Journal of Computational Science xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Journal of Computational Science

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

Solving the cardiac bidomain equations using graphics processing units

Ronan Mendonça Amorim, Rodrigo Weber dos Santos*

Graduate Program in Computational Modeling, University of Juiz de Fora, Brazil

ARTICLE INFO

Article history:
Received 23 December 2011
Received in revised form 22 June 2012
Accepted 28 June 2012
Available online xxx

Keywords:
Cardiac modeling
Bidomain equations
Graphics processing units
Preconditioned conjugate gradient
Multigrid method

ABSTRACT

The computational modeling of the heart has been shown to be a very useful tool. The models, which become more realistic each day, provide a better understanding of the complex biophysical processes related to the electrical activity in the heart, e.g., in the case of cardiac arrhythmias. However, the increasing complexity of the models challenges high performance computing in many aspects. This work presents a cardiac simulator based on the bidomain equations that exploits the new parallel architecture of graphics processing units (GPUs). The initial results are promising. The use of the GPU accelerates the cardiac simulator by about 6 times compared to the best performance obtained in a general-purpose processor (CPU). In addition, the GPU implementation was compared to an efficient parallel implementation developed for cluster computing. A single desktop computer equipped with a GPU is shown to be 1.4 times faster than the parallel implementation of the bidomain equations running on a cluster composed of 16 processing cores.

1. Introduction

Modern cell models of cardiac electrophysiology are described by nonlinear systems of differential equations with tens of variables and nearly hundreds of parameters whereas the number of variables explodes to the millions or tens of millions in cardiac tissue or whole organ models. The set of bidomain equations [1] is currently the most complete mathematical model for describing the spread of cardiac electrical activity and is especially suited to simulate activity on the organ level. The complexity associated with cardiac modeling demands high performance computing and efficient numerical methods. Several efficient numerical methods have been proposed to tackle this problem. For instance, we refer the reader to the performance evaluation of multigrid-based methods reviewed in [2] and to the recent work on mesh adaptivity presented in [3]. A common feature of these works is that the proposed numerical methods are tailored to parallel environments based on clusters of computers. Thus far, this combination of sophisticated numerical methods and cluster computing has been shown to be one of the most efficient ways to solve the cardiac bidomain equations.

Recently, new parallel architectures have emerged. In particular, as the performance of modern graphics hardware increases and becomes more flexible in terms of programmability, many researchers are applying this new technology to problems previously solved with general-purpose processors (CPUs) or with clusters of computers [4]. Some recent papers have evaluated the new computer architecture based on graphics processing units (GPUs) for the solution of cardiac models. In [5,6], two different implementations that exploit the new architecture of GPUs were presented for the monodomain equations, a simplified model for cardiac electrophysiology. In [7,8], GPUs were used to accelerate the numerical solution of cardiac myocyte models.

In this work, we present a GPU implementation of the full cardiac bidomain equations that uses implicit and sophisticated numerical methods such as multigrid preconditioners. The GPU implementation was compared both in terms of speed and numerical accuracy to an equivalent CPU implementation. In addition, the GPU implementation was compared to an efficient parallel implementation developed for cluster computing [9]. The preliminary results are promising: the use of the GPU accelerated the cardiac simulator by about 6 times compared to the best performance obtained using a CPU. In addition, a single desktop computer equipped with a GPU is shown to be 1.4 times faster than the parallel implementation of the bidomain equations running on a cluster composed of 16 processing cores.

Current GPUs offer limited support for double-precision operations. In this paper we show that the numerical errors associated with single-precision operations of the GPUs may impair the solution of the bidomain equations. Therefore, this work presents the special methods that are needed by the GPU implementation in order to keep the numerical errors under control. Finally, we present a detailed performance analysis that points out the tasks of the numerical algorithms associated to the bidomain equations that deserve better GPU implementations on future works.

* Corresponding author.
E-mail address: rodrigo.weber@yahoo.com (R. Weber dos Santos).

1877-7503/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jocs.2012.06.007

2. Methods

The bidomain equations [1] represent a homogenization of cardiac tissue by replacing the individually coupled cells with a syncytium. Each point in space is considered to exist in two domains, the intra- and extracellular domains, which are superimposed on one another. Any discrete objects are averaged over the volume. The equations model how cellular electric potentials (transmembrane voltage, V_m, and extracellular potential, ϕ_e or V_e) depend on the concentrations of several ionic species and on ionic currents (I_{ion}) that cross the cell membrane. The bidomain equations may be casted in different ways [10], in this work we write the equations as follows:

$$\nabla \cdot (\overline{\sigma_i} + \overline{\sigma_e}) \nabla V_e = -\nabla \cdot \overline{\sigma_i} \nabla V_m \quad (1)$$

$$\nabla \cdot \overline{\sigma_i} \nabla V_m = -\nabla \cdot \overline{\sigma_i} \nabla V_e + \beta I_m \quad (2)$$

$$I_m = C_m \frac{\partial V_m}{\partial t} + I_{ion}(V_m, \upsilon) \quad (3)$$

where $\overline{\sigma_i}$ and $\overline{\sigma_e}$ are, the intracellular and extracellular conductivity tensors, respectively, i.e., 3×3 symmetric matrices that vary in space and describe the anisotropy of the cardiac tissue; β is the surface to volume ratio of the cardiac cells; C_m is the membrane capacitance per unit area; V_m is the transmembrane voltage; and I_{ion} is the ionic current density flowing through the membrane ionic channels and depends on the transmembrane voltage and several other variables that we represent by υ. The intracellular and extracellular domains are modeled as linear electrostatic media. The nonlinearity arises in the current–voltage relationship across the membrane (Eq. (3)) which is described by a set of nonlinear ordinary differential equations (ODEs). The bidomain equations may be considered as a coupled set of an elliptic partial differential equation (PDE, Eq. (1)), a parabolic PDE (Eq. (2)), and a non-linear system of ODEs (Eq. (3)).

The cardiac simulations performed in this work were based on the Left Ventricular *Wedge* models previously published in [11]. This model includes a 2D discretization of the bidomain equations that considers three different ventricular myocyte models for the epicardium, M cells, and endocardium [12]. The tissue was immersed in an isotropic and homogeneous conducting medium that simulates the experimental bath. The parameters of the tissue-bath model such as the conductivity tensors were identical to those described in [11]. A stimulus current was applied on a selected part of the endocardial region. The space discretization was performed using the finite elements method with square elements and bilinear interpolation. The time discretization for the parabolic equation was achieved by using the Crank–Nicolson method. The space and time discretizations used were 40 μm and 10 μs, respectively. A total of 40 time steps were simulated.

Five different tissues were simulated with a fixed width of 513 mesh points (2 cm) and different heights: 33, 65, 129, 257, 513 and 1025 mesh points. These rectangular setups were based on our previous work that discussed the influence of different phenotypes for ventricular myocyte models (epicardial cells, M cells, and endocardial cells) on the waveform of electrograms (with specific focus on repolarization and electrotonic effect) taken from a Left Ventricular Wedge model [11]. Fig. 1 presents a tissue composed of 513×257 mesh points.

Using an operator splitting scheme, each time step of the numerical method involved the solution of three different tasks: a non-linear system of ODEs, a linear system derived from the discretization of a parabolic PDE, and a linear system derived from the discretization of an elliptic PDE. The preconditioned conjugate gradient method (PCG) was used to solve the linear systems derived from the discretization of the PDEs. The PCG convergence condition was set to an absolute tolerance of 10^{-6}. We used three different

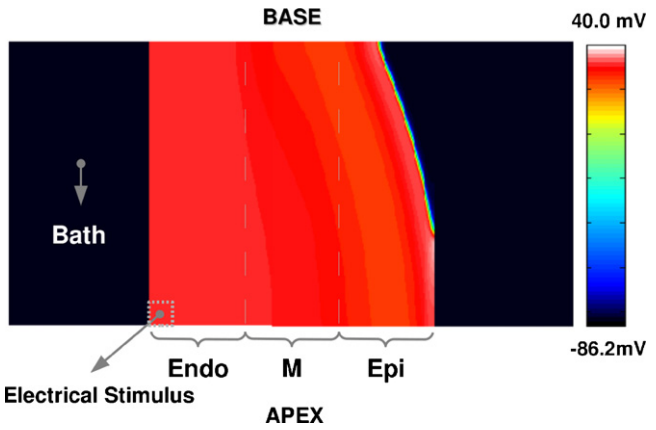

Fig. 1. Tissue with 513×257 mesh points simulated with an current stimulus applied on the endocardial site. The tissue is placed in a bath.

preconditioners combined with the PCG algorithm: block Jacobi, block ILU(0) (incomplete LU factorization), and geometric multigrid [13]. The multigrid method consists of three basic operations: relaxation by using an iterative solver, restriction of the residual to a coarser grid, and interpolation of the solution to a finer grid.

In this work, we compared three different implementations of the bidomain equations: a GPU implementation, **GPU**; a CPU implementations that is equivalent to the GPU one, **CPU**; and a parallel CPU implementation, **Cluster**.

The **GPU** implementation used the explicit Euler method to solve the nonlinear ODEs system, a conjugate gradient preconditioned with Jacobi to solve the linear system from the parabolic equation, and a conjugate gradient preconditioned with geometric multigrid to solve the linear system from the elliptic equation. The multigrid preconditioner used ω-Jacobi as the relaxation method with $\omega = 0.8$. The number of steps used, i.e., the number of Jacobi sweeps per grid level as part of the MG preconditioner, was 2 (level + 1) where level = 0 for the coarsest grid. On the coarsest level, the 'exact' solution was given by the conjugate gradient method with an absolute tolerance of 10^{-15}. The number of grid levels used was 6, 7, 8, 9, 10 and 10 for the tissues with dimensions of 513×33, 513×65, 513×128, 513×257, 513×513 and 513×1025, respectively. The values of these parameters were found empirically by testing a large set of values: ω was tested with values from 0.05 to 0.90 in steps of 0.05; the number of ω-Jacobi iterations uses the formula n (level + 1) and n was tested with the values 1, 2 and 3; the number of grid levels experimented were 6, 7, 8, 9 and 10; and the tolerance for the convergence test used at the coarsest level was tested with the values 10^{-6}, 10^{-9}, 10^{-12}, 10^{-15} and 10^{-18}.

The second implementation, **CPU**, was the CPU sequential implementation using the same methods implemented in **GPU**.

The third implementation, **Cluster**, was a parallel implementation tailored to cluster of CPUs. **Cluster** used the conjugate gradient preconditioned with ILU(0) (with block Jacobi in parallel) to solve the linear system from the parabolic equation and a conjugate gradient preconditioned with geometric multigrid to solve the linear system from the elliptic equation. The geometric multigrid used only two levels with one processor, a fine and a coarse grid, and four levels in parallel (with more than one processor). Gauss-Seidel was used as the relaxation method. On the coarsest level, the direct LU method was used. To solve the non-linear system of ODEs, the explicit Euler method was used. The **Cluster** implementation used the PETSc [14] and MPI [15] libraries. More details about this parallel implementation can be found in [9,16,17].

Section 3 compares the performance of the three implementations. Because the **GPU** implementation uses single precision operations, Section 3 also presents an analysis of the numerical

Please cite this article in press as: R.M. Amorim, R. Weber dos Santos, Solving the cardiac bidomain equations using graphics processing units, J. Comput. Sci. (2012), http://dx.doi.org/10.1016/j.jocs.2012.06.007