutta for $t \geq t_{\text{thresh}}$. The subcycling algorithm for a single timestep is summarized in Algorithm 1.

In contrast to fully adaptive timestepping methods, where error estimates are computed at each timestep to possibly adjust the timestep size, our subcycling algorithm uses only a temporal threshold to turn off subcycling for the repolarization phase. Thus, we utilize problem-specific information on the structure of a cardiac electrophysiology cycle to inform the subcycling algorithm. This represents a simpler, more efficient solver for a given ionic model, but at the expense of flexibility, since the subcycling threshold might change based on stimulus patterns and cardiac geometry.

4 | NUMERICAL EXPERIMENTS AND OBSERVATIONS

In order to test the subcycling algorithm, we consider two domains of different shapes and several meshes for each domain. The first domain is a rectangular slab, conforming to the specifications in the Niederer benchmark. This domain is used for verification purposes, by comparing activation times to other cardiac electrophysiology codes at a predefined set of points in the domain. The second domain is a patient-specific heart with the left and right ventricles, and the mesh is generated via the mesh generation pipeline described in Section 2.3.

Algorithm 1: Ionic model subcycling algorithm

algorithm = getAlgorithm() ▷ Possible algorithms are SFE, HFE, HRK2, HRK4
$\zeta = 10$
$\Delta t_i = \Delta t_o$
if algorithm = = SFE **then**
  $\Delta t_i = \frac{\Delta t_o}{\zeta}$
else
  if $t < t_{\text{thresh}}$ **then**
    $\Delta t_i = \frac{\Delta t_o}{\zeta}$
    time integrator = Forward Euler
  else
    $\zeta = 1$
    time integrator = getIntegrator(algorithm) ▷ Possible time integrators are Forward Euler, RK2, RK4
  end if
end if
for ζ iterations **do**
  compute ionic model with time integrator using Δt_i
end for

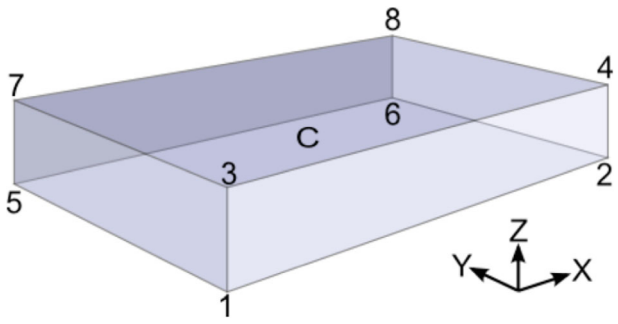

FIGURE 4 Niederer benchmark geometry,[22] where points 1–8 and C are used to measure activation times

4.1 | Experimental results

The first test will be to establish a reference solution to compare against the Niederer benchmark. To that end, we ran the non-subcycling version of the algorithm with a variety of temporal and spatial resolutions, and measured the activation times at a set of predefined points, shown in Figure 4.

Here, we define activation at a point as the time at which the voltage crosses the 0 mV threshold. For all tests on this rectangular geometry, we used a single 1.5 × 1.5 × 1.5 mm stimulus with a 2 ms duration in the corner at P1, as is used in the Niederer benchmark. The results of this test, shown in Table 1, indicate good spatial and temporal convergence, exhibiting only small variations across resolutions. Furthermore, if we plot the activations times for the $\Delta t = 0.01$ ms test across a diagonal line from P1 to P8, we can see in Figure 5 that the results are robust across the three spatial resolutions.

The next test utilizes biventricular patient-specific geometry. Specifically, we generate three meshes of the same geometry but with different resolutions, and compare activation times at a set of predefined points, similar to the Niederer benchmark. We chose a set of nine points, four are located in the RV (P1–P4) and four in the LV (P5–P8),

respectively, with another point near the center of the septum (P9). These points were chosen such that each part of the domain is roughly represented in the activation time analysis. For these tests, we choose a sufficiently small timestep size of $\Delta t = 0.01$ ms so that there are no accuracy concerns attributed to using a larger timestep. Unlike the single stimulus used with the rectangular geometry, we used a seven stimulus protocol for the patient-specific geometry, four in the LV and three in the RV, based on existing data.[49] Each of these seven stimulii has the same strength and duration as those in the Niederer benchmark, the only difference being they are spherical stimulii with a 3 mm radius.

The results in Table 2 show the activation times (in ms) at the aforementioned points for all three meshes. As with the Niederer benchmark, we see little difference in activation times between the three mesh resolutions, indicating good spatial convergence. It should be noted that because of the use of an unstructured grid, the points at which activation times are measured are not exactly the same between the different meshes. Instead, after choosing the reference points, we use the closest possible points relative to the reference points and measure the activation times there.

In order to test the various subcycling algorithms' accuracy against the existing method, we created a set of tests on the same geometry. For these tests, we use the 0.1 mm resolution rectangular mesh from the Niederer benchmark. The reference solution is computed with $\Delta t = 0.01$ ms, while the SFE, HFE, HRK2, and HRK4 methods use $\Delta t = 0.1$ ms

TABLE 1 Activation times for the Niederer benchmark

Δx, Δt	P1	P2	P3	P4	P5	P6	P7	P8	C
0.1, 0.005	1.27	33.53	7.95	34.56	24.28	41.32	25.48	42.14	20.36
0.1, 0.01	1.27	33.88	8.14	34.90	24.52	41.69	25.71	42.51	20.38
0.1, 0.05	1.37	36.70	8.76	37.81	26.48	45.17	27.78	46.06	22.01
0.2, 0.005	1.26	33.09	7.63	33.98	22.65	39.90	23.37	40.70	19.89
0.2, 0.01	1.27	33.40	7.70	34.30	22.83	40.27	23.57	41.07	20.07
0.2, 0.05	1.37	35.91	8.20	36.82	24.29	43.16	25.10	43.95	21.43
0.5, 0.005	1.25	35.68	7.63	36.80	23.03	43.98	25.37	44.89	21.97
0.5, 0.01	1.26	35.87	7367	36.99	23.12	44.18	25.47	45.09	22.07
0.5, 0.05	1.36	37.40	7.99	38.53	23.90	45.79	26.26	46.67	22.85

Note: Spatial data is measured in mm, while temporal data is measured in ms.

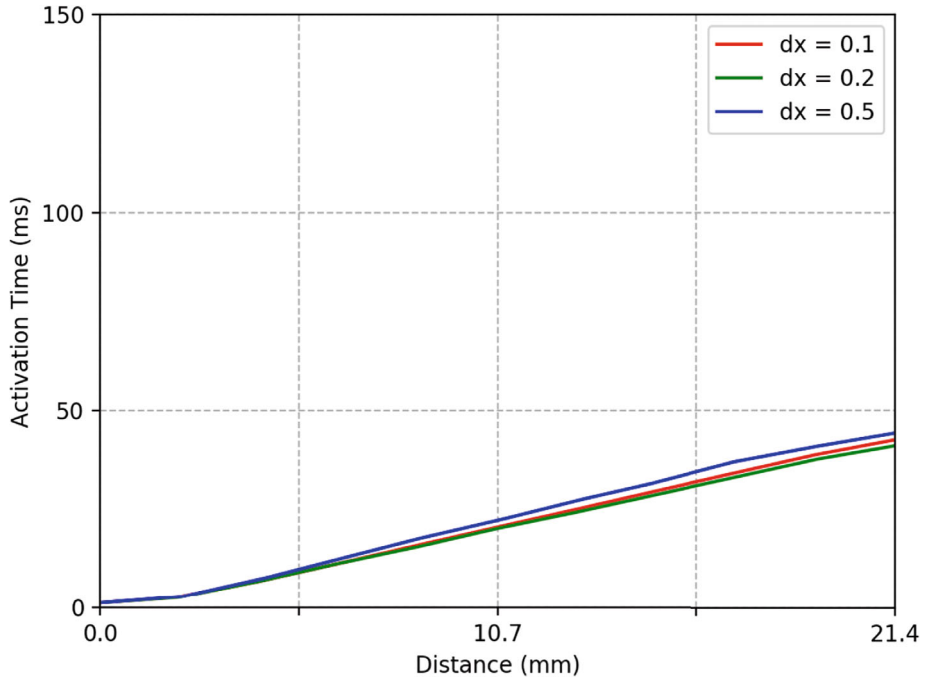

FIGURE 5 Activation times across a diagonal line of the Niederer benchmark domain

with a subcycling factor of $\zeta = 10$. Additionally, we include a test for $\Delta t = 0.1$ ms without any subcycling. Before running these tests, we need to determine what value to use for t_{thresh} for the HFE, HRK2, and HRK4 methods. Since we are using the Niederer mesh, we know that the whole domain undergoes electrical activation by the 50 ms mark, so we will use $t_{\text{thresh}} = 50$ ms. The importance of choosing t_{thresh} correctly will be discussed further in Section 4.2.

Since the algorithmic changes were only applied to the computation of the non-gating variables, it is natural to look at the solutions for these variables for all of the aforementioned algorithms, shown in Figures 6 and 7. In these figures, the solutions of the non-gating variables are plotted over time, for a single point in the same 24 processor partitioning of the Niederer mesh. From these figures, it can be seen that without any subcycling, simply setting $\Delta t = 0.1$ ms causes an inaccuracy in the form of an activation delay, causing the solutions to be shifted away from the reference solution until we reach the plateau phase where the solutions converge. Furthermore, we can see that our subcycling algorithms can use a timestep of $\Delta t = 0.1$ ms without significant loss of accuracy, i.e. the non-gating solutions closely match those of the reference solution. We can also notice that the solutions for the various subcycling methods are almost indistinguishable until we zoom in closely.

Using the same mesh, we can also compute the activation times (using the HFE algorithm) at points P1–P9, as well as the center of the domain C, and compare them to two reference solutions (which use a globally small timestep). We chose the HFE algorithm for this test because it is the fastest, and the other subcycling algorithms use the same pre-threshold subcycling method, so their activation times do not differ from HFE, as the threshold is set to occur after entire domain activation. These activation times, shown in Table 3, are very similar to the reference activation times, which is not surprising given the accuracy retention in the non-gating variables. When compared to the $\Delta x = 0.1$ mm, $\Delta t = 0.005$ ms reference solution, our HFE algorithm has, on average over all points in the domain, a difference of only 0.56% in the reference activation times, while for the $\Delta t = 0.01$ ms solution, the average difference is approximately 0.86%. This, in turn, means our subcycling algorithm retains roughly the same conduction velocities as our reference solutions, which are 0.59 m/s longitudinally and 0.12 m/s transversely.

4.2 | Subcycling time threshold

Since the goal of our subcycling algorithms is to better resolve the sharp jumps in the solutions of the ionic model variables with a larger timestep, it is important to perform subcycling until most of the domain has undergone electrical activation. To see why, we performed a test on the Niederer geometry using our HFE algorithm, but setting $t_{\text{thresh}} = 25$ ms instead of 50 ms. We then compare the overall solution (voltage) to all previously mentioned algorithms for a point which has not yet undergone electrical activation prior to the subcycling change. The results can be seen in Figure 8, where we can see that when setting t_{thresh} to 25 ms for the Niederer geometry, points which have not undergone electrical activation have a subsequent delay in activation, similar to the poor accuracy observed when using a global timestep of $\Delta t = 0.1$ ms without subcycling. This delay is not seen when t_{thresh} is set to a time after entire domain activation. Thus, when using a larger timestep with subcycling, the selection of the subcycling threshold is critical to retaining the accuracy of a smaller timestep. By choosing the threshold to turn off subcycling only after whole domain activation, we can achieve similar accuracy with a larger timestep.

4.3 | Performance and parallel scalability

Now that the accuracy of our subcycling algorithms has been established, we can look at their performance, in terms of the computational time. The whole point of using a larger timestep is to reduce the compute time taken to complete a

TABLE 2 Activation times for patient-specific geometry

Mesh size	P1	P2	P3	P4	P5	P6	P7	P8	P9
3M	28.50	31.16	7.35	30.40	27.40	26.91	14.77	15.47	13.46
2M	28.67	30.76	7.69	30.27	27.04	26.71	14.53	15.22	12.71
750K	28.61	30.41	7.54	29.66	26.35	25.91	14.53	14.93	14.16

Note: Mesh size is given by the number of points in the mesh.

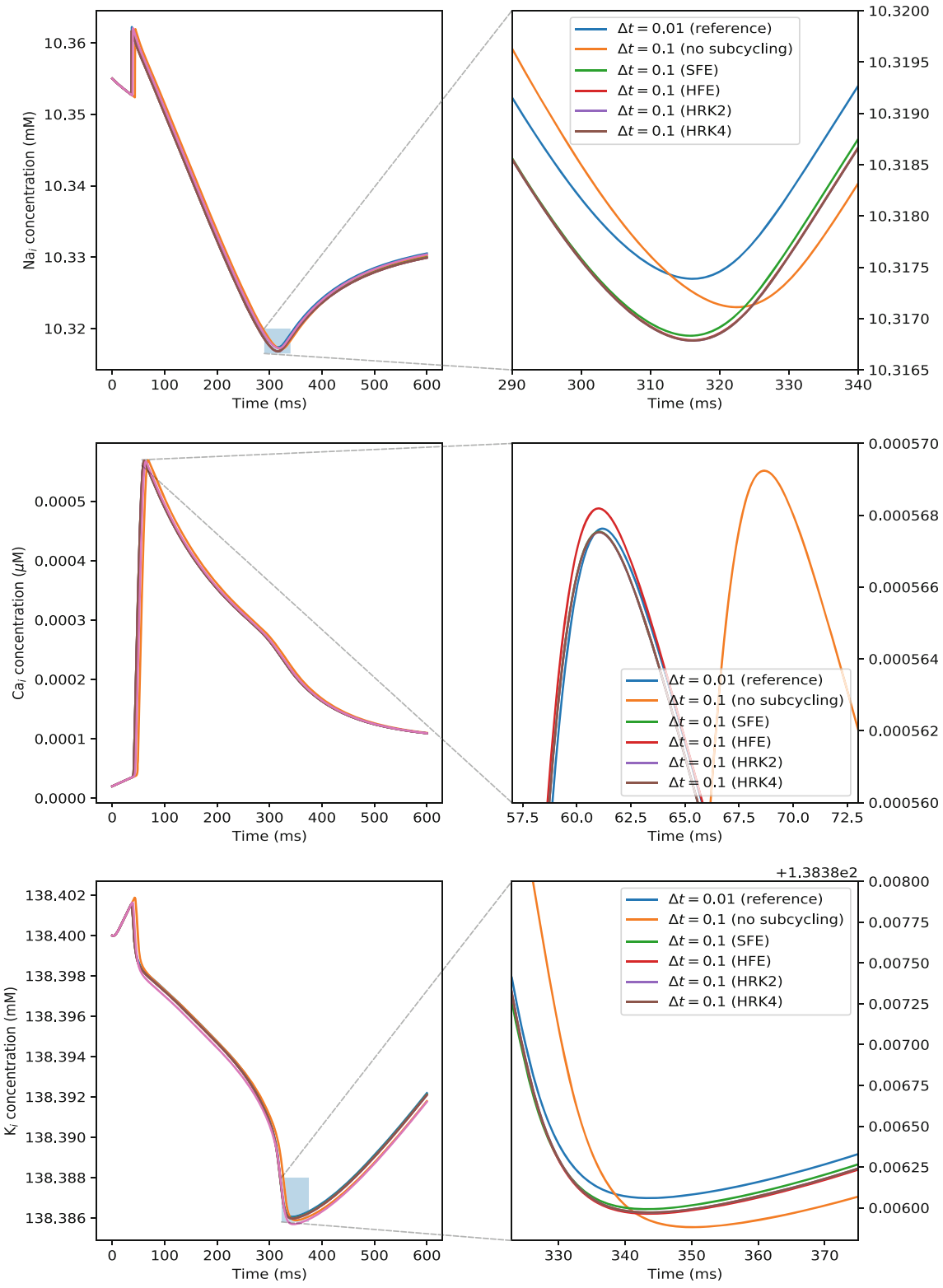

FIGURE 6 Solution comparison over time for Na_i (top), Ca_i (middle), and K_i (bottom) for various ionic model timestepping algorithms

given simulation. Using a larger timestep means fewer timesteps are needed to complete a cardiac cycle. For instance, consider our tests in Section 4.1, where we compared our use of subcycling with $\Delta t = 0.1$ ms to the Niederer benchmark, which uses $\Delta t = 0.01$ ms. Since our timestep is 10 times larger, we need 10 times fewer timesteps to simulate the same time period. There is a bit of a tradeoff here, because when $t < t_{\text{thresh}}$, a single timestep using subcycling is more

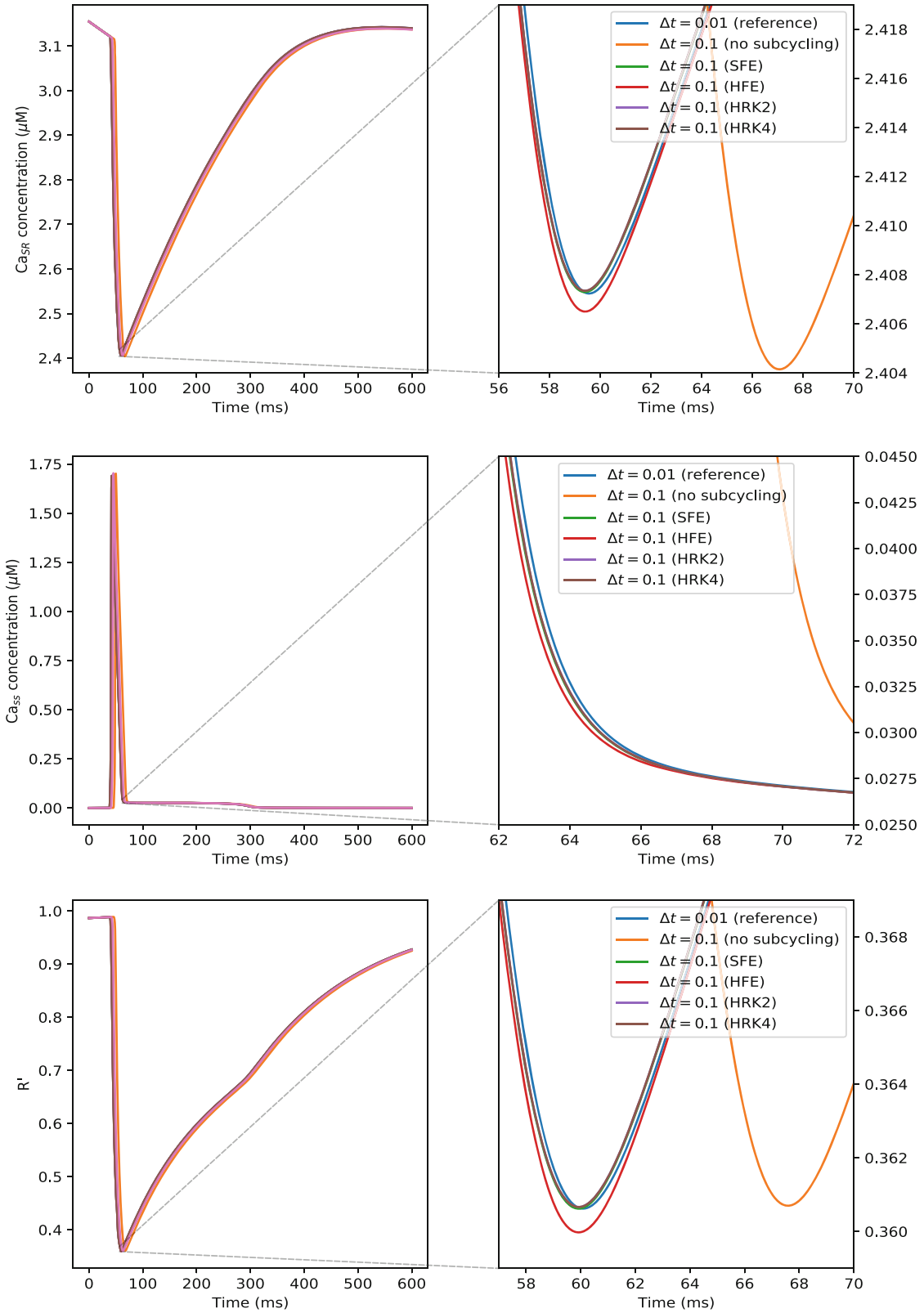

FIGURE 7 Solution comparison over time for Ca_{SR} (top), Ca_{SS} (middle), and R' (bottom) for various ionic model timestepping algorithms

expensive than one without subcycling. However, for our problem, it is still much more expensive to solve a single PDE timestep than a single ODE timestep, so the tradeoff becomes favorable. This is made even more favorable by the hybrid algorithms, because electrical activation occurs relatively early in the cardiac cycle, which means the subcycling threshold occurs very early as well. Thus, for most of a cardiac cycle, no subcycling is being performed.

TABLE 3 Comparison of activation times for the two smallest timestep reference solutions and our HFE algorithm on Niederer benchmark

Δx	Δt	ζ	P1	P2	P3	P4	P5	P6	P7	P8	C
0.1	0.005	–	1.27	33.53	7.95	34.56	24.28	41.32	25.48	42.14	20.36
0.1	0.01	–	1.27	33.88	8.14	34.90	24.52	41.69	25.71	42.51	20.38
0.1	0.1	10	1.25	33.68	8.07	34.68	24.32	41.36	25.50	42.19	20.21

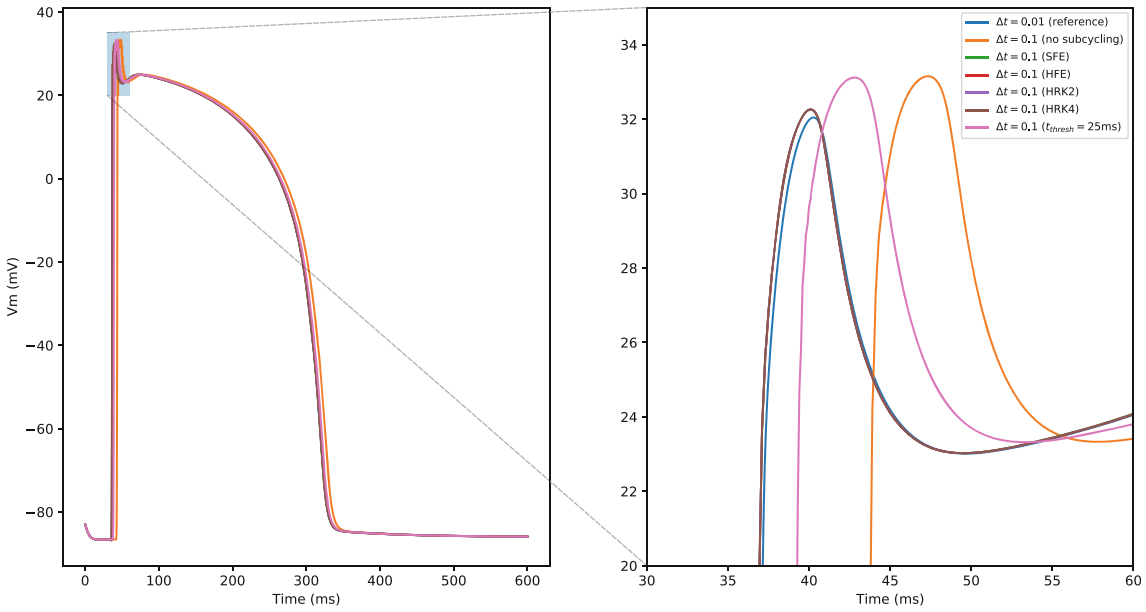

FIGURE 8 Prematurely ceasing subcycling causes a delay in activation

In terms of testing raw timings of the various subcycling algorithms, we use the same test setup as in Section 4.1, namely the 0.1 mm Niederer geometry. We run all tests until with 24 cores until $t = 600$ ms to get to the plateau phase, the results of which can be seen in Table 4. As expected, the subcycling algorithms outperform the reference method, in this case by 85%–90%, due to computing 10 times fewer timesteps. It can also be seen that the average time taken (per timestep) to solve the ionic model and perform the PCG solve increase when using the subcycling methods. This, however, is more than offset by the reduction in the number of timesteps. It should be noted that for all component timings, like those in Table 4, we exclude the time taken for the assembly of the matrices and vectors associated with the finite element method, as those are intrinsic to external packages. These assembly operations constitute a majority of the time for a simulation (especially for a large mesh), but they are not the focus of this paper.

In order to assess the parallel scalability of the code, we first perform a strong scalability study on the Niederer benchmark without the use of subcycling to establish reference scalability data. In particular, we created a finer mesh on the same rectangular geometry, with approximately 20 M points. We then set the timestep size to $\Delta t = 0.02$ ms and ran our code until $t = 50$ ms with a number of different processor counts in order to determine the speedup from using more processors. A slightly larger timestep of 0.02 ms was used here to allow for quicker results without affecting the solution accuracy significantly. To complement this scalability study, we also performed the same strong scalability tests on patient-specific geometry, using a finer version of the mesh from Section 4.1 with approximately 36 M points and using the same seven stimulus activation protocol. The only difference is that we ran the scalability test on the realistic geometry until $t = 20$ ms due to wall-time constraints for small core counts.

The result of these scalability studies, shown in Figure 9, shows good scalability performance up to 1000 cores, with parallel efficiencies of approximately 90% at 1000 processors, and an almost linear speedup. The drop in efficiency is likely due to the communication overhead of using a large number of processors relative to the size of the mesh. In other words, the amount of work per processor becomes too low, and communication costs start to dominate the timings.

TABLE 4 Timing comparison of subcycling algorithms and the reference solution on the Niederer geometry

Method	Total	Ionic model	Ionic model (avg)	PCG	PCG (avg)
Reference	13152.9	765.92	0.0128	1116.32	0.0186
SFE	2071.21	758.28	0.1264	161.38	0.0269
HFE	1447.89	135.76	0.0226	159.48	0.0266
HRK2	1474.41	153.29	0.0255	169.04	0.0282
HRK4	1505.88	185.83	0.0310	165.31	0.0276

Note: All timings are given in seconds, and averages are per timestep.

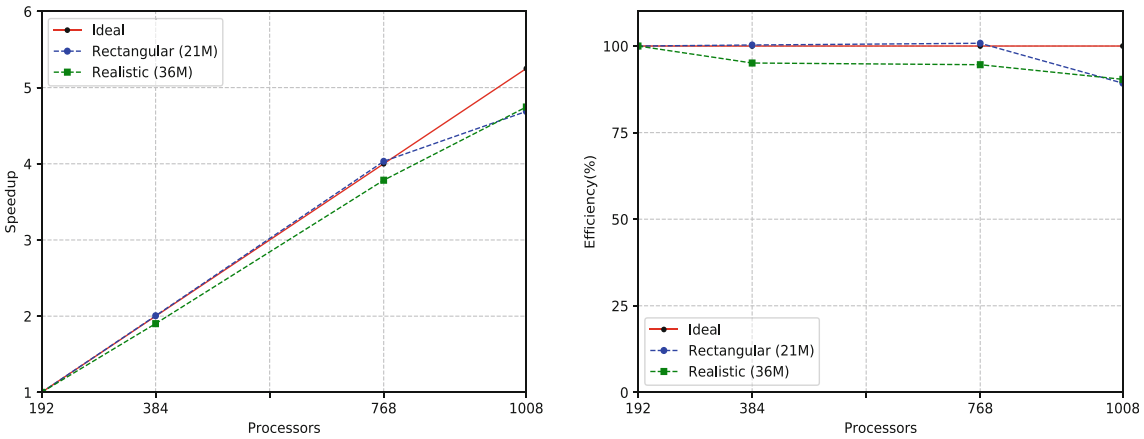

FIGURE 9 Speedup (left) and parallel efficiency (right) for both the rectangular geometry from the Niederer benchmark and the patient-specific geometry

Furthermore, in Figure 10 we can see the different components of the solver, and how each one scales with the number of processors. We can see that the evaluation of the ionic model scales very well even to 1000 processors, which is to be expected due to minimal communication costs involved in solving the ionic model. Additionally, the PCG setup and solve phases scale well, with slight performance degradations at 1000 processors due to communication overhead.

The scalability of our subcycling algorithms is not expected to change, since the only change occurs in the ionic model which scales almost perfectly. To compare the scalability using subcycling against the reference scalability, we run the same tests (both on ractangular and patient-specific geometry), except using the HFE algorithm. Additionally, we run this test until $t = 100$ ms with $t_{\text{thresh}} = 50$ ms to allow for equal time with and without subcycling. Table 5 shows overall timings for pre and post threshold, as well as solver components as the processour count increases. We can see that 0–50 ms timeframe (with subcycling) takes significantly longer to compute than the subsequent 50–100 ms timeframe, again due to the need to resolve the sharpness in the depolarization phase. Once the initial 50 ms have been resolved via subcycling, the entire rest of the cardiac cycle can be computed using a larger global timestep, yielding significant speedup.

As is the case for the reference simulation, scalability using the HFE algorithm is good up to 1000 processors, with parallel efficiency at approximately 90%–95% at 1000 processors (see Figure 11). Similar scalability results can be expected from the other subcycling methods (SFE, HRK2, and HRK4), since they are all variations of the same algorithm.

5 | CONCLUDING REMARKS

Given the predictive potential of cardiac simulations in clinical applications such as drug discovery and treatment planning, it is important to be able to rapidly simulate cardiac electrical propagation on patient-specific geometry.

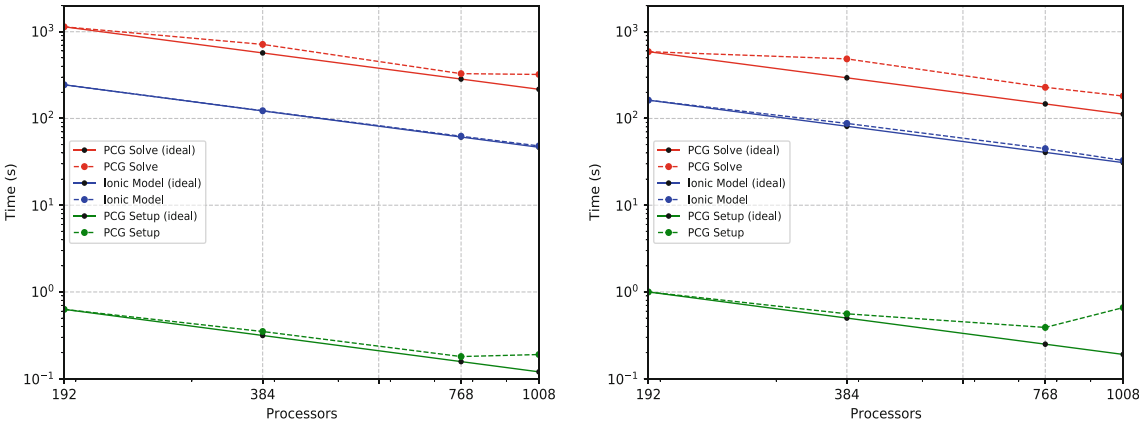

FIGURE 10 Solver component scalability for the rectangular geometry (left) and patient-specific geometry (right)

TABLE 5 Timings of 100 ms of simulation time using the 20 M node rectangular mesh for different processor counts

Cores	Total	$t < t_{\text{thresh}}$	$t \geq t_{\text{thresh}}$	Ionic	PCG Setup	PCG Solve
24	20045.2	12419.5	7625.74	4102.15	7.25	4600.16
48	9563.4	5911.58	3651.82	2055.91	3.5	1911.63
96	4882.16	2984.82	1897.34	1045.32	1.49	1013.85
192	2441.48	1490.6	950.88	536.04	0.77	528.8
384	1298.29	801.09	497.2	271.15	1.25	322.97
768	662.75	402.27	260.48	138.51	0.79	154.61
1008	519.05	316.66	202.39	107.52	1.02	134.73

Note: Here, $\Delta t = 0.1$ with $\zeta = 10$.

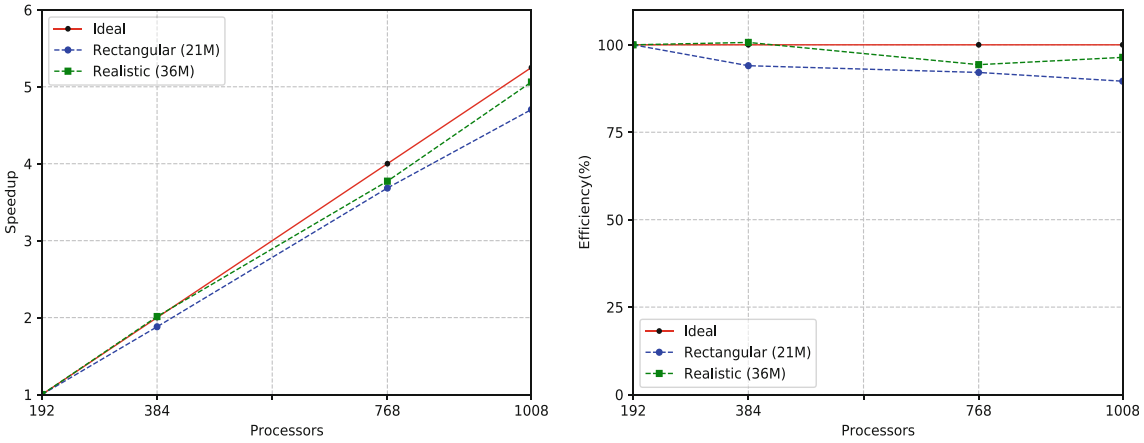

FIGURE 11 Speedup (left) and parallel efficiency (right) for both the rectangular geometry from the Niederer benchmark and the patient-specific geometry, using the HFE algorithm

Our base cardiac electrophysiology pipeline, adapted from the Cardioid package from LLNL, compares well to other approaches via the Niederer benchmark, exhibits both spatial and temporal convergence, and scales well up to 1000 cores, including on realistic patient geometry. In order to improve the turnaround time for a cardiac electrophysiology simulation, we proposed a novel subcycling time integration algorithm for the non-gating variables in the TT06 ionic model. This subcycling method allows a larger overall timestep to be used, while utilizing

subcycling (with a smaller timestep) for the rapid depolarization phase. The accuracy of this method is shown through various tests, including the Niederer benchmark and a seven stimulii activation pattern on patient-specific geometry, to be almost the same as when using a smaller global timestep. Furthermore, because the subcycling method only modifies the ODE solver in the ionic model, scalability remains excellent up to 1000 cores. When compared to the small timestep reference solutions, the subcycling methods show significant improvement to the overall completion time of a simulation, being 85%–90% faster than the original Cardioid FEM code for the Niederer benchmark.

The subcycling time integration algorithm presented has been shown to be promising for improving cardiac monodomain simulation time, but can still be improved in future work. One such improvement could be an automatic selection of the subcycling threshold, based on the underlying ionic model. This would further reduce simulation turnaround time, but also allow the subcycling algorithm to be seamlessly used on other potential ionic models, including the more recent ORd model.[50] However, further testing is required to determine the applicability of our subcycling algorithm to other ionic models.

ACKNOWLEDGMENTS

This work utilized resources from the University of Colorado Boulder Research Computing Group, which is supported by the National Science Foundation (awards ACI-1532235 and ACI-1532236), the University of Colorado Boulder, and Colorado State University.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID


Xiao-Chuan Cai https://orcid.org/0000-0003-0296-8640

REFERENCES

1. Srinivasan N, Schilling R. Sudden cardiac death and arrhythmias. *Arrhythmia Electrophysiol Rev*. 2018;7(2):111-117.
2. Colli Franzone P, Pavarino LF, Scacchi S. *Mathematical Cardiac Electrophysiology*. Springer; 2014.
3. Cooper J, Spiteri R, Mirams G. Cellular cardiac electrophysiology modeling with chaste and CellML. *Front Physiol*. 2015;5:511.
4. Mayourian J, Sobie E, Costa K. An introduction to computational modeling of cardiac electrophysiology and arrhythmogenicity. *Methods Mol Biol*. 2018;1816:17-35.
5. Krishnamoorthi S, Perotti L, Borgstrom N, et al. Simulation methods and validation criteria for modeling cardiac ventricular electrophysiology. *PLoS One*. 2014;9:1-29.
6. Bourgault Y, Pierre C. *Comparing the Bidomain and Monodomain Models in Electrocardiology through Convergence Analysis*. HAL Archives-ouvertes; 2010.
7. Plank G, Burton RA, Hales P, et al. Generation of histo-anatomically representative models of the individual heart: tools and application. *Philos Trans A: Math Phys Eng Sci*. 1896;2009(367):2257-2292.
8. Richards DF, Glosli JN, Draeger EW, et al. Towards real-time simulation of cardiac electrophysiology in a human heart at high resolution. *Comput Methods Biomech Biomed Eng*. 2013;16(7):802-805.
9. MFEM: Modular Finite Element Methods Library. mfem.org. Accessed September 10, 2021.
10. Falgout R, Yang U. Hypre: a library of high performance preconditioners. *ICCS '02*. Springer; 2002:632-641.
11. Arevalo H, Vadakkumpadan F, Guallar E, et al. Arrhythmia risk stratification of patients after myocardial infarction using personalized heart models. *Nat Commun*. 2016;7(11437):1-8.
12. Strocchi M, Augustin CM, Gsell MAF, et al. A publicly available virtual cohort of four-chamber heart meshes for cardiac electromechanics simulations. *PLoS One*. 2020;15(6):1-26.
13. Pathmanathan P, Bernabeu MO, Bordas R, et al. A numerical guide to the solution of the bidomain equations of cardiac electrophysiology. *Prog Biophys Mol Biol*. 2010;102:136-155.
14. Colli Franzone P, Pavarino LF. A parallel solver for reaction-diffusion systems in computational electrocardiology. *Math Models Methods Appl Sci*. 2004;14:883-911.
15. Munteanu M, Pavarino LF, Scacchi S. A scalable Newton-Krylov-schwarz method for the bidomain reaction-diffusion system. *SIAM J Sci Comput*. 2009;31:3861-3883.
16. Plank G, Liebmann M, Weber dos Santos R, Vigmond EJ, Haase G. Algebraic multigrid preconditioner for the cardiac bidomain model. *IEEE Trans Biomed Eng*. 2007;54(4):585-596.
17. Pavarino LF, Scacchi S. Multilevel additive schwarz preconditioners for the bidomain reaction-diffusion system. *SIAM J Sci Comput*. 2008;31:420-443.

18. Murillo M, Cai XC. A fully implicit parallel algorithm for simulating the non-linear electrical activity of the heart. *Numer Linear Algebra Appl*. 2004;11:261-277.
19. Bernabeu MO, Southern J, Wilson N, Strazdins P, Cooper J, Pitt-Francis J. Chaste: a case study of parallelisation of an open source finite-element solver with applications to computational cardiac electrophysiology simulation. *Int J High Performance Comput Appl*. 2014;28(1):13-32.
20. Niederer SA, Mitchell L, Smith N, Plank G. Simulating human cardiac electrophysiology on clinical time-scales. *Front Physiol*. 2011;2:14.
21. Vázquez M, Arís R, Houzeaux G, Aubry R, Villar P, Garcia-Barnés J. A massively parallel computational electrophysiology model of the heart. *Int J Numer Methods Biomed Eng*. 2011;27(12):1911-1929.
22. Niederer SA, Kerfoot E, Benson A, et al. Verification of cardiac tissue electrophysiology simulators using an N-version benchmark. *Philos Trans A: Math Phys Eng Sci*. 2011;369(1954):4331-4351.
23. Southern J, Gorman GJ, Piggott MD, Farrell PE, Bernabeu MO, Pitt-Francis J. Anisotropic mesh adaptivity for cardiac electrophysiology. *Proc Comput Sci*. 2010;1(1):935-944.
24. Belhamadia Y, Fortin A, Bourgault Y. Towards accurate numerical method for monodomain models using a realistic heart geometry. *Math Biosci*. 2009;220(2):89-101.
25. Heidenreich EA, Ferrero JM, Doblare M, Rodríguez JF. Adaptive macro finite elements for the numerical solution of monodomain equations in cardiac electrophysiology. *Ann Biomed Eng*. 2010;38(7):2331-2345.
26. Qu Z, Garfinkel A. An advanced algorithm for solving partial differential equation in cardiac conduction. *IEEE Trans Biomed Eng*. 1999; 46(9):1166-1168.
27. Mountris KA, Pueyo E. A dual adaptive explicit time integration algorithm for efficiently solving the cardiac monodomain equation. *Int J Numer Methods Biomed Eng*. 2021;37(7):e3461.
28. Gomes JM, Alvarenga A, Campos RS, Rocha BM, da Silva APC, Weber dos Santos R. Uniformization method for solving cardiac electrophysiology models based on the Markov-chain formulation. *IEEE Trans Biomed Eng*. 2015;62(2):600-608.
29. Gomes JM, Oliveira RS, Lobosco M, Weber dos Santos R. Adaptive-step methods for Markov-based membrane models. *Commun Nonlinear Sci Numer Simul*. 2020;85:85.
30. Hodgkin AL, Huxley AF. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol*. 1952;117:500-544.
31. ten Tusscher KHWJ, Panfilov A. Alternans and spiral breakup in a human ventricular tissue model. *Am J Physiol Heart Circ Physiol*. 2006;291(3):1088-1100.
32. Pagani S, Dede L, Manzoni A, Quarteroni A. Data integration for the numerical simulation of cardiac electrophysiology. *Pacing Clin Electrophysiol*. 2021;44(4):726-736.
33. Shade JK, Cartoski MJ, Nikolov P, et al. Ventricular arrhythmia risk prediction in repaired tetralogy of Fallot using personalized computational cardiac models. *Heart Rhythm*. 2020;17(3):408-414.
34. Crank J, Nicolson P. A practical method for numerical evaluation of solutions of partial differential equations of the heat conduction type. *Proc Camb Phil Soc*. 1947;43(1):50-67.
35. Fedorov A, Beichel R, Kalpathy-Cramer J, Finet J. 3D slicer as an image computing platform for the quantitative imaging network. *Magn Reson Imaging*. 2012;30(9):1323-1341.
36. Ungi T, Lasso A, Fichtinger G. Open-source platforms for navigated image-guided interventions. *Med Image Anal*. 2016;33:181-186.
37. Edelsbrunner H, Kirkpatrick DG, Seidel R. On the shape of a set of points in the plane. *IEEE Trans Inform Theory*. 1983;29(4):551-559.
38. Schroeder W, Martin K, Lorensen B. *The Visualization Toolkit*. Kitware, Inc; 2006.
39. Radau P, Lu Y, Connelly K, Paul G, Dick A, Wright GA. Evaluation framework for algorithms segmenting short axis cardiac MRI MIDAS J. *Cardiac MR Left Ventricle Segmentation Challenge*. 2009;49.
40. Valette S, Chassery JM, Prost R. Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams. *IEEE Trans Vis Comput Graph*. 2008;14(2):369-381.
41. Hang S. TetGen, A Delaunay-based quality tetrahedral mesh generator. *ACM Trans Math Softw*. 2015;41(2):11.
42. Frey P. *MEDIT: an Interactive Mesh Visualization Software. Technical Report*. Institut National de Recherche en Informatique et en Automatique; 2001.
43. Rush S, Larsen H. A practical algorithm for solving dynamic membrane equations. *IEEE Trans Biomed Eng*. 1978;25(4):389-392.
44. Marsh ME, Torabi Ziaratgahi S, Spiteri RJ. The secrets to the success of the Rush-Larsen method and its generalizations. *IEEE Trans Biomed Eng*. 2012;59(9):2506-2515.
45. Hestenes MR, Stiefel E. Methods of conjugate gradients for solving linear systems. *J Res Natl Bur Stand*. 1952;49:409-436.
46. Baker AH, Falgout RD, Kolev TV, Yang UM. Multigrid smoothers for ultraparallel computing. *SIAM J Sci Comput*. 2011;33(5): 2864-2887.
47. De Sterck H, Yang UM, Heys JJ. Reducing complexity in parallel algebraic multigrid preconditioners. *SIAM J Matrix Anal Appl*. 2005; 27(4):1019-1039.
48. De Sterck H, Falgout RD, Nolting JW, Yang UM. Distance-two interpolation for parallel algebraic multigrid. *Numer Linear Algebra Appl*. 2008;15(2–3):115-139.
49. Cardonne-Noott L, Bueno-Orovio A, Mincholé A, Zemzemi N, Rodriguez B. Human ventricular activation sequence and the simulation of the electrocardiographic QRS complex and its variability in healthy and intraventricular block conditions. *Europace*. 2016;18:iv4-iv15.

50. O'Hara T, Virág L, Varró A, Rudy Y. Simulation of the undiseased human cardiac ventricular action potential: model formulation and experimental validation. *PLoS Comput Biol*. 2011;7(5):e1002061.

How to cite this article: Laudenschlager S, Cai X-C. An inner-outer subcycling algorithm for parallel cardiac electrophysiology simulations. *Int J Numer Meth Biomed Engng*. 2023;39(3):e3677. doi:10.1002/cnm.3677