Hindawi Publishing Corporation
Computational and Mathematical Methods in Medicine
Volume 2012, Article ID 824569, 13 pages
doi:10.1155/2012/824569

Research Article

Simulations of Complex and Microscopic Models of Cardiac Electrophysiology Powered by Multi-GPU Platforms

Bruno Gouvêa de Barros,[1] Rafael Sachetto Oliveira,[2,3] Wagner Meira Jr.,[3] Marcelo Lobosco,[1] and Rodrigo Weber dos Santos[1]

[1] *Computational Modeling, Federal University of Juiz de Fora, 36036-900 Juiz de Fora, MG, Brazil*
[2] *Computer Science, Federal University of São João del-Rei, 36307-352 São João del-Rei, MG, Brazil*
[3] *Computer Science, Federal University of Minas Gerais, 31270-901 Belo Horizonte, MG, Brazil*

Correspondence should be addressed to Rodrigo Weber dos Santos, rodrigo.weber@ufjf.edu.br

Received 3 August 2012; Revised 28 September 2012; Accepted 1 October 2012

Academic Editor: Ling Xia

Key aspects of cardiac electrophysiology, such as slow conduction, conduction block, and saltatory effects have been the research topic of many studies since they are strongly related to cardiac arrhythmia, reentry, fibrillation, or defibrillation. However, to reproduce these phenomena the numerical models need to use subcellular discretization for the solution of the PDEs and nonuniform, heterogeneous tissue electric conductivity. Due to the high computational costs of simulations that reproduce the fine microstructure of cardiac tissue, previous studies have considered tissue experiments of small or moderate sizes and used simple cardiac cell models. In this paper, we develop a cardiac electrophysiology model that captures the microstructure of cardiac tissue by using a very fine spatial discretization ($8\,\mu$m) and uses a very modern and complex cell model based on Markov chains for the characterization of ion channel's structure and dynamics. To cope with the computational challenges, the model was parallelized using a hybrid approach: cluster computing and GPGPUs (general-purpose computing on graphics processing units). Our parallel implementation of this model using a multi-GPU platform was able to reduce the execution times of the simulations from more than 6 days (on a single processor) to 21 minutes (on a small 8-node cluster equipped with 16 GPUs, i.e., 2 GPUs per node).

1. Introduction

Heart diseases are responsible for one third of all deaths worldwide [1]. Cardiac electrophysiology is the trigger to the mechanical deformation of the heart. Therefore, the knowledge of cardiac electrophysiology is essential to understand many aspects of cardiac physiological and physiopathological behavior [2]. Computer models of cardiac electrophysiology [3, 4] have become valuable tools for the study and comprehension of such complex phenomena, as they allow different information acquired from different physical scales and experiments to be combined to generate a better picture of the whole system functionality. Not surprisingly, the high complexity of the biophysical processes translates into complex mathematical and computational models. Modern cardiac models are described by nonlinear system of partial differential equations (PDEs) that may result in a problem with millions of unknowns.

Mathematical models for cell electrophysiology are a key component of cardiac modeling. They serve both as standalone research tools, to investigate the behavior of single cardiac myocytes, and as an essential component of tissue and organ simulation based on the so-called bidomain or monodomain models [4]. The cell models can be written as a general non-linear system of ordinary differential equations (ODEs) and may vary in complexity from simple phenomenological models [5] (based on two variables) to complex models describing a large number of detailed physiological processes [6] (based on 40 to 80 differential variables). Simple models focus on the genesis of action potential (AP), that propagates from cell to cell and generates an electric wave that propagates on the heart. Complex models

account not only for the genesis of AP but also describe how this phenomenon is related to cardiac homeostasis and to different sub-cellular components, such as cell membrane's ion channels. Advances in genetics, molecular biology, and electrophysiology experiments have provided new data and information related to the structure and function of ion channels. The Markov Chain (MC) model formalism has been increasingly used to describe both function and structure of ion channels. MC-based models have enabled simulations of structural abnormalities due to genetic diseases and drug-biding effects on ion channels [7–9]. Unfortunately, these modern cardiac myocyte models pose different challenges to both numerical methods, due to the stiffness of the ODEs introduced by MCs, and to high performance computing, due to the size of the problems, since the number of differential variables rises from a couple to near a hundred [10].

On the tissue level, the bidomain model [4] is considered to be the most complete description of the electrical activity. This nonlinear system of PDEs can be simplified to the so-called monodomain model, which may be less accurate but less computationally demanding than the bidomain model. Unfortunately, large scale simulations, such as those resulting from the discretization of an entire heart, remain a computational challenge. In addition, key aspects of cardiac electrophysiology, such as slow conduction, conduction block, and saltatory or sawtooth effects, demand sub-cellular discretization for the solution of the PDEs and nonuniform, heterogeneous tissue electric conductivity. These aspects of cardiac electrophysiology are strongly related to cardiac arrhythmia, reentry, fibrillation or defibrillation, and have been the research topic of many studies [11–20].

However, the demand of sub-cellular discretization for the solution of the PDEs and nonuniform, heterogeneous tissue electric conductivity have prevented the study of the aforementioned phenomena on large-scale tissue simulations. In addition, due to the high computational costs associated with the simulations of these microscopic models of cardiac tissue, previous works have adopted simple myocyte models, instead of modern MC-based models [6, 10].

In this work, we present a solution for this problem based on multi-GPU platforms (clusters equipped with graphics processing units) that allows fast simulations of microscopic tissue models combined with modern and complex myocyte models. The solution is based on merging two different high-performance techniques. We have previously investigated for cardiac modeling: cluster computing based on message passing communications (MPI) [21–24] and GPGPU (General-purpose computing on graphics processing units) [25–30]. We developed a two-dimensional model that is based on the previous work of Spach and collaborators [11, 17] that accounts for the microstructure of cardiac tissue, gap junction heterogeneous distribution, and discretizations of $8\,\mu$m. This microscopic tissue model was combined with the model of Bondarenko et al. [6] which is a modern and complex myocyte model based on MCs. Our parallel implementation of this model using a multi-GPU platform was able to reduce the execution times of the simulations from more than 6 days (on a single processor) to 21 minutes (on a small 8-node cluster equipped with 16 GPUs, that is, 2 GPUs per node). As a result, using this very fast parallel implementation we were able to simulate the formation of spiral waves, a form of self-sustained reentrant activity strongly associated with cardiac arrhythmia. To the best of our knowledge, this is the first time spiral waves are simulated using a cardiac model that accounts for both the microstructure of cardiac tissue and a modern and complex myocyte model.

2. Methods

2.1. Modeling Cardiac Microstructure. We developed a two-dimensional model that is based on the previous work of Spach and collaborators [11, 17] that accounts for the microstructure of cardiac tissue, gap junction heterogeneous distribution, and discretizations of $8\,\mu\text{m} \times 8\,\mu\text{m}$. A basic template for myocyte connections was developed and is presented in Figure 1. This basic unit accounts for the connection of a total of 32 cardiac myocytes with different shapes and numbers of neighboring cells. The mean and SD (standard deviation) values for cell length and width are $120.9 \pm 27.8\,\mu$m and $18.3 \pm 3.5\,\mu$m, respectively. These values are close to those reported in the literature: [31] (length = $140\,\mu$m and width = $19\,\mu$m), [32] (length = $134\,\mu$m and width = $18\,\mu$m), and [20] (length = $100\,\mu$m and width = $17.32\,\mu$m). On average, each cell connects to other 6 neighboring myocytes. Our two-dimensional model considers a homogeneous depth $d = 10\,\mu$m [11, 17].

This basic unit was created in such a way that it allows the generation of larger tissue preparations via the connections of multiple instances of it. Figure 2 presents how this can be achieved.

Figure 3 presents an example of how the connections between different myocytes can be arranged. The code was developed in a flexible way, so that it allows the user to set up for each discretized volume $\text{Vol}_{i,j}$ (with area $= h \times h$) conductivity or conductance values for the north ($\sigma_{x_{i,j+1/2}}$), south ($\sigma_{x_{i,j-1/2}}$), west ($\sigma_{x_{i-1/2,j}}$), and east ($\sigma_{x_{i+1/2,j}}$) volume faces. These can be any nonnegative values. In this work, we set the discretization h to $8\,\mu$m. In addition, based on the work of Spach and collaborators [11, 17], we chose only 5 possible types of connections between neighboring volumes that are membrane ($\sigma_m = 0.0$), cytoplasm ($\sigma_c = 0.4\,\mu\text{S}/\mu\text{m}$), gap junction plicate ($G_p = 0.5\,\mu$S), interplicate ($G_i = 0.33\,\mu$S), and combined plicate ($G_c = 0.062\,\mu$S), where we use σ for conductivity and G for conductance. For the simulations presented in this work, the distribution of the different gap junctions within the 32 myocytes was not randomly generated. Instead, the gap junction distribution of the basic template unit was manually chosen to reproduce the distribution presented before in [11, 17]. With this setup and conductivity values we found that conduction velocity along the fibers was around $410\,\mu$m/ms (LP) and was $130\,\mu$m/ms transversal to fiber direction (TP). This results in a ratio LP/TP of 0.32, which is close to the conduction ratio reported in [11].

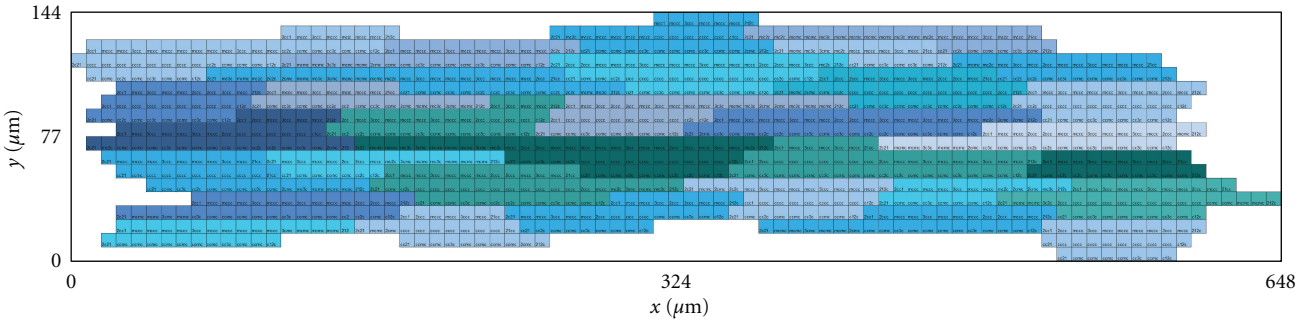


Figure 1: Basic unit of cardiac myocyte distribution based on a total of 32 cells. Cells are displayed in different alternating colors along the x-axis. The basic unit spans a total of 648 μm in the longitudinal direction versus 144 μm in the transversal direction.

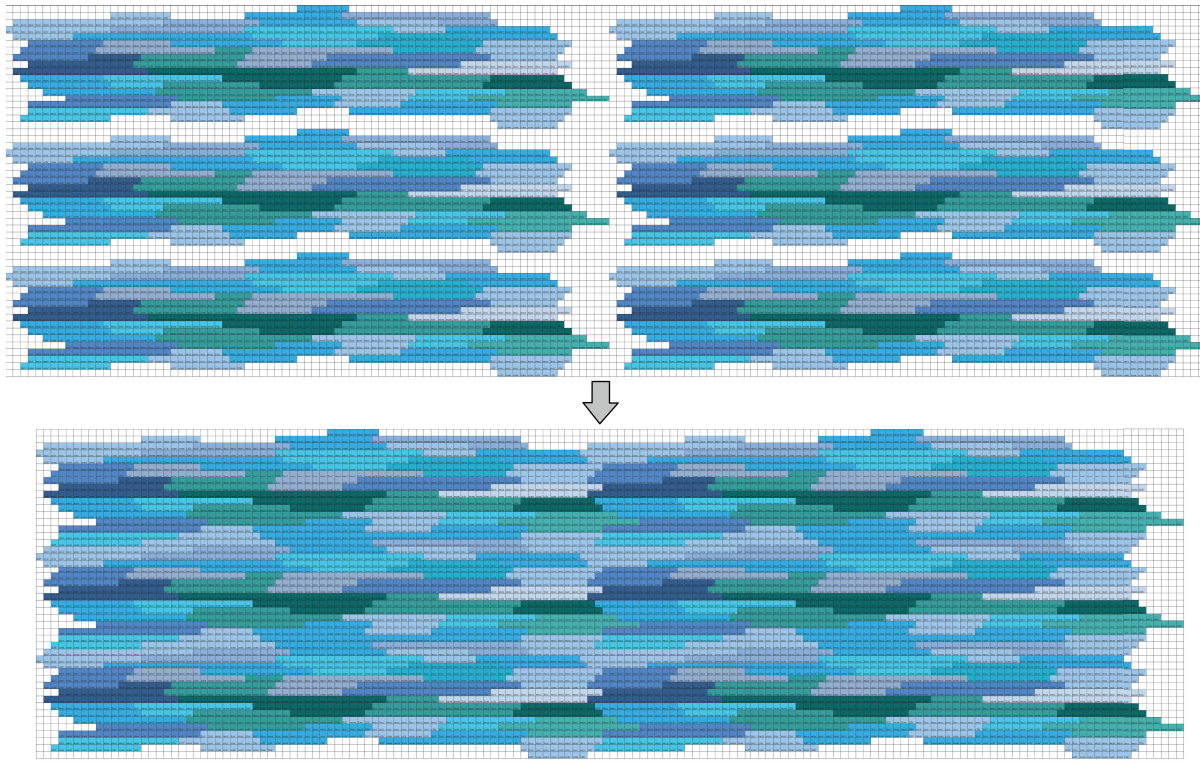


Figure 2: Six basic units being combined to form a larger tissue.

2.2. The Heterogeneous Monodomain Model. Action potentials propagate through the cardiac tissue because the intracellular space of cardiac cells is electrically coupled by gap junctions. In this work, we do not consider the effects of the extracellular matrix. Therefore, the phenomenon can be described mathematically by a reaction-diffusion type partial differential equation (PDE) called monodomain model, given by

$$\begin{aligned}\beta C_m \frac{\partial V(x,y,t)}{\partial t} + \beta I_{\text{ion}}(V(x,y,t), \boldsymbol{\eta}(x,y,t)) \\ = \nabla \cdot (\boldsymbol{\sigma}(x,y) \nabla V(x,y,t)) + I_{\text{stim}}(x,y,t), \\ \frac{\partial \boldsymbol{\eta}(x,y,t)}{\partial t} = \mathbf{f}(V(x,y,t), \boldsymbol{\eta}(x,y,t)),\end{aligned} \tag{1}$$

where V is the variable of interest and represents the transmembrane potential, that is, the difference between intracellular to extracellular potential; $\boldsymbol{\eta}$ is a vector of state variables that also influences the generation and propagation of the electric wave and usually includes the intracellular concentration of different ions (K^+, Na^+, Ca^{2+}) and the permeability of different membrane ion channels; β is the surface-volume ratio of heart cells; C_m is the membrane capacitance, I_{ion} the total ionic current, which is a function of V and a vector of state variables $\boldsymbol{\eta}$; I_{stim} is the current due to an external stimulus, $\boldsymbol{\sigma}$ is the monodomain conductivity tensor. We assume that the boundary of the tissue is isolated, that is, no-flux boundary conditions ($\mathbf{n} \cdot \sigma \nabla V = 0$ on $\partial\Omega$).

In this work, the modern and complex Bondarenko et al. model [6] that describes the electrical activity of left

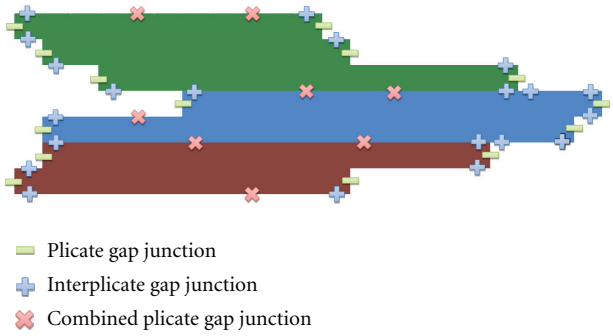


Figure 3: In this work, there are only 5 possible types of connections between neighboring volumes that are membrane, which indicated no-flux between neighboring volumes; cytoplasm, which indicates that the neighboring volumes are within the same cell; three possible types of gap junctions, plicate, interplicate, and combined plicate. For the simulations presented in this work, the gap junction distribution of the basic template unit was manually chosen to reproduce the distribution presented before in [11, 17]. This figure presents an example of how different gap junctions are distributed in three neighboring myocytes that belong to the basic unit.

ventricular cells of mice was considered to simulate the kinetics of I_{ion} in (1). The Bondarenko et al. model (BDK) was the first model presented for mouse ventricular myocytes [6]. The ionic current term I_{ion} in this model consists of the sum of 15 transmembrane currents. In short, Bondarenko's model is based on a ordinary differential equation (ODE) with 41 differential variables that control ionic currents and cellular homeostasis. In this model, most of the ionic channels are represented by Markov chains (MCs).

2.3. Numerical Discretization in Space and Time. The finite volume method (FVM) is a mathematical method used to obtain a discrete version of partial differential equations. This method is suitable for numerical simulations of various types of conservation laws (elliptical, parabolic, or hyperbolic) [33]. Like the finite element method (FEM), the FVM can be used in several types of geometry, using structured or unstructured meshes, and generates robust numerical schemes. The development of the method is intrinsically linked to the concept of flow between regions or adjacent volumes, that is, it is based on the numerical calculation of net fluxes into or out of a control volume. For some isotropic problems discretized with regular spatial meshes, the discretization obtained with the FVM is very similar to the one obtained with the standard finite difference method (FDM).

This section presents a brief description of the FVM application to the time and spatial discretization of the heterogeneous monodomain equations. Detailed information about the FVM applied to the solution of monodomain can be found in [34, 35].

2.3.1. Time Discretization. The reaction and diffusion parts of the monodomain equations were split by employing the Godunov operator splitting [36]. Therefore, each time step involves the solution of two different problems: a nonlinear system of ODEs

$$\begin{aligned}\frac{\partial V}{\partial t} &= \frac{1}{C_m}\left[-I_{\text{ion}}(V,\boldsymbol{\eta}) + I_{\text{stim}}\right],\\ \frac{\partial \boldsymbol{\eta}}{\partial t} &= f(V,\boldsymbol{\eta}),\end{aligned} \tag{2}$$

and a parabolic PDE

$$\beta\left(C_m\frac{\partial V}{\partial t}\right) = \nabla\cdot(\boldsymbol{\sigma}\nabla V). \tag{3}$$

Since the spatial discretization of our model, h, is extremely small, the CFL [37] condition that assures numerical stability is very restrictive. Therefore, for the PDE we used the unconditionally stable implicit Euler scheme. The time derivative presented in (3), which operates on V is approximated by a first-order implicit Euler scheme as follows:

$$\frac{\partial V}{\partial t} = \frac{V^{n+1} - V^n}{\Delta t_p}, \tag{4}$$

where V^n represents the transmembrane potential at time t_n and Δt_p is the time step used to advance in time the partial differential equation.

For the discretization of the nonlinear system of ODEs, we note that its stiffness demands very small time steps. For simple models based on Hodgkin-Huxley formulation, this problem is normally overcome by using the Rush-Larsen (RL) method [38]. However, for the most modern and complex models that are highly based on MCs, the RL method seems to be ineffective in terms of allowing larger time steps during the numerical integration. For the case of the Bondarenko et al. model, we tested both methods, Euler and RL, and both demanded the same time step, $\Delta t_o = 0.0001$ ms for stability issues. Since the RL method is more expensive per time step than the Euler method, in this work, we used the simple explicit Euler method for the discretization of the nonlinear ODEs.

However, as already indicated above, we use different time steps for the solution of the two different uncoupled problems, the PDE and the ODEs. Since we use an unconditionally stable method for the PDE, the time step Δt_p could be much larger than that used for the solution of the nonlinear system of ODEs, $\Delta t_o = 0.0001$ ms. In this work, we use $\Delta t_p = 0.01$ ms, that is, a hundred times larger than Δt_o. This has not introduced any significant numerical error. We calculated the L2 relative error for the transmembrane potential between a solution that uses the same time step for both the ODE and the PDE, $\Delta t_o = \Delta t_p = 0.0001$ ms, $V_{m_{\text{ref}}}$ and a solution that uses $\Delta t_o = 0.0001$ ms and $\Delta t_p = 0.01$ ms, V as follows:

$$\text{error} = \frac{\sqrt{\sum_{i=1}^{nt}\sum_{j=1}^{nv}\left(V(i,j) - V_{m_{\text{ref}}}(i,j)\right)^2}}{\sqrt{\sum_{i=1}^{nt}\sum_{j=1}^{nv}V_{m_{\text{ref}}}(i,j)^2}}, \tag{5}$$

where nt is the number of time steps and nv is the total number of discretized volumes. For the simulation of a tissue of size 0.5×0.5 cm during 20 ms (stimulus at the center of the tissue), the error found was 0.01%.

2.3.2. Spatial Discretization. The diffusion term in (3) must be discretized in space. For this we will consider the following:

$$\mathbf{J} = -\boldsymbol{\sigma}\nabla V, \tag{6}$$

where $\mathbf{J}$ (μA/cm^2) expresses the density of intracellular current flow and

$$\nabla \cdot \mathbf{J} = -I_v. \tag{7}$$

In this equation, I_v (μA/cm^3) is a volumetric current and corresponds to the left-hand side of (3), serving as the base for this finite volume solution.

For the space discretization, we will consider a two-dimensional uniform mesh, consisting of regular quadrilaterals (called "volumes"). Located in the center of each volume is a node. The quantity of interest V is associated with each node of the mesh.

After defining the mesh geometry and dividing the domain in control volumes, the specific equations of the FVM can be presented. Equation (7) can be integrated spatially over an individual volume $V_{i,j}$ of size $h^2 d$, leading to

$$\int_\Omega \nabla \cdot \mathbf{J}\, dv = -\int_\Omega I_v\, dv. \tag{8}$$

Applying the divergence theorem yields

$$\int_\Omega \nabla \cdot \mathbf{J}\, dv = \int_{\partial\Omega} \mathbf{J} \cdot \vec{\xi}\, ds, \tag{9}$$

where $\vec{\xi}$ is the unitary normal vector to the boundary $\partial\Omega$. Then, we have

$$\int_{\partial\Omega} \mathbf{J} \cdot \vec{\xi}\, ds = -\int_\Omega I_v\, dv. \tag{10}$$

Finally, assuming that I_v represents an average value in each particular quadrilateral, and substituting (3) in (10), we have

$$\beta \left(C_m \frac{\partial V}{\partial t} \right) \Bigg|_{(i,j)} = \frac{-\int_{\partial\Omega} \mathbf{J}_{i,j} \cdot \vec{\xi}\, ds}{h^2 d}. \tag{11}$$

For this particular two-dimensional problem, consisting of a uniform grid of quadrilaterals with side h, the calculation of $\mathbf{J}_{i,j}$ can be subdivided as a sum of flows on the following faces:

$$\int_{\partial\Omega} \mathbf{J}_{i,j} \cdot \vec{\xi}\, ds = \left(I_{x_{i+1/2,j}} - I_{x_{i-1/2,j}} + I_{y_{i,j+1/2}} - I_{y_{i,j-1/2}} \right), \tag{12}$$

where $I_{x_{m,n}}$ and $I_{y_{m,n}}$ are calculated at faces ($(m,n) = (i + 1/2, j)$, $(i - 1/2, j)$, $(i, j + 1/2)$, or $(i, j - 1/2)$) as follows. For the case in which we have defined a conductivity value at face (m,n), for instance the intracellular, or cytoplasm conductivity, σ_c, as described in Section 2.1, we have

$$\begin{aligned} I_{x_{m,n}} &= -\sigma_c(m,n) \left. \frac{\partial V}{\partial x} \right|_{(m,n)} hd, \\ I_{y_{m,n}} &= -\sigma_c(m,n) \left. \frac{\partial V}{\partial y} \right|_{(m,n)} hd. \end{aligned} \tag{13}$$

For the case in which we have defined a conductance value at face (m,n), for instance a gap junction conduction G, as describes in Section 2.1, we have:

$$\begin{aligned} I_{x_{m,n}} &= -G(m,n)\Delta_x V|_{(m,n)}, \\ I_{y_{m,n}} &= -G(m,n)\Delta_y V|_{(m,n)}. \end{aligned} \tag{14}$$

Using centered finite difference, we have for (13)

$$\begin{aligned} \left. \frac{\partial V}{\partial x} \right|_{(i+1/2,j)} &= \frac{V_{i+1,j} - V_{i,j}}{h}, \\ \left. \frac{\partial V}{\partial y} \right|_{(i,j+1/2)} &= \frac{V_{i,j+1} - V_{i,j}}{h}. \end{aligned} \tag{15}$$

For (14), we have

$$\begin{aligned} \Delta_x V|_{(i+1/2,j)} &= V_{i+1,j} - V_{i,j}, \\ \Delta_y V|_{(i,j+1/2)} &= V_{i,j+1} - V_{i,j}. \end{aligned} \tag{16}$$

Equations for $\partial V/\partial x|_{(i-1/2,j)}$, $\partial V/\partial y|_{(i,j-1/2)}$, $\Delta_x V|_{(i-1/2,j)}$ and $\Delta_y V|_{(i,j+1/2)}$ can be obtained analogously.

Rearranging and substituting the discretizations of (4) and (12) in (11) and decomposing the operators as described by (2), and (3) yields

$$\begin{aligned} &\left(\sigma_{i+1/2,j} + \sigma_{i-1/2,j} + \sigma_{i,j+1/2} + \sigma_{i,j-1/2} + \alpha \right) V^*_{i,j} - \sigma_{i,j-1/2} V^*_{i,j-1} \\ &- \sigma_{i+1/2,j} V^*_{i+1,j} - \sigma_{i,j+1/2} V^*_{i,j+1} - \sigma_{i-1/2,j} V^*_{i-1,j} = \alpha V^n_{i,j}, \end{aligned} \tag{17}$$

$$\begin{aligned} C_m \frac{V^{n+1}_{i,j} - V^*_{i,j}}{\Delta t_o} &= -I_{\text{ion}}\left(V^*_{i,j}, \boldsymbol{\eta}^n \right), \\ \frac{\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^n}{\Delta t_o} &= f(\boldsymbol{\eta}^n, V^*, t), \end{aligned} \tag{18}$$

where $\alpha = (\beta C_m h^2)/\Delta t_p$, n is the current step, $*$ is an intermediate step, and $n+1$ is the next time step. In addition σ can stand for any of the gap junction conductance (G_p, G_i, G_c) divided by the depth d or for any conductivity value (σ_c, σ_m) defined for each volume face as described in Section 2.1. This defines the equations for each finite volume $\text{Vol}_{i,j}$. First we solve the linear system associated with (17) to advance time by Δt_p and then we solve the nonlinear system of ODEs associated with (18) N_o times until we have $N_o \Delta t_o = \Delta t_p$.

2.4. Parallel Numerical Implementations. Large scale simulations, such as those resulting from fine spatial discretization of a tissue, are computationally expensive. For example, when an 8 μm discretization is used in a 1 cm $\times$ 1 cm tissue and the Bondarenko et al. model (BDK), which has 41 differential variables, is used as cardiac cell model, a total of $1250 \times 1250 \times 41 = 64{,}062{,}500$ unknowns must be computed at each time step. In addition, to simulate 100 ms of cardiac electrical activity, 64 millions of unknowns of the nonlinear systems of ODEs must be computed one million times (with $\Delta t_o = 0.0001$ ms) and the PDE with 1.5 million of unknowns must be computed ten thousand times.

To deal with this high computational cost, two distinct tools for parallel computing were used together: MPI and GPGPU.

2.4.1. Cluster Implementation. The *cluster* implementation is a parallel implementation tailored to cluster of CPUs. The *cluster* implementation uses the PETSc [39] and MPI [40] libraries. It uses a parallel conjugate gradient preconditioned with ILU(0) (with block Jacobi in parallel) to solve the linear system associated with the discretization of the PDE of the monodomain model. More details about this parallel implementation can be found in our previous works on this topic [21–24].

To solve the non-linear systems of ODEs, the explicit Euler method was used. This is an embarrassingly parallel problem. No dependency exists between the solutions of the different systems of ODEs of each finite volume $\text{Vol}_{i,j}$. Therefore, it is quite simple to implement a parallel version of the code: each MPI process is responsible for computing a fraction Np of the total number of volumes of the simulation, where Np is the number of processes involved in the computation.

2.4.2. Multi-GPU Implementation. In our *multi-GPU* implementation, we have decided to keep the cluster approach for the solution of the linear system associated with the discretization of the PDE of the monodomain model. Therefore, *multi-GPU* also solves the discretized PDE with the parallel conjugate gradient preconditioned with ILU(0) (with block Jacobi in parallel) available in the PETSc library.

However, we have accelerated the solution of the systems of ODEs by using multiple GPUs. This is a different strategy from those we have used before when the full Bidomain equations (elliptic PDE, parabolic PDE, and systems of ODEs) were completely implemented in a single GPU [30], or the full Monodomain equations (parabolic PDE and system of ODEs) were completely implemented in a single GPU [25, 26, 29].

The motivation for choosing a different strategy is based on several reasons. As presented in [29], the monodomain model can be accelerated using a single GPU by 35-fold when compared to a parallel OpenMP [41] implementation running on a quad-core computer. However, this final speedup obtained by the GPU comes from a near 10-fold speedup for the solution of the PDE and a near 450-fold speedup for the solution of the nonlinear systems of ODEs. Nowadays, as manycore architecture evolves, one may easily find in the market a single computer equipped with 64 processing cores. Therefore, we believe that solving the PDE on these new machines with traditional MPI or OpenMP-based parallel implementations may outperform a single GPU implementation. On the other side, for the parallel solution of the nonlinear systems of ODEs a single GPU still easily outperforms these new manycore-based computers. This bring, us to focus GPU implementations to the parallel solutions of the millions of nonlinear systems of ODEs. A second motivation is related to the preconditioners that can be easily and efficiently implemented for the conjugate gradient method in GPUs. For the bidomain equations, efficient geometric multigrid preconditioners [30] were implemented in a single GPU, and sophisticated algebraic multigrid preconditioners [42] were implemented in a multi-GPU platform. However, both implementations are only viable for the solution of the linear system associated with the elliptic PDE of the bidomain equations. Multigrid preconditioners are too expensive and turns out to be an inefficient option for the solution of the parabolic PDE, which is the PDE type of the monodomain model. Until now, the cheap but inefficient w-Jacobi preconditioner has been the best choice for GPU implementations when it concerns the solution of the parabolic PDE [29, 42]. However, it is well known that incomplete LU (ILU) preconditioners combined with block Jacobi or additive Schwarz domain decomposition methods [23] greatly outperform Jacobi-like preconditioners on cluster computing for the solution of the PDE of the monodomain model. This argument favors cluster-like implementations as the best choice for the parallel solution of the parabolic PDE of the monodomain model (see [43] and the references cited therein). Finally, a third and last motivation is related to the particular problem we propose to investigate in this work: models that reveal the microstructure of cardiac tissue. Another recent work presented an implementation for the bidomain model for multi-GPU platforms [44]. Both PDEs and systems of ODEs were implemented on GPUs using explicit methods, Jacobi relaxation, and explicit Euler, respectively. We note that for our particular microscopic tissue model with spatial discretization of 8 μm, the approach of using an explicit and cheap solver for the PDE would be very inefficient due to the severe stability restrictions imposed by the CFL conditions [37]. Therefore, once more, this argument also favors cluster-like implementations based on implicit methods for the parallel solution of the parabolic PDE of the monodomain model.

Our *multi-GPU* implementation uses CUDA [45] to implement the numerical solution of the BDK cardiac cell model. The CUDA model extends the C programming language with a set of abstractions to express parallelism, that is, CUDA includes C software development tools and libraries to hide the GPGPU hardware details from programmers that can focus on important issues of the parallelism of their code rather than dealing with unfamiliar and complicated concepts from computer graphics in order to explore the computational power of GPUs for general purpose computation.

In order to run an application, the programmer must create a parallel function called kernel. A kernel is a special C function callable from the CPU but executed on the GPU simultaneously by many threads. Each thread is run by a GPU stream processor. They are grouped into blocks of threads or just blocks. The blocks can be one-, two-, or three-dimensional. A set of blocks of threads form a grid, that can be one- or two-dimensional. When the CPU calls the kernel, it must specify how many blocks and threads will be created at the GPU to execute the kernel. The syntax that specifies the number of threads that will be created to execute a kernel is formally known as the execution configuration and is flexible

to support CUDA's hierarchy of threads, blocks of threads, and grids of blocks. Since all threads in a grid execute the same code, a unique set of identification numbers is used to distinguish threads and to define the appropriate portion of the data they must process. These threads are organized into a two-level hierarchy composed by blocks and grids and two unique coordinates, called *blockId* and *threadId*, are assigned to them by the CUDA runtime system. These two built-in variables can be accessed within the kernel functions and they return the appropriate values that identify a block and thread, respectively. All the threads within a single block are allowed to synchronize with each other via a special barrier operator, called *syncthread*, and have access to a high-speed, per-block shared memory which allows interthread communication. Threads from different blocks in the same grid can coordinate their execution only through the use of atomic global memory operations. No assumptions are made about the execution order of thread blocks, which means that a kernel must execute correctly no matter the order in which blocks are scheduled by the hardware to run.

Some additional steps must be followed to use the GPU: (a) the device must be initialized; (b) memory must be allocated in the GPU and data transferred to it; (c) the kernel is then called. After the kernel have finished its execution, results are transferred back to the CPU.

Two kernels have been developed to solve each of the systems of ODEs related to BDK model. The first kernel is responsible for setting the initial conditions of the systems of ODEs, whereas the second one integrates the systems of ODEs at each time step.

Both kernel implementations were optimized in many different ways. The state variables of M cardiac cells were stored in an array called SV, whose size is equal to MN_{eq}, where N_{eq} is the number of differential equations of the ionic model (in this work, N_{eq} is equal to 41). The SV array was organized in such a way that the first M entries correspond to the first state variable, followed by M entries of the next state variable, and so on. Moreover, for all ionic models, the first M entries of the SV array correspond to the transmembrane potential V. During the solution of the systems of PDEs, after the integration of the ODEs systems, the transmembrane potential of each node should be passed to the PETSC solver. Due to the memory organization chosen for the SV array, this is a straightforward task since, as stated before, the M first entries of the array correspond to the transmembrane potential V of each node. This organization allows us to avoid extra memory transactions between CPU and GPU, improving performance. Another implementation choice that impact performance positively was the way the SV array has been allocated. The SV array was allocated in global GPU memory using the *cudaMallocPitch* routine from the CUDA API. This routine may pad the allocation in order to ensure that corresponding memory addresses of any given row will continue to meet the alignment requirements for the coalescing operations performed by the hardware. In short, a strict coalescing requires that thread j out of n threads has to access data $u[j]$ if $u[0]$ is accessed by thread 0, that is, each thread should perform data access by stride n. Therefore, in the first kernel, to set the initial conditions, each thread sets the values of all its state variables. The kernel that solves the system of ODEs operates similarly, that is, each thread computes and updates its state variables writing to the right position in memory that corresponds to their variables. In addition, the second kernel was optimized to use as much as local memory operations as possible.

Pure domain decomposition was used for parallelism. The tissue domain was linearly decomposed on Np nonoverlapping subdomains (or Np tasks, T_1 to T_{Np}, see Figure 4), where Np is the number of MPI processes or processing cores. The parallel solution of the PDE is implemented via PETSc (see [21]), with each processing core p responsible for updating the variables associated to subdomain T_p. In our computational environment each machine or node has more CPU cores (8) than GPUs (2). Therefore, for the solution of the ODEs each GPU device will be responsible for processing more than one task. The tasks assigned with one node are distributed to the GPUs in a round-robin fashion. For example, if $Np = 16$ and we have two machines (each with 8 cores and 2 GPU devices), Figure 4 presents how the tissue domain will be partitioned. Four tasks would be assigned to each GPU device. For instance, at node 0, GPU 0 would process tasks T_1, T_3, T_5, and T_7, GPU 1 the tasks T_2, T_4, T_6, and T_8.

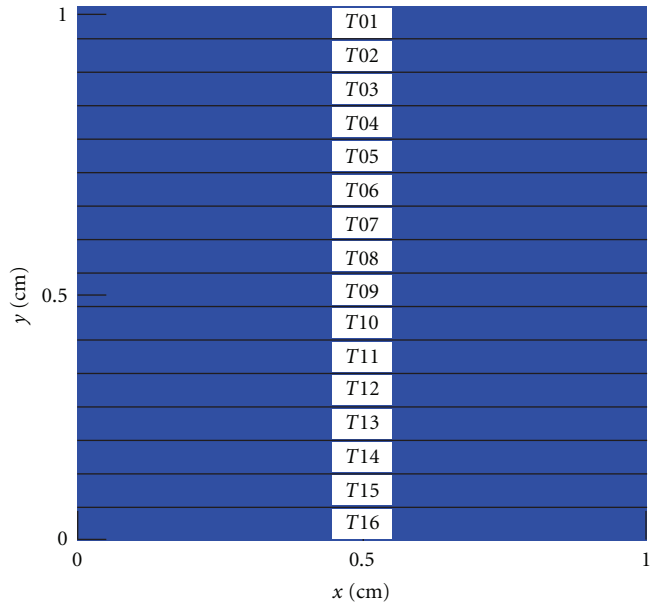
For the solution of the ODEs, both sequential and parallel (CUDA) codes used single precision. For the solution of the PDE we have used double precision. For the case of monodomain simulations, we have shown in [25] that the use of single precision in CUDA does not affect the numerical precision of the solver.

3. In Silico Experiments, Computational Environment, and Metrics

The simulations were performed using the microscopic model with spatial discretization of 8 μm and heterogeneous conductivity values as described in Section 2.1. The values used for β and C_m were set to 0.14 cm^{-1} and 1.0 μF/cm^2, respectively. The time step used to solve the linear system associated with (17) was set to $\Delta t_p = 0.01$ ms and to solve the nonlinear system of ODEs associated to (18) was set with $\Delta t_o = 0.0001$ ms.

Three different tissue setups were used to test our model and parallel implementations: a cardiac tissue of 0.5 cm $\times$ 0.5 cm size that was stimulated in the center and was executed for 10 ms, a cardiac tissue of 1.0 cm $\times$ 1.0 cm size that was stimulated in the center and was executed for 10 ms, and a cardiac tissue of 1.0 cm $\times$ 1.0 cm that was stimulated using the S1-S2 protocol to generate a spiral wave, a form of self-sustained reentrant activity strongly associated with cardiac arrhythmia.

Our experiments were performed on a cluster of 8 SMP computers. Each computer contains two Intel E5620 Xeon quad-core processors and 12 GB of RAM. All nodes run Linux version 2.6.18 – 194.17.4.el5. The codes were compiled with gcc 4.1.2 and CUDA 3.2. Each node contains two Tesla C1060. The Tesla C1060 card has 240 CUDA cores and 4 GB of global memory.



Linear parallel partitioning

	Node 0				Node 1			
PDE solve	Core 0	Core 1	Core 2	Core 3	Core 0	Core 1	Core 2	Core 3
	$T01$	$T02$	$T03$	$T04$	$T09$	$T10$	$T11$	$T12$
	Core 4	Core 5	Core 6	Core 7	Core 4	Core 5	Core 6	Core 7
	$T05$	$T06$	$T07$	$T08$	$T13$	$T14$	$T15$	$T16$
ODE solve	GPU 0		GPU 1		GPU 0		GPU 1	
	$T01 + T03 + T05 + T07$		$T02 + T04 + T06 + T08$		$T09 + T11 + T13 + T15$		$T10 + T12 + T14 + T16$	

Figure 4: Linear parallel decomposition of tissue. Example for the case of two nodes (each with 8 cores and 2 GPU devices, that is, a total of 16 CPU cores and 4 GPU devices). Each CPU core processes one task. Four tasks are assigned to each GPU device. For instance, at node 0, GPU 0 processes tasks T_1, T_3, T_5, and T_7, and GPU 1 the tasks T_2, T_4, T_6, and T_8.

All tests were performed three times. The average execution time, in seconds, is then used to calculate the speedup, defined as the sequential execution time divided by the parallel execution time.

4. Results

Figure 5 presents the propagation of a central stimulus on the tissue of size 1 cm × 1 cm for different time instants. As expected, macroscopically, the propagation looks very smooth and continuous. However, when highlighting a smaller region of size 1 mm × 1 mm, see Figure 6, we can already observe the discrete nature of propagation, that is, the influence of the cardiac microstructure on the propagation of action potentials.

Table 1 presents the results obtained by the parallel implementations for the experiment with a square tissue of 0.5 cm × 0.5 cm. As one can observe, the time spent solving the ODEs is responsible for near 90% of the execution time. It can also be observed that although the obtained speedups with the *cluster* are respectable and almost linear (near 61 with 64 cores), the total execution time remains high. With respect to the *multi-GPU* implementation, the results are much better. It must be stressed that although 64 cores where used in this simulation, only 16 GPGPU devices where available for executing the simulation, so 8 processes share 2 GPGPU devices per machine. As one can observe,

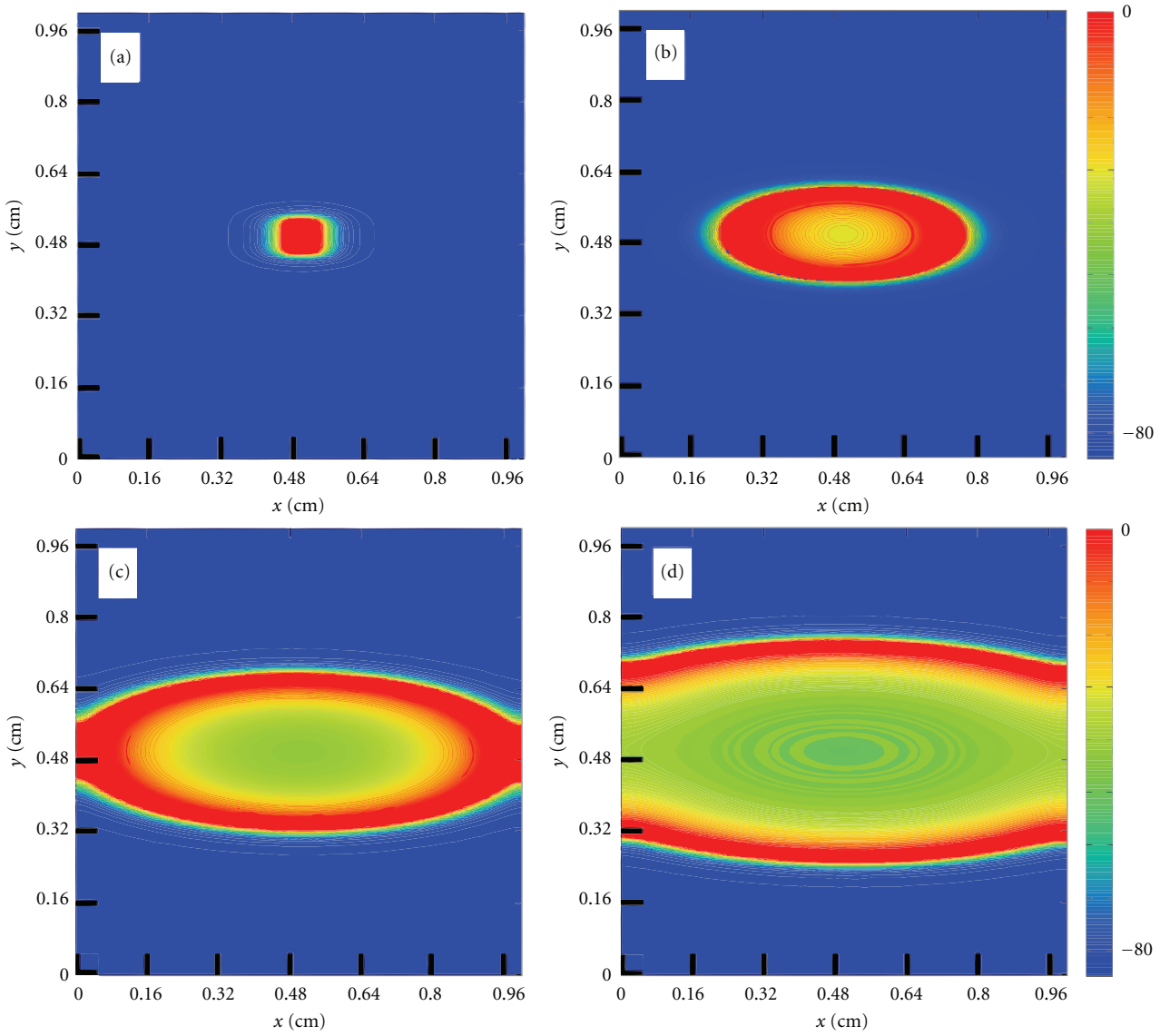


Figure 5: Action potential propagation (transmembrane potential) after a central stimulus on a tissue of size 1 cm × 1 cm for different time instants. (a) $t = 1$ ms, (b) $t = 7$ ms, (c) $t = 13$ ms, and (d) $t = 20$ ms.

the obtained speedup was huge, about 343 times faster than a single core processor. The execution time drops from 1.6 days (using one processing core) to only 6.7 minutes (using the 8-node multi-GPU platform).

Table 2 presents the results obtained by the parallel implementations for the experiment with a square tissue of 1.0 cm × 1.0 cm. Once again, the speedups obtained with the *cluster* implementation, were almost linear (61 with 64 cores). With respect to the *multi-GPU* implementation the results are much better. The speedup was huge, about 420 times faster than a single core processor. The execution time drops from more than 6 days (using one processing core) to only 21 minutes (using the 8-node multi-GPU platform). We can also observe that the *multi-GPU* implementation was near 7 times faster than the *cluster* implementation when running on the 8 computers.

As a result, using this very fast parallel implementation, we were able to simulate the formation of spiral waves, a form of self-sustained reentrant activity strongly associated with cardiac arrhythmia, see Figure 7. To the best of our knowledge, this is the first time spiral waves are simulated using a cardiac model that accounts for both the microstructure of cardiac tissue and a modern and complex myocyte model. After a couple of tries using the S1-S2 protocol to find the correct vulnerable window, we managed to generate a sustained spiral wave using this cardiac model that accounts for both the microstructure of cardiac tissue and a modern and complex myocyte model. The whole process took less than one day (around 13 hours with each simulation taking between 3 and 7 hours). Without our multi-GPU parallel implementation, this process would have taken 227 days using a single core computer or near 4 days using our

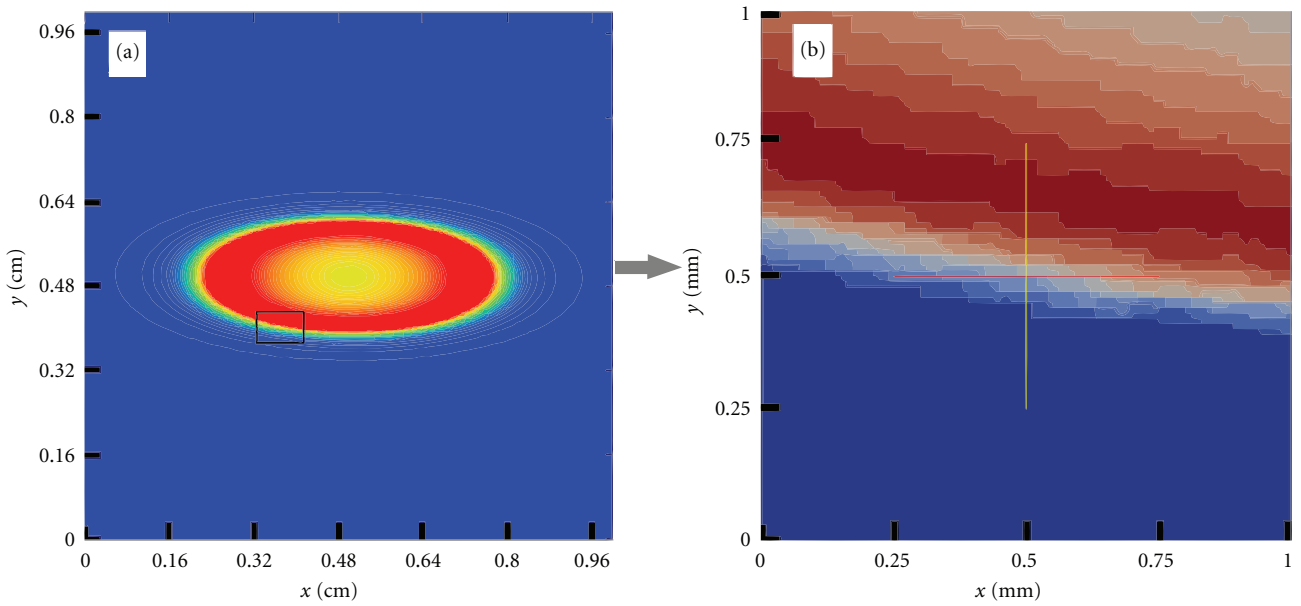


Figure 6: (a) Transmembrane potential at $t = 7$ ms after a central stimulus on a tissue of size 1 cm $\times$ 1 cm. (b) Microscopic details revealing the discrete nature of AP propagation, that is, the influence of cardiac microstructure on a large tissue size simulation.

Table 1: Average execution time and speedup of parallel implementations for a tissue of 0.5 cm $\times$ 0.5 cm. The execution times are presented in seconds.

Parallel implementation	Cores	Total Time	ODE time	PDE time	Speedup
Cluster	1	137,964	132,590	5,264	—
Cluster	8	18,492	17,210	1,262	7.5
Cluster	16	9,922	9,316	595	13.9
Cluster	32	4,198	3,884	311	32.9
Cluster	64	2,283	2,087	191	60.4
Multi-GPU	64 + 16 GPUs	401.84	209.4	187	343

Table 2: Average execution time and speedup of parallel implementations for a tissue of 1.0 cm $\times$ 1.0 cm. The execution times are presented in seconds.

Parallel implementation	Cores	Total Time	ODE time	PDE time	Speedup
Cluster	1	546,507	523,331	23,177	—
Cluster	64	8,934	8,313	607	61.2
Multi-GPU	64 + 16 GPUs	1,302	682	611	420

cluster implementation running with 64 cores but without the GPUs.

5. Discussion and Future Works

Our results show that our *multi-GPU* parallel implementation described in Section 2.4.2 was able to significantly accelerate the numerical solution of a cardiac electrophysiology model that captures the microstructure of cardiac tissue (using a very fine spatial discretization) and is based on a very modern and complex cell model (with Markov chain formulation that has been extensively used for the characterization of ion channels). Speedups around 420 times were obtained, reducing execution times from more than 6 days (using one processing core) to only 21 minutes (using the 8-node multi-GPU platform). The hybrid *Multi-GPU* parallel implementation presented in this work is even more attractive if one considers that the architectures of GPUs and multicore processors continue to evolve on a fast pace.

Nevertheless, we believe our parallel implementation can be further improved. For instance, in the current implementation, the CPU cores are idle while waiting for the results of the nonlinear ODEs that are being computed by the GPU devices. For future work, we intend to evaluate different load balancing techniques to better distribute the parallel tasks between GPU devices and CPU cores and make a more efficient use of all the computational resources. Another possible improvement is related to the multilevel

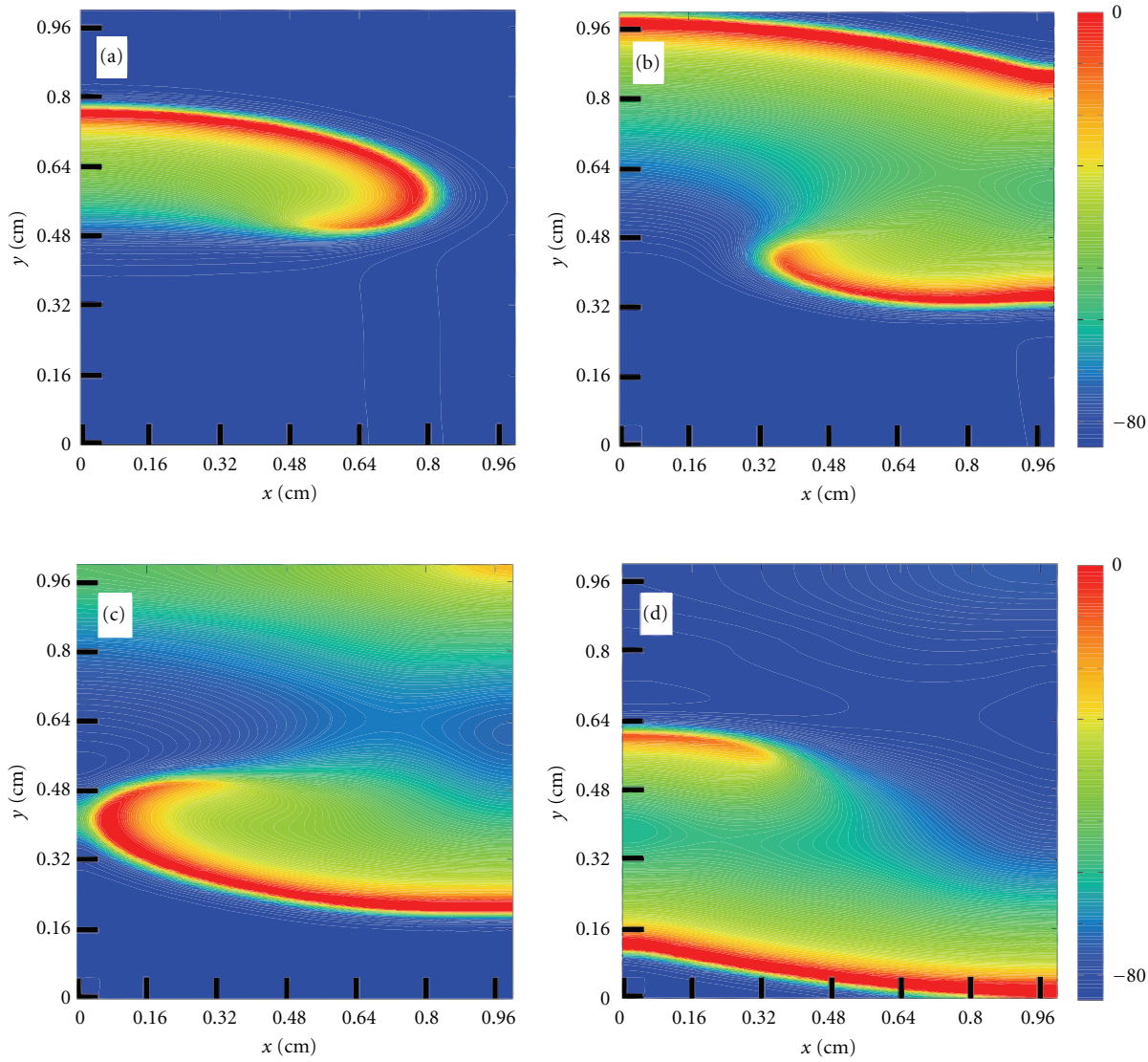


Figure 7: Spiral wave formation after an S1-S2 stimulus protocol at different time instants: (a) $t = 80$ ms, (b) $t = 100$ ms, (c) $t = 112$ ms, and (d) $t = 120$ ms.

parallelism introduced for the solution of the bidomain equations [24] that combines task parallelism (via pipeline) and data parallelism (via data decomposition). We believe a similar combination of data and task parallelism could be also exploited for the solution of the monodomain equations to further enhance the parallel efficiency of our algorithms.

Recent studies that focus on the discrete or discontinuous nature of AP propagation have avoided the computational challenges that arise from microscopic models via the development and use of discrete models, where each cardiac myocyte is represented by a single point connected with the neighboring myocytes by different conductivities [46, 47]. This description has allowed the study of the effects of randomly distributed conductivities in the conduction velocity and on the formation of reentry patterns on cardiac tissue. Discrete models were introduced by Keener in [48] to describe the electrical propagation in a 1D cable of nc connected cells for the case of low gap-junctional coupling. In this model, the cells are assumed to be isopotential. Therefore, only gap junction conductances are considered for the connection of neighboring myocytes, that is, cytoplasmic resistance is considered to be insignificant. Recently, we have compared discrete and microscopic models for a 1D cable of connected cells [49]. We have shown that the numerical results obtained by the discrete model are similar to those obtained by the heterogeneous microscopic model for the case of low gap-junctional coupling (1%–10% of normal coupling). However, the discrete model failed for the case of normal gap-junctional coupling or moderate reduced gap-junctional coupling (50%–100% of normal coupling). The two-dimensional microscopic model developed in this work will allow us to further compare these two approaches (detailed microscopic models versus discrete models) and to better understand the benefits and limitations of each one of

them. In addition, we hope that our microscopic model may also suggest ways to better develop discrete models, which are computationally less expensive than the detailed microscopic ones.

6. Conclusion

In this paper, we developed a cardiac electrophysiology model that captures the microstructure of cardiac tissue by using a very fine spatial discretization and uses a very modern and complex cell model based on Markov chains for the characterization of ion channel's structure and dynamics. To cope with the computational challenges, the model was parallelized using a hybrid approach: cluster computing and GPGPUs. Different *in silico* tissue preparations were used in this work for the performance tests. We have shown that in all cases, our parallel multi-GPU implementation was able to significantly reduce the execution times of the simulations, for instance, from more than 6 days (on a single processor) to 21 minutes (on a small 8-node cluster equipped with 16 GPUs, that is, 2 GPUs per node). We believe that this new parallel implementation paves the way for the investigation of many open questions associated with the complex and discrete propagation nature of action potentials on cardiac tissue.

Authors' Contribution

B. G. de Barros and R. S. Oliveira contributed equivalently in this paper. "Therefore they can be both considered as first authors appearing in alphabetical order".

Acknowledgments

The authors would like to thank CAPES, CNPq, FAPEMIG, FINEP, UFMG, UFSJ, and UFJF for supporting this work.

References

[1] WHO, World health organization, 2010, http://www.who.int/.

[2] F. B. Sachse, *Computational Cardiology: Modeling of Anatomy, Electrophysiology, and Mechanics*, vol. 2966, Springer, 2004.

[3] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[4] R. Plonsey, "Bioelectric sources arising in excitable fibers (alza lecture)," *Annals of Biomedical Engineering*, vol. 16, no. 6, pp. 519–546, 1988.

[5] R. R. Aliev and A. V. Panfilov, "A simple two-variable model of cardiac excitation," *Chaos, Solitons and Fractals*, vol. 7, no. 3, pp. 293–301, 1996.

[6] V. E. Bondarenko, G. P. Szigeti, G. C. L. Bett, S. J. Kim, and R. L. Rasmusson, "Computer model of action potential of mouse ventricular myocytes," *American Journal of Physiology*, vol. 287, no. 3, pp. H1378–H1403, 2004.

[7] C. E. Clancy and Y. Rudy, "Linking a genetic defect to its cellular phenotype in a cardiac arrhythmia," *Nature*, vol. 400, no. 6744, pp. 566–569, 1999.

[8] T. Brennan, M. Fink, and B. Rodriguez, "Multiscale modelling of drug-induced effects on cardiac electrophysiological activity," *European Journal of Pharmaceutical Sciences*, vol. 36, no. 1, pp. 62–77, 2009.

[9] C. E. Clancy, Z. I. Zhu, and Y. Rudy, "Pharmacogenetics and anti-arrhythmic drug therapy: a theoretical investigation," *American Journal of Physiology*, vol. 292, no. 1, pp. H66–H75, 2007.

[10] V. Iyer, R. Mazhari, and R. L. Winslow, "A computational model of the human left-ventricular epicardial myocyte," *Biophysical Journal*, vol. 87, no. 3, pp. 1507–1525, 2004.

[11] M. S. Spach and J. F. Heidlage, "The stochastic nature of cardiac propagation at a microscopic level: electrical description of myocardial architecture and its application to conduction," *Circulation Research*, vol. 76, no. 3, pp. 366–380, 1995.

[12] V. Jacquemet and C. S. Henriquez, "Loading effect of fibroblast-myocyte coupling on resting potential, impulse propagation, and repolarization: insights from a microstructure model," *American Journal of Physiology*, vol. 294, no. 5, pp. H2040–H2052, 2008.

[13] M. L. Hubbard, W. Ying, and C. S. Henriquez, "Effect of gap junction distribution on impulse propagation in a monolayer of myocytes: a model study," *Europace*, vol. 9, supplement 6, pp. vi20–vi28, 2007.

[14] A. G. Kléber and Y. Rudy, "Basic mechanisms of cardiac impulse propagation and associated arrhythmias," *Physiological Reviews*, vol. 84, no. 2, pp. 431–488, 2004.

[15] H. J. Jongsma and R. Wilders, "Gap junctions in cardiovascular disease," *Circulation Research*, vol. 86, no. 12, pp. 1193–1197, 2000.

[16] Y. Wang and Y. Rudy, "Action potential propagation in inhomogeneous cardiac tissue: safety factor considerations and ionic mechanism," *American Journal of Physiology*, vol. 278, no. 4, pp. H1019–H1029, 2000.

[17] M. S. Spach and R. C. Barr, "Effects of cardiac microstructure on propagating electrical waveforms," *Circulation Research*, vol. 86, no. 2, pp. e23–e28, 2000.

[18] R. M. Shaw and Y. Rudy, "Ionic mechanisms of propagation in cardiac tissue: roles of the sodium and L-type calcium currents during reduced excitability and decreased gap junction coupling," *Circulation Research*, vol. 81, no. 5, pp. 727–741, 1997.

[19] J. Stinstra, R. MacLeod, and C. Henriquez, "Incorporating histology into a 3D microscopic computer model of myocardium to study propagation at a cellular level," *Annals of Biomedical Engineering*, vol. 38, no. 4, pp. 1399–1414, 2010.

[20] S. F. Roberts, J. G. Stinstra, and C. S. Henriquez, "Effect of nonuniform interstitial space properties on impulse propagation: a discrete multidomain model," *Biophysical Journal*, vol. 95, no. 8, pp. 3724–3737, 2008.

[21] R. Weber Dos Santos, G. Plank, S. Bauer, and E. J. Vigmond, "Parallel multigrid preconditioner for the cardiac bidomain model," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 11, pp. 1960–1968, 2004.

[22] G. Plank, M. Liebmann, R. W. dos Santos, E. J. Vigmond, and G. Haase, "Algebraic multigrid preconditioner for the cardiac bidomain model," *IEEE Transactions on Biomedical Engineering*, vol. 54, pp. 585–596, 2007.

[23] R. W. dos Santos, G. Plank, S. Bauer, and E. J. Vigmond, "Preconditioning techniques for the bidomain equations," *Lecture Notes in Computational Science and Engineering*, vol. 40, pp. 571–580, 2004.

[24] C. R. Xavier, R. S. Oliveira, V. Da Fonseca Vieira, R. W. Dos Santos, and W. Meira, "Multi-level parallelism for the cardiac

bidomain equations," *International Journal of Parallel Programming*, vol. 37, no. 6, pp. 572–592, 2009.

[25] B. M. Rocha, F. O. Campos, R. M. Amorim et al., "Accelerating cardiac excitation spread simulations using graphics processing units," *Concurrency Computation Practice and Experience*, vol. 23, no. 7, pp. 708–720, 2011.

[26] B. M. Rocha, F. O. Campos, G. Plank, R. W. dos Santos, and M. Liebmann, "Simulations of the electrical activity in the heart with graphic processing units," in *Parallel Processing and Applied Mathematics*, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski, Eds., vol. 6067, pp. 439–448, Lecture Notes in Computer ScienceSpringer, Berlin, Germany, 2010.

[27] R. M. Amorim, B. M. Rocha, F. O. Campos, and R. W. Dos Santos, "Automatic code generation for solvers of cardiac cellular membrane dynamics in GPUs," in *Proceedings of the 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC '10)*, pp. 2666–2669, September 2010.

[28] R. Amorim, G. Haase, M. Liebmann, and R. W. D. Santos, "Comparing CUDA and OpenGL implementations for a Jacobi iteration," in *Proceedings of the International Conference on High Performance Computing and Simulation (HPCS '09)*, pp. 22–32, June 2009.

[29] R. S. Oliveira, B. M. Rocha, R. M. Amorim et al., "Comparing cuda, opencl and opengl implementations of the cardiac monodomain equations," in *Parallel Processing and Applied Mathematics*, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski, Eds., vol. 7204 of *Lecture Notes in Computer Science*, pp. 111–120, Springer, Berlin, Germany, 2012.

[30] R. M. Amorim and R. W. dos Santos, "Solving the cardiac bidomain equations using graphics processing units," *Journal of Computational Science*. In press.

[31] A. M. Gerdes, S. E. Kellerman, J. A. Moore et al., "Structural remodeling of cardiac myocytes in patients with ischemic cardiomyopathy," *Circulation*, vol. 86, no. 2, pp. 426–430, 1992.

[32] R. E. Tracy and G. E. Sander, "Histologically measured cardiomyocyte hypertrophy correlates with body height as strongly as with body mass index," *Cardiology Research and Practice*, vol. 2011, Article ID 658958, 2011.

[33] R. Eymard, T. Gallouët, and R. Herbin, "Finite volume methods," *Handbook of Numerical Analysis*, vol. 7, pp. 713–1018, 2000.

[34] D. M. Harrild and C. S. Henriquez, "A finite volume model of cardiac propagation," *Annals of Biomedical Engineering*, vol. 25, no. 2, pp. 315–334, 1997.

[35] Y. Coudiere, C. Pierre, and R. Turpault, "A 2d/3d finite volume method used to solve the bidomain equations of electrocardiology," in *Proceedings of the Algoritmy*, pp. 1–10, 2009.

[36] J. Sundnes, *Computing the Electrical Activity in the Heart*, Springer, 2006.

[37] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Society for Industrial Mathematics, 2004.

[38] S. Rush and H. Larsen, "A practical algorithm for solving dynamic membrane equations," *IEEE Transactions on Biomedical Engineering*, vol. 25, no. 4, pp. 389–392, 1978.

[39] S. Balay, K. Buschelman, V. Eijkhout et al., "PETSc users manual," Tech. Rep., Citeseer, 2004.

[40] W. Groop and E. Lusk, "User's guide for mpich, a portable implementation of MPI," Tech. Rep., Argonne National Laboratory, 1994.

[41] L. Dagum and R. Menon, "Openmp: an industry standard api for shared-memory programming," *IEEE Computational Science and Engineering*, vol. 5, no. 1, pp. 46–55, 1998.

[42] A. Neic, M. Liebmann, E. Hoetzl et al., "Accelerating cardiac bidomain simulations using graphics processing units," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 8, pp. 2281–2290, 2012.

[43] E. J. Vigmond, R. Weber dos Santos, A. J. Prassl, M. Deo, and G. Plank, "Solvers for the cardiac bidomain equations," *Progress in Biophysics and Molecular Biology*, vol. 96, no. 1–3, pp. 3–18, 2008.

[44] V. K. Nimmagadda, A. Akoglu, S. Hariri, and T. Moukabary, "Cardiac simulation on multi-GPU platform," *Journal of Supercomputing*, vol. 59, no. 3, pp. 1360–1378, 2012.

[45] D. B. Kirk and W. W. Hwu, *Massively Parallel Processors: A Hands-on Approach*, Morgan Kaufmann, 2010.

[46] S. Alonso, M. Bär, and A. V. Panfilov, "Effects of reduced discrete coupling on filament tension in excitable media," *Chaos*, vol. 21, no. 1, Article ID 013118, 2011.

[47] S. Alonso, M. Bar, and A. V. Panfilov, "Negative tension of scroll wave filaments and turbulence in three-dimensional excitable media and application in cardiac dynamics," *Bulletin of Mathematical Biology*. In press.

[48] J. P. Keener, "The effects of gap junctions on propagation in myocardium: a modified cable theory," *Annals of the New York Academy of Sciences*, vol. 591, pp. 257–277, 1990.

[49] C. M. Costa and R. W. Dos Santos, "Limitations of the homogenized cardiac Monodomain model for the case of low gap junctional coupling," in *Proceedings of the 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC '10)*, pp. 228–231, September 2010.