Journal of Computational and Applied Mathematics 295 (2016) 70–82

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

Lattice Boltzmann method for parallel simulations of cardiac electrophysiology using GPUs

J.O. Campos [a], R.S. Oliveira [b], R.W. dos Santos [a,c], B.M. Rocha [a,c,*]

[a] Programa de Pós-Graduação em Modelagem Computacional, Universidade Federal de Juiz de Fora, Brazil

[b] Departamento de Ciência da Computação, Universidade Federal de São João del Rei, Brazil

[c] Departamento de Ciência da Computação, Universidade Federal de Juiz de Fora, Brazil

ARTICLE INFO

Article history:
Received 30 September 2014
Received in revised form 31 January 2015

Keywords:
Lattice Boltzmann method
Cardiac electrophysiology
Monodomain
High performance computing

ABSTRACT

This work presents the lattice Boltzmann method (LBM) for computational simulations of the cardiac electrical activity using monodomain model. An optimized implementation of the lattice Boltzmann method is presented which uses a collision model with multiple relaxation parameters in order to consider the anisotropy of the cardiac tissue. With focus on fast simulations of cardiac dynamics, due to the high level of parallelism present in the LBM, a GPU parallelization was performed and its performance was studied under regular and irregular three-dimensional domains. The results of our optimized lattice Boltzmann parallel implementation for cardiac simulations have shown acceleration factors as high as 500× for the overall simulation and for the LBM a performance of 419 mega lattice updates per second was achieved. With near real time simulations in a single computer equipped with a modern GPU these results show that the proposed framework is a promising approach for application in a clinical workflow.

1. Introduction

Recently the fields of computational modeling and medical image processing have advanced significantly allowing the biomedical engineering community to focus on the development of patient-specific models. In the context of cardiac electrophysiology, such personalized models offer the ability to better understand the heart in pathological conditions, to develop and improve new therapies and also to optimize complicated procedures. However, before introducing these computer models for clinicians, efficient numerical and computational techniques for the fast solution of the mathematical models underlying the complex phenomena of electrical activity of the heart have to be pursued.

There exist an extensive literature about numerical methods for the solution of the reaction–diffusion partial differential equations (PDEs) that govern cardiac electrophysiology. Among them, the finite element method (FEM) [1] is the most used method for spatial discretization, although the finite difference and the finite volume methods [2,3] and also alternative schemes such as spectral methods have been proposed [4].

In the field of computational fluid dynamics (CFD), the lattice Boltzmann method (LBM) [5] has been proposed for thesimulations of fluid flows as an alternative to these conventional methods such as the FEM. The LBM has been successfully

* Correspondence to: Departamento de Ciência da Computação, Universidade Federal de Juiz de Fora, Rua José Kelmer, s/n, 36036-900, Juiz de Fora/MG, Brazil. Tel.: +55 32 2102 3481.

E-mail addresses: joventinoo@gmail.com (J.O. Campos), sachetto@ufsj.edu.br (R.S. Oliveira), rodrigo.weber@ufjf.edu.br (R.W. dos Santos), bernardomartinsrocha@ice.ufjf.br (B.M. Rocha).

http://dx.doi.org/10.1016/j.cam.2015.02.008
0377-0427/

applied to study several fluid flows problems [5,6] as well as in the context of biomedical engineering for simulating arterial blood flow [7]. The method has also been used to simulate reaction–diffusion equations [8,9], which are similar to the ones used to describe the electrical activity in cardiac tissue.

Among the computational techniques used for accelerating simulations of cardiac electrophysiology, some studies have shown that the use of Graphics Processing Units (GPUs) for the numerical simulations is a promising alternative which can achieve a near real time performance [10–14]. In CFD some works report a real time performance for their simulations using the LBM on GPUs [15,16].

The objective of this study is to describe the lattice Boltzmann method for the simulations of cardiac electrophysiology dynamics. The LBM for the reaction–diffusion equations of electrophysiology is first described. Then it is validated and compared against a traditional FEM solution in a benchmark problem [17]. Considering that the LBM is very suitable to parallel computing due to its high locality, it turns out that a parallel GPU implementation of the LBM for solving cardiac electrophysiology models is a promising approach for performing near real time simulations. The GPU parallelization of the cardiac LBM solver is described and discussed in terms of its performance when applied to some test problems.

2. Cardiac electrophysiology equations

Cardiac tissue is made of interconnected myocytes coupled through a conductive intracellular space, which is surrounded by a conductive fluid in the extracellular space between the cells. Mathematically this can be modeled as two interpenetrating spaces (intra- and extracellular spaces) that occupy the same volume and are separated from each other by the cardiac cell membrane.

The dynamics of cardiac electrophysiology can be described by reaction–diffusion PDEs that consider the kinetics of a single myocardial cell and ionic current flow between the intra- and extracellular spaces as well as the gap junctions, which are specialized proteins that form a intercellular connection between neighboring myocytes. The most complete mathematical model of cardiac electrical activity is the bidomain model that considers the intra- and extracellular spaces in an averaged sense as conductive regions coupled through the transmembrane current. It can be formally derived by considering conservation of intra- and extracellular currents, by coupling the two domains with the transmembrane current, and making use of Ohm's law and Kirchhoff's law; see [18] for a detailed derivation.

Since the bidomain model consists of a complex PDE system, that involves computational expensive numerical solution, it is common to assume that the intra- and extracellular domains have equal anisotropy ratios to obtain a simplified model called the monodomain model. Following this assumption the monodomain model can be obtained by a reduction from the bidomain model and is entirely written in terms of the transmembrane potential $v = v(\mathbf{x})$, defined as the difference between the intra- and extracellular potentials.

The monodomain model is given by

$$\chi \left(C_m \frac{\partial v}{\partial t} + I_{ion}(v, \boldsymbol{\eta}) \right) = \nabla \cdot (\boldsymbol{\sigma} \nabla v), \tag{1}$$

$$\frac{\partial \boldsymbol{\eta}}{\partial t} = \mathbf{f}(v, \boldsymbol{\eta}), \tag{2}$$

where χ is the surface–volume ratio and C_m is the membrane capacitance. The first term on the left-hand side is the rate of change of the transmembrane potential with time, the second term I_{ion} controls the total ion current, while the term on the right-hand side describes the diffusion. The reaction term I_{ion} is a function of v and a vector of state variables $\boldsymbol{\eta}$ and is coupled to the system of ordinary differential equations (2) that controls the kinetics of the state variables. These variables represent the dynamics of the ion channels in the cell membrane, as discussed in Cell-level models section. We consider appropriate initial conditions and apply no-flux boundary conditions, which are given by $\mathbf{n} \cdot \boldsymbol{\sigma} \nabla v = 0$, where $\mathbf{n}$ is the unit vector normal to the boundary.

In the monodomain model, $\boldsymbol{\sigma}$ is the conductivity tensor that describes the electrical properties of the tissue. Anatomic studies have shown that heart muscle is a strongly anisotropic material due to the fiber structure that comprises the tissue. Therefore, the electrical propagation is faster in the fiber direction than in the cross-fiber directions. If we consider the cardiac tissue as a transversely isotropic material, whose preferential direction is given by the fiber direction $\mathbf{a}_l$, then the conductivity is given by

$$\boldsymbol{\sigma}(\mathbf{x}) = \sigma_t \mathbf{I} + (\sigma_l - \sigma_t) \mathbf{a}_l(\mathbf{x}) \mathbf{a}_l^T(\mathbf{x}), \tag{3}$$

where $\mathbf{I}$ is the identity matrix and σ_l and σ_t are the values of the conductivity in the fiber and cross-fiber direction, respectively.

2.1. Cell-level models

The dynamics of the biophysical processes that underlie the action potential (AP) in a cardiac cell is typically described by a set of ordinary differential equations (ODEs) that model the total ion current I_{ion} through ion channels (mainly Na^+, K^+ and Ca^{2+}) in the cell membrane.

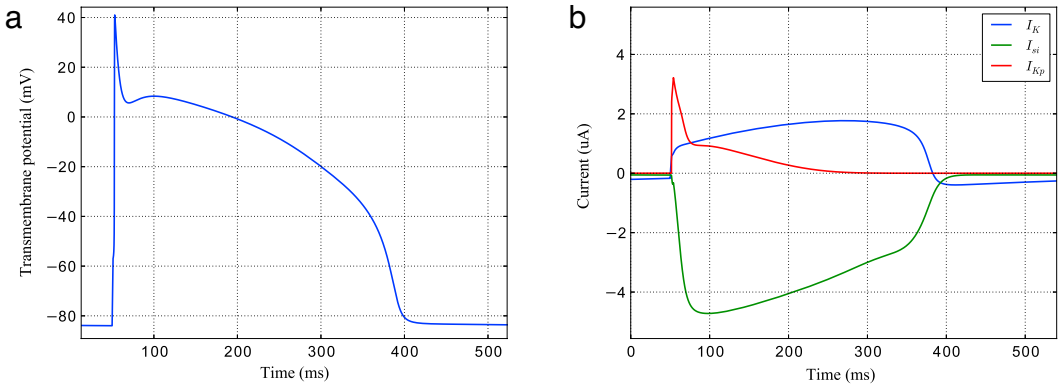


Fig. 1. (a) Action potential of the LRI cell model and (b) some of the ion currents of the model.

In this work three different cell models from distinct species with different levels of complexity were considered to simulate the kinetics of the reaction term I_{ion} in Eq. (1). The following mathematical models were used: the Luo–Rudy model I for guinea pig ventricular tissue [19], the ten Tusscher–Panfilov model for human ventricular tissue [20] and the Mahajan–Shiferaw model [21] for rabbit ventricular tissue. In the following a brief description of each cell model is given.

Luo–Rudy model. The Luo–Rudy model I (LRI) presented in [19], is a mathematical model that describes the electrical activity in ventricular cells of mammalians. In the LRI model, the total ion current I_{ion} is defined as

$$I_{ion}(v, \boldsymbol{\eta}) = I_{\mathrm{Na}} + I_{si} + I_{\mathrm{K}} + I_{\mathrm{K1}} + I_{\mathrm{Kp}} + I_b, \tag{4}$$

where I_{Na} is the rapid sodium current, I_{si} is the slow inward current, I_{K} is the time-dependent potassium current, I_{K1} is the time-independent potassium current, I_{Kp} is the plateau potassium current and I_b is the time-independent background current.

Four of the six currents present in the model are controlled by the so called gating variables, which are described by ODEs of the following form:

$$\frac{dn}{dt} = \frac{n_\infty - n}{\tau_n}, \quad \text{with } n_\infty = \frac{\alpha}{\alpha + \beta} \text{ and } \tau_n = \frac{1}{\alpha + \beta}, \tag{5}$$

where α and β are functions of v. Fig. 1(a) shows the action potential and (b) shows some of the ion currents present in Eq. (4) of the LRI cell model. The complete description of this model can be found in [19].

ten Tusscher–Panfilov model. One of the most used mathematical models for computer simulations of human ventricular cells is the ten Tusscher–Panfilov model (TT2) [20]. This model has 19 state variables which are described by ODEs, and it also describes the intracellular calcium concentration and conductances of the ionic channels. The total ion current I_{ion} of this model is given as the sum of the following currents:

$$I_{ion} = I_{\mathrm{Na}} + I_{\mathrm{K1}} + I_{to} + I_{\mathrm{Kr}} + I_{\mathrm{Ks}} + I_{\mathrm{CaL}} + I_{\mathrm{NaCa}} + I_{\mathrm{NaK}} + I_{p\mathrm{Ca}} + I_{p\mathrm{K}} + I_{b\mathrm{Ca}} + I_{b\mathrm{Na}}. \tag{6}$$

A complete description of the currents that form the total ion current, the equations of the model and its parameters can be found in the original publication [20].

Mahajan–Shiferaw model. The Mahajan–Shiferaw model (MSH) is a modification of a previous rabbit ventricular model to reproduce action potentials at rapid rates relevant to the study of ventricular arrhythmias. The model consists of 26 nonlinear ODEs, including voltage, gating variables, Markov states for the L-type calcium current, troponin buffers, average calcium concentration and others. For a complete description of the model see [21].

3. Lattice Boltzmann method

In order to solve the reaction–diffusion model (1), that describes the electrical activity in cardiac tissue we will use the lattice Boltzmann method (LBM). We will describe the LBM adapted for the numerical simulation of reaction–diffusion equations as discussed in [8,9], but here we focus on cardiac electrophysiology equations.

For simulations using the LBM, the domain is discretized by an equally spaced Cartesian grid. Every node of the grid has N velocity directions $\mathbf{e}_i$ ($i = 0, \ldots, N-1$) and N particle distribution functions f_i which describe the probability of a certain number of particles moving in the velocity direction $\mathbf{e}_i$. In the LBM framework, the distribution function of the scalar quantity v at time t, position $\mathbf{x}$ with velocity in the direction $\mathbf{e}_i$ is denoted by $f_i(\mathbf{x}, t)$. Here v is the transmembrane potential.

The lattice Boltzmann equation that describes the evolution of the particle distribution functions $f_i(\mathbf{x}, t)$ may be written as:

$$f_i(\mathbf{x} + \mathbf{e}_i \Delta x, t + \Delta t) - f_i(\mathbf{x}, t) = \Omega(\mathbf{x}, t), \tag{7}$$

where $\Omega(\mathbf{x}, t)$ is the collision operator for v and depends on distribution functions $f_i(\mathbf{x}, t)$, and Δt and Δx are the time step and grid interval, respectively. The left-hand side represents free transport of the particles (streaming), while the right-hand side describes interactions of the particles through collisions. Although the LBM kinetic equation can be combined into a single statement as in Eq. (7), collision and streaming steps are usually considered separated, since the collision step is purely local and the streaming step involves the traveling of the particles to neighboring nodes along velocity directions.

For reaction–diffusion problems, as is the case of the monodomain equation, the collision term may be written as the sum of a reactive term Ω^R and a non-reactive term denoted by Ω^{NR}, as described in [8]. The non-reactive part of the collision operator is described by the traditional Bhatnagar–Gross–Krook (BGK) approximation [22], which models the collisions as a single-time relaxation towards a local equilibrium. In this case the non-reactive collision operator is given by

$$\Omega^{NR}(\mathbf{x}, t) = -\frac{1}{\tau}(f_i(\mathbf{x}, t) - f_i^{\mathrm{eq}}(\mathbf{x}, t)), \tag{8}$$

where τ is the relaxation parameter and f_i^{eq} is the equilibrium distribution function that depends on the variable v and velocity $\mathbf{u}$. In general, when the LBM is applied for the numerical simulation of fluids through the Navier–Stokes (NS) equations, the equilibrium distribution function f_i^{eq} is given by

$$f_i^{\mathrm{eq}}(\mathbf{x}, t) = w_i\, v\left(1 + \frac{(\mathbf{e}_i \cdot \mathbf{u})}{c_s^2} + \frac{(\mathbf{e}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}}{2c_s^2}\right), \tag{9}$$

where c_s is the velocity of sound on the lattice and the w_i are coefficients that depend on the type of lattice in use. However, since we are interested in modeling a pure reaction–diffusion problem, we will consider that the velocity is zero, i.e., $\mathbf{u} = 0$. In this case, the equilibrium distribution function reduces to the following form

$$f_i^{\mathrm{eq}}(\mathbf{x}, t) = w_i\, v. \tag{10}$$

For the reactive part of the collision operator, it is assumed [8] that R represents the change in v due to the reaction, which enables the reactive part to be distributed among the different directions proportionally to w_i, that is

$$\Omega^R(\mathbf{x}, t) = w_i R. \tag{11}$$

We recover the macroscopic quantities by taking the moments of the distribution functions f_i. Thus, the macroscopic variable v, which is the solution of Eq. (1), can be recovered as the sum of the distributions f_i as $v(\mathbf{x}, t) = \sum_{i=0}^{N-1} f_i(\mathbf{x}, t)$, where N is the number of lattice directions and depends on the lattice model used. The lattice model used in this work will be described next.

3.1. LBM for the monodomain model

The reaction term R that appears in the reactive collision operator Ω^R is determined by the cell model used to describe the kinetics of the cellular membrane. In general, these models are described by a system of non-linear ODEs. An example of system of ODEs for these state variables is given by Eq. (5) of the Luo–Rudy model I. In particular, for the Luo–Rudy model the reactive term is $R = I_{ion} + I_{stim}$ where for the LRI model I_{ion} is given by (4) and for the TT2 model it is given by Eq. (6).

In this work the D3Q7 lattice model was used to solve three dimensional reaction–diffusion problems. The notation DnQm represents a lattice Boltzmann model in n dimensions with m discrete velocity directions. In our case we consider 7 velocity directions for the distribution functions, that is, the particles can stream to six directions (shown in red and blue in Fig. 2), or remain at rest. The coefficients w_i for the D3Q7 model associated with the distribution function in direction $\mathbf{e}_i$ are given by

$$w_0 = \frac{1}{4}, \qquad w_{1-6} = \frac{1}{8}, \tag{12}$$

and the corresponding velocity directions are shown in Fig. 2. For two dimensional diffusion problems the D2Q5 model can be used; more details can be found in [23].

In our implementation we proceed as follows: first, given v^n and $\boldsymbol{\eta}^n$ the current approximations to v and $\boldsymbol{\eta}$ at time $t = t_n$, we solve the system of ODEs given by

$$\begin{aligned} C_m \frac{dv}{dt} &= -I_{ion}(v, \boldsymbol{\eta}) + I_{stim}, \qquad v(t_n) = v^n, \\ \frac{d\boldsymbol{\eta}}{dt} &= \mathbf{f}(v, \boldsymbol{\eta}), \qquad \boldsymbol{\eta}(t_n) = \boldsymbol{\eta}^n, \end{aligned} \tag{13}$$

and then, we use the intermediate solution of (13), denoted by v^*, and the $R = I_{ion}(v^n, \boldsymbol{\eta}^n) + I_{stim}$ term to compute the equilibrium distribution functions f_i^{eq} of the LBM during the collision step. After the collision, we apply the streaming and boundary conditions steps of LBM to obtain the approximate solution v^{n+1} at $t = t_n + \Delta t$. Note that here I_{stim} denotes an external stimulus that is usually applied in order to initiate electrical activity of cardiac myocytes.

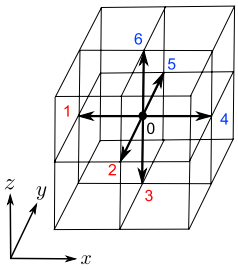


Fig. 2. The D3Q7 lattice model used to solve reaction–diffusion problems in three dimensions with seven velocity directions. That is, each node can deliver particle distributions to each of the six neighboring nodes or remain at rest. Blue and red labels were used to emphasize opposing directions that are exploited by the optimized streaming algorithm used in this work. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.2. LBM for anisotropic reaction–diffusion equation

Most of the lattice Boltzmann models used for diffusion phenomena are limited to isotropic diffusion. This is because the LBM with the traditional BGK collision operator, does not have sufficient parameters to describe anisotropic diffusion. In fact, since it is based on only one relaxation time parameter τ, as shown in Eq. (8), this model is usually referred to as single-relaxation-time (SRT).

In order to overcome this limitation, some studies [24,25,23] have proposed different methods for incorporating anisotropic diffusion into the LBM. To account for the anisotropic nature of cardiac tissue we used the multiple-relaxation-time (MRT) collision operator, as it was proposed in [23] for a convective and anisotropic diffusion equation.

First, before we introduce the MRT model, note that if we denote by $\mathbf{f}$ and $\mathbf{f}^{\text{eq}}$ the column vectors formed by the distribution functions f_i and equilibrium distribution functions f_i^{eq}, then the SRT model can be written in terms of the following collision operator

$$\Omega^{NR} = -\mathbf{A}(\mathbf{f} - \mathbf{f}^{\text{eq}}), \tag{14}$$

where the collision matrix $\mathbf{A}$ is defined in terms of τ as $\mathbf{A} = (1/\tau)\mathbf{I}$.

The MRT model projects the vector $\mathbf{f}$ in a vector space where each component corresponds to a certain moment of $\mathbf{f}$. Then, the collision operator is applied in the moment space in order to relax each component towards the equilibrium with a different relaxation coefficient. Finally, the vector is projected back onto the original space. The MRT allows to tune the relaxation coefficient for each moment separately, which makes possible to consider anisotropic diffusion.

Following [23], the MRT model is then given by

$$\Omega^{NR} = -\mathbf{M}^{-1}\mathbf{S}\mathbf{M}(\mathbf{f} - \mathbf{f}^{\text{eq}}), \tag{15}$$

where $\mathbf{M}$ is the matrix that projects a vector onto the moment space. The definition of the matrix $\mathbf{M}$ depends on the choice of the moments. Here we consider the first moment as the conserved quantity, that is, the transmembrane potential v. The second to fourth moments are related to the flux of v in the i direction, and the remaining moments are obtained using Gram–Schmidt's procedure. The matrix $\mathbf{M}$ and the relaxation-time matrix $\mathbf{S}$ are defined, respectively, by

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 6 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 2 & 2 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 1 & 1 & -1 & -1 \end{bmatrix}, \qquad \mathbf{S}^{-1} = \begin{bmatrix} \tau_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \bar{\tau}_{xx} & \bar{\tau}_{xy} & \bar{\tau}_{xz} & 0 & 0 & 0 \\ 0 & \bar{\tau}_{yx} & \bar{\tau}_{yy} & \bar{\tau}_{yz} & 0 & 0 & 0 \\ 0 & \bar{\tau}_{zx} & \bar{\tau}_{zy} & \bar{\tau}_{zz} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \tau_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \tau_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \tau_6 \end{bmatrix}.$$

Here τ_0, τ_4, τ_5, τ_6 and $\bar{\tau}_{ij}$ are the relaxation-time coefficients. It is shown in [23] that the relationship between the relaxation coefficients and the conductivity tensor components σ_{ij} is given by

$$\bar{\tau}_{ij} = \frac{1}{2}\delta_{ij} + 4\frac{\Delta t}{\Delta x^2}\sigma_{ij}. \tag{16}$$

The coefficient τ_0 is related to the potential and τ_4, τ_5 and τ_6, which are related to the higher order moments, do not directly affect the diffusion, only the stability of the method [23].

3.3. Boundary conditions

The no-flux boundary conditions for the transmembrane potential v is a homogeneous Neumann boundary condition. This kind of boundary condition can be easily implemented in the lattice Boltzmann method by imposing that the incoming

distribution at the boundary node is equal to the outgoing one. In the context of fluid flow LBM simulations, this is known as the *bounce-back* rule [5].

To describe this rule, consider a 2D domain and the D2Q5 lattice model. Suppose we are at node located at $\mathbf{x} = (0, y)$ where the incoming distribution function f_1 is unknown. The Neumann-type boundary condition at this face can be imposed simply by taking $f_1(\mathbf{x}, t + \Delta t) = f_2(\mathbf{x}, t)$. In general, we can express this boundary treatment as

$$f_a(\mathbf{x}, t + \Delta t) = f_b^*(\mathbf{x}, t), \tag{17}$$

where f_b^* is the post-collision distribution function and the b index indicates the opposite direction to a, that is, $\mathbf{e}_a = -\mathbf{e}_b$.

3.4. Implementation

We now describe some details of the numerical implementation of the LBM solver for the cardiac monodomain equation. Initially, we present a general overview of the solution algorithm. Then, we describe the details of the two most important parts of the solver: the method used for the numerical solution of the nonlinear system of ODEs and the optimized LBM solver used for evolution of the diffusion part of the monodomain equation.

3.4.1. Solution algorithm

An overview of the complete solution strategy is presented in Algorithm 1. The first steps (lines 1 and 2) set up the initial conditions of the ODEs for each node of the grid and allocate memory for the LBM solver. The initial setup of the LBM initializes the distribution functions f_i to the equilibrium distribution function f_i^{eq}, which are computed with the initial value of the transmembrane potential $v = v_0$.

Algorithm 1: Solution of the monodomain problem by the lattice Boltzmann method.

```
1.  InitializationODEs();
2.  InitializationLBM();
3.  for k ← 1 to N_steps do
4.  |  t ← t + Δt;
5.  |  for j ← 1 to N_nodes do
6.  |  |  Advance ODEs of node j using the RL method;
7.  |  |  for i ← 0 to N − 1 do
8.  |  |  |  CollisionAndStreamingSwap();
9.  |  |  |    f_i^*(x, t) = f_i(x, t) + Ω(x, t);
10. |  |  |    f_i(x + e_i Δx, t + Δt) = f_i^*(x, t);
11. |  |  end
12. |  end
13. |  ApplyBoundaryConditions();
14. |  OutputData();
15. end
```

During the time integration loop (lines 3–15), for each node of the computational grid, we first advance the ODEs in time using the Rush–Larsen method (line 6) and then with the updated values of the reaction term I_{ion} and v, for each of the N velocity directions (lines 7–11), we apply the collision and streaming steps of the lattice Boltzmann method. The collision and streaming steps, described by the formulas in lines 9 and 10, respectively, are fused in the `CollisionAndStreamingSwap()` routine in order to reduce memory transactions. Thus, the solver traverses the grid only once at each time step and first, using the updated values of v and I_{ion}, the collision operator is applied to obtain the intermediate values for distribution functions f_i^*. Then, the streaming of the particles to the neighboring nodes, as defined by the lattice model defined in Fig. 2, is performed. After that, we apply the boundary conditions to the nodes on the boundary and, if required, data is written to file.

3.4.2. ODEs solver

The numerical solution of the system of ODEs from Eq. (13) at each node of the grid was performed using the Rush–Larsen (RL) method [26]. The RL method is an explicit method that takes advantage of ODEs of the form shown in Eq. (5) for the numerical integration. The RL method is easy to implement and has better stability properties than the explicit Euler method. Thus, it allows the use of larger time steps resulting in an efficient method for the numerical solution of cell models of cardiac electrophysiology.

3.4.3. Optimizations of the LBM solver

Here we describe some strategies that optimize the LBM solver used for the diffusion part of the monodomain equation. These optimizations were used for both the CPU and GPU implementations.

To achieve an optimized GPU implementation for fast simulations, the memory layout of the particle distribution functions f_i has to be considered carefully. For optimized GPU memory transactions, we store the particle distribution functions f_i in memory using a one dimensional array, such that f_i is addressed as `f[i*Nx*Ny*Nz+z*Ny*Nx+y*Nx+x]`, where `Nx`, `Ny`, `Nz` and `x`, `y`, `z` are the number of nodes and their indices in the x, y and z directions, respectively. Thus, the ordering of the data in `f` follows the Structure of Arrays (SoA) format.

Traditional implementations of the LBM often use an additional temporary array for storing the particle distribution functions f_i since during the streaming step data is shifted in space. Otherwise, if no temporary memory is used, some data would be overwritten during the streaming step. One approach to overcome this extra memory usage is to use the swap algorithm [27,28] for the streaming step, which never copies data to a temporary array, but always swaps it with another variable. This requires that the indexes of the particle distribution functions are reorganized (see Fig. 2) in such a way that the rest-particle function has index 0, and that the opposite of an index from the lower half is found by adding $(N-1)/2$ to it: $opposite(i) = i + (N-1)/2$ for $i = 1 \ldots (N-1)/2$. The algorithm requires a strict processing order of the lattice nodes and proceeds by continuously swapping half of a particle distribution function of a node with the ones from its neighbors. With this approach we are able to perform the collision and streaming steps together so that grid traversal is performed only once.

To efficiently represent irregular geometries, like the ones shown in Fig. 3, a sparse data structure [29] (or indirect addressing scheme) was used in our implementation instead of the traditional *full matrix* representation (or direct addressing scheme). This scheme requires an adjacency array of $N_{nodes} \times N - 1$ integers to store the information about the neighbors of each node along each of the directions $\mathbf{e}_i$. Since there is no streaming in the $\mathbf{e}_0$ direction, there is no neighbor to store in the adjacency array. In the full matrix scheme, for a node labeled as `k` with grid indexes given by `ix`, `iy` and `iz`, we access a neighbor in the x direction by simply specifying `ix-1`. With the indirect addressing scheme, first we need to consult the adjacency array `adj[k][0]` to find out the index of the neighboring node and then proceed with the computation (in this example we considered that the previous neighbor in the x direction has column index 0).

At first glance it may seem that the sparse data structure approach requires more memory, since it needs to store an additional array of integers. However, for the geometries used here (see Fig. 3) the sparse data structure scheme uses only 36% of the memory required by the full matrix scheme.

4. Parallel implementation

After the CPU version of the code was implemented, and validated using a benchmark problem that will be described in the numerical experiments, we parallelized it using CUDA. The present GPU LBM solver was parallelized such that one thread performs the complete LBM algorithm at one spatial location $\mathbf{x}$ in the computational domain. Since we store f in order of lattice direction i and then by spatial coordinates, we ensure a contiguous memory access pattern, which is important for a GPU implementation to achieve good performance.

The CUDA environment allows to define C functions, named CUDA kernels, that will be executed in the device (GPU) when they are called by the CPU code at the host. For executing CUDA kernels, it is necessary to define the execution configuration that describes how the threads are organized. The threads are organized in blocks, each block is executed by some GPU multiprocessor and this configuration may impact the performance of the code. Specific CUDA kernels were developed to solve each part of the problem, such as the solution of the ODEs and the collision and streaming steps of the LBM solver. In this work all CUDA kernels were launched with a fixed number of threads per block n_{tb} and due to the data layout used to store the distribution functions only one-dimensional blocks were used. To assess the performance of the GPU code we used a different number of threads per block: $n_{tb} \in \{64, 128, 256, 512\}$.

During the parallel simulations, all computations were performed on the GPU without any data transfer to CPU, which are very costly and may degrade performance. A temporary array was created to store the macroscopic variable v at some time instants and then at the end of the simulation one data transfer was performed from the GPU to the CPU to write the results to a file.

To assess the performance of the implementations of this work we used two metrics: speedup and MLUPS (Mega Lattice Updates Per Second). The parallel speedup shows by how much the optimized parallel GPU implementation was faster in terms of execution time when compared to the CPU implementation. Therefore, speedup was computed as $S = T_{GPU}/T_{CPU}$, where T_{GPU} and T_{CPU} are the execution times of the GPU and CPU implementations. MLUPS accounts for the time to calculate one time step for a given grid and is a handy unit for measuring performance of LBM codes. MLUPS was calculated as $MLUPS = (N_{\text{nodes}} \times N_{steps})/T_p$, where T_p is the time spent to solve the parabolic PDE.

5. Numerical experiments

In this section we present the numerical experiments carried out in this work and the computing environment used to perform them. We also report the results of the experiments and compare the performance of the GPU implementation with the single core CPU implementation.

5.1. Computing environment

The numerical experiments were carried out a Linux 2.6.32 machine, with 12 GB of memory, Intel Xeon E5620 2.40 GHz processor with 4 cores and 12 MB of cache memory, which will be simply referred here as CPU code. The LBM solver

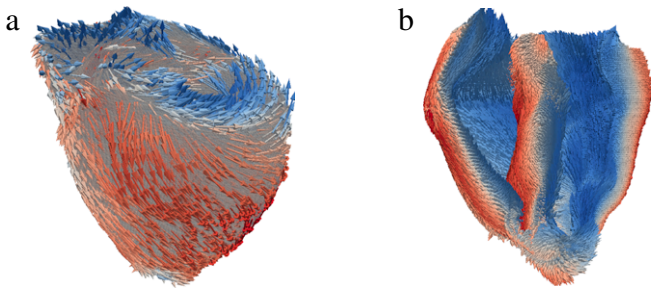


Fig. 3. Realistic geometries used in the simulations. (a) Rabbit bi-ventricular geometry and (b) fiber orientation. The vectors that define the fiber orientation were colored according to its long axis component. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

was implemented in C++ and compiled with GNU GCC version 4.4.7 with the following optimization flags enabled: `-O3 -ffast-math`. Our GPU testbed is a NVIDIA Tesla M2050 processor, comprising 448 CUDA cores organized as 14 streaming multiprocessors. This GPU has a total of 3 GB of memory and a memory bandwidth of 148 GB/s. The CUDA code was compiled with NVIDIA nvcc compiler version 6.0.1 with the same optimization flags of the CPU code. The simulations were carried out using both single (SP) and double (DP) precision floating point arithmetic and the GPU comparisons against the CPU were always made using the same floating point representation.

5.2. Test problems

Three different test problems to evaluate the LBM code for the simulation of the electrical activity of cardiac tissue were considered. Some of these problems use simplified geometries, whereas the others use realistic geometries of the heart. Here we describe these problems briefly.

Benchmark problem. A benchmark problem was proposed in [17] for the validation of implementations of the monodomain model. The domain consists of a rectangular region of size $2 \times 0.7 \times 0.3$ cm^3 and the fibers are defined to be parallel to the longitudinal direction and the conductivity tensor is considered transversely isotropic as defined by Eq. (3). The conductivity in the fiber direction is 1.334 mS/cm, whereas the conductivity in the cross-fiber direction is 0.176 mS/cm. Other parameters are defined as: $\chi = 1400$ cm^{-1} and $C_m = 1$ μF/cm^2. The 2006 version of the ten Tusscher cell model [20] is used in this test case. An external stimulus of $-50\,000$ μA/cm^3 is applied during 2 ms in a small cubic region of 0.15 cm^3 at one corner of the tissue to trigger the electrical activity.

Cubic region. A slightly modified version of the benchmark problem is used, wherein the domain consists of a cube of dimensions $1.28 \times 1.28 \times 1.28$ cm^3. The parameters, conductivity and stimulation protocol are the same used for the benchmark problem. The performance was evaluated using an increasing number of nodes for the three directions of the cube. Two different cell models having different levels of complexity, the LRI and TT2 models, were used to simulate 1 s of the electrical activity using a time step of $\Delta t = 0.1$ ms.

Rabbit bi-ventricular geometry. A realistic mesh of both left and right ventricles of the rabbit including fiber orientation was used for anisotropic simulations of the electrical activity. This rabbit geometry was obtained by a procedure where the rabbit heart was mechanically fixed, the ventricles were filled with silicone and then appropriately sliced for imaging. Segmentation was applied to each image in order to determine the inside and outside of the tissue. Tissue blocks were extracted from these slices, and fiber orientation was determined. Then, data were fitted to a finite element description in prolate spheroidal co-ordinates. More details can be found in [30]. The version of this mesh used in the present work consists of a regular hexahedral grid suitable for our lattice Boltzmann simulations.

Two grids were used in the simulations, a coarse one which comprises 522 841 nodes with a spacing of $\Delta x = 250$ μm and a refined version with 3 760 560 nodes and $\Delta x = 125$ μm. The refined grid was obtained by simply halving Δx in each direction. The MSH cell model was used for these simulations using the RL method with a time step of $\Delta t = 0.1$ ms. A transversely isotropic conductivity tensor was used with $\sigma_l = 1.334$ mS/cm and $\sigma_t = 0.176$ mS/cm. The geometry and orientation of the fibers are shown in Fig. 3(a) and (b).

5.3. Validation using the benchmark problem for electrophysiology

The benchmark problem was solved using the lattice Boltzmann method with different time steps ($\Delta t = 0.05$, 0.01 and 0.005 ms) and space steps ($\Delta x = 500$, 200 and 100 μm). In order to measure the accuracy of the code we considered the activation time of a node, which was defined as the time at which the transmembrane potential v reaches the value of 0 mV. The same metric, taken from control points along the diagonal line of the domain, was used in [17] to compare the results of several codes. The propagation of the wave that electrically activates the slab of cardiac tissue of the benchmark problem, from the application of the electrical stimulus until its complete activation, is shown in Fig. 4(a) to (d).

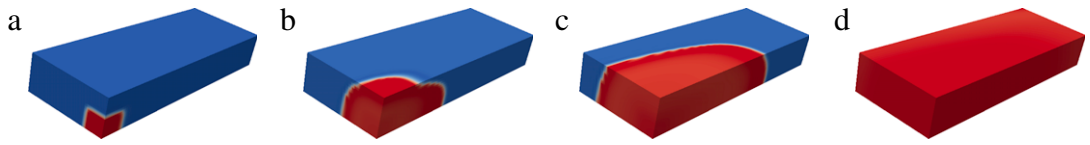


Fig. 4. Propagation of the electrical wave of excitation through the domain defined by the benchmark problem at different time steps.

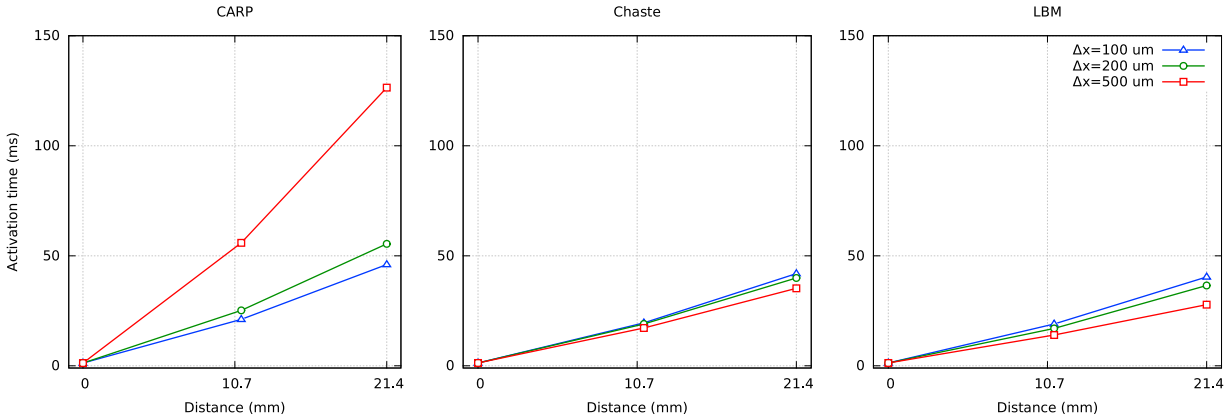


Fig. 5. Activation times (the time instant that the potential v passes through 0 mV upon first activation) of the CARP and Chaste codes, as reported in [17], and the activation times obtained with the present implementation using the LBM. The figures present the results of the simulations with a fixed time step $\Delta t = 0.005$ ms and with different spacings $\Delta x = \{500, 200, 100\}$ μm.

We present in Fig. 5 the activation times of two codes, CARP [31] and Chaste [32], which were reported in [17] and represent the different behaviors obtained by the codes tested in the benchmark experiment. The results of the LBM for the benchmark problem are also shown in Fig. 5 for the smallest time step $\Delta t = 0.005$ ms. The activation times resulting from the LBM simulation also converge to the same value obtained by the other codes, which is around 41 ms. We also note that the behavior of the LBM is very similar to the one presented by the Chaste code. Thus we conclude that the LBM can be used to perform simulations of the electrical activity of cardiac tissue.

Next, we carried out a convergence study with respect to the time step Δt used for the simulations, keeping the spacing Δx constant. To verify the convergence with respect to time step, we performed simulations with the following time steps $\Delta t = \{0.1, 0.05, 0.025, 0.0125, 0.00625\}$ ms. Although spacing was fixed for each test, we carried out simulation with different spacings $\Delta x = \{500, 200, 100\}$ μm (the same values used in the benchmark study [17]). As in the previous experiment, the activation time of the corner node was used as reference.

Fig. 6 shows the results of this experiment. In general, for large time steps the conduction velocity of the electrical wave that spreads over the cardiac tissue is slower and thus, the activation time at the reference node is larger. However, the figure clearly shows that as we reduce Δt, the value of the activation time at the reference node converges. With smaller spacings Δx of 200 μm and 100 μm the converged value of the activation time at the reference node is very close (about 40 ms). The converged value at the reference node for the $\Delta x = 500$ μm case is different, mainly due to the coarse discretization employed.

5.4. Electrical activity in a regular domain

The performance results of the parallel LBM solver for the cubic region of cardiac tissue are reported in Fig. 7 in terms of the speedups obtained. The reported speedup considers the time to solve the entire problem, that is, it includes the solution of ODEs and of the parabolic PDE for all time steps. The GPU results reported in this test were obtained using different execution configurations for the kernels, in particular, we used 64, 128, 256 and 512 threads per block. It is important to note that, for some execution configurations, such as the $n_{tb} = 64$ case, since the maximum number of blocks per grid is 65 535, it was not possible to perform the simulations with domain sizes larger than 128^3. For the $n_{tb} = 128$ case, it was not possible to launch the kernels when the domain size was larger than 192^3 nodes.

Fig. 7 shows that, as the problem size increased, better speedups were observed. This is because as we increase the number of lattice nodes, we were able to better take advantage of the massive parallelism provided by the GPU. The speedups obtained for the LRI model were greater than for the TT2 model, since the latter has a more complex ODE system and therefore involves more floating point operations as well as memory transactions. In single precision, speedup factors of about 500 were obtained for some cases using the LRI model, whereas factors of about 400 were achieved for the TT2 model. In double precision, we observed good performances as well, but the speedups were halved, as expected. The best performance was achieved with 128 threads per block and large domains for both LRI and TT2 models.

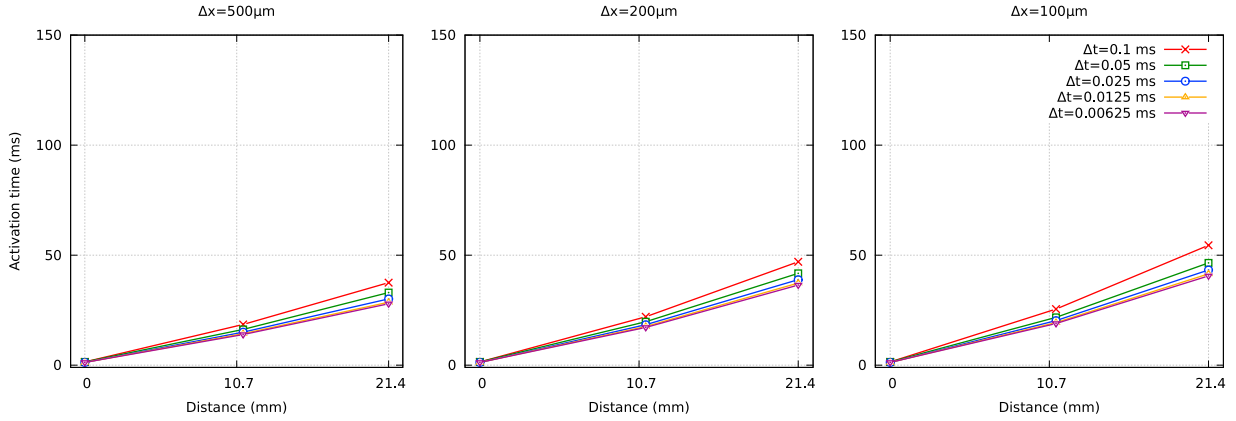
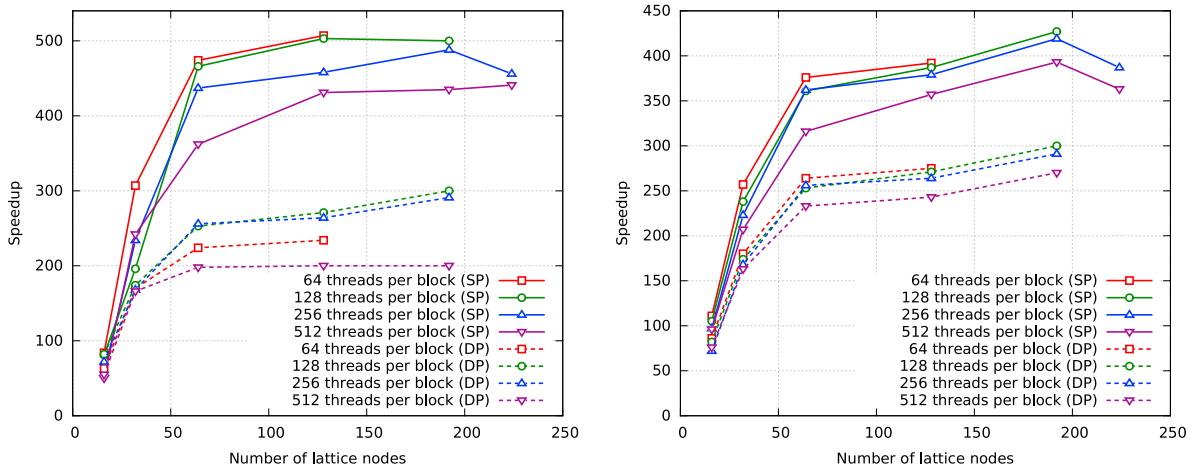


Fig. 6. Activation times obtained with the present implementation using the LBM varying the time step Δt for a fixed spacing Δx. The plots show the convergence with respect to time step.



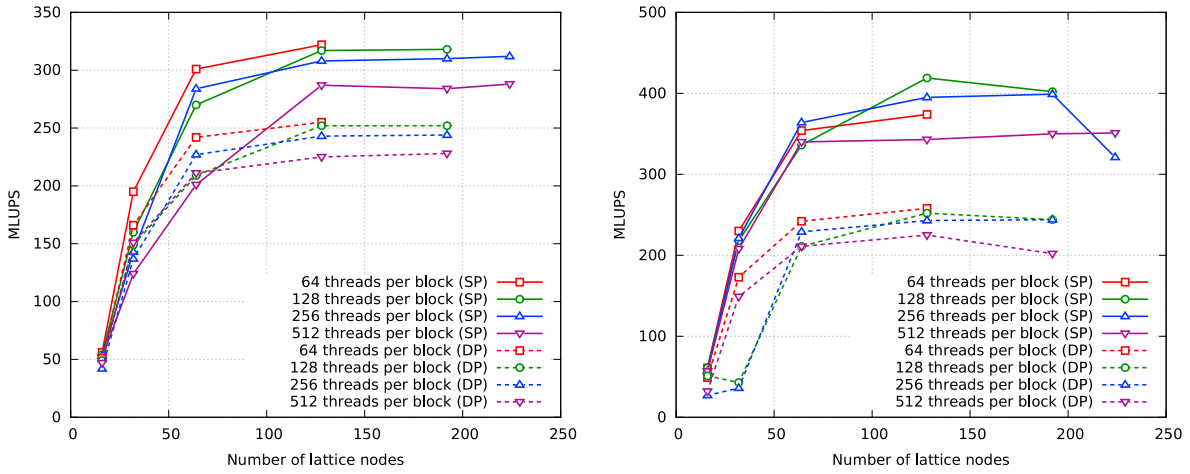
(a) LRI. (b) TT2.

Fig. 7. Performance of the code with respect to speedup for the D3Q7 lattice with different discretizations (number of lattice nodes per direction) of the cubic domain. The simulations were performed using both single (SP) and double precision (DP) floating point arithmetic and with a different number of threads per block.

Table 1
Total elapsed time (in s) and speedup obtained for the LRI and TT2 cell models.

Cell model	N_{nodes}	CPU	GPU	Speedup
LRI	32^3	689	3	230
	64^3	5 550	13	427
	128^3	44 400	97	458
	192^3	149 000	305	488
	224^3	237 000	519	457
TT2	32^3	1 420	6	237
	64^3	13 500	37	365
	128^3	109 000	287	380
	192^3	405 000	966	419
	224^3	589 000	1520	387

In Table 1 we report the execution times of the CPU and GPU and the respective speedup for the simulations using $n_{tb} = 256$ using single precision, which achieved the best performance. First note that the execution times of the GPU for small grid sizes were close to a real-time performance, taking 6 s and 37 s of execution time to simulate 1 s of electrical activity for the 32^3 and 64^3 case, respectively, with the more complex TT2 cell model. Different speedups were obtained for the two cell models, with higher accelerations for the LRI model (8 ODEs) and slightly smaller speedups for the TT2 model (19 ODEs). For the 192^3 grid, a 488-fold speedup factor and a near real-time performance were achieved using the LRI model.



(a) Full conductivity tensor.

(b) Diagonal conductivity tensor.

Fig. 8. Performance of the code for the D3Q7 lattice with different discretizations of the cubic domain.

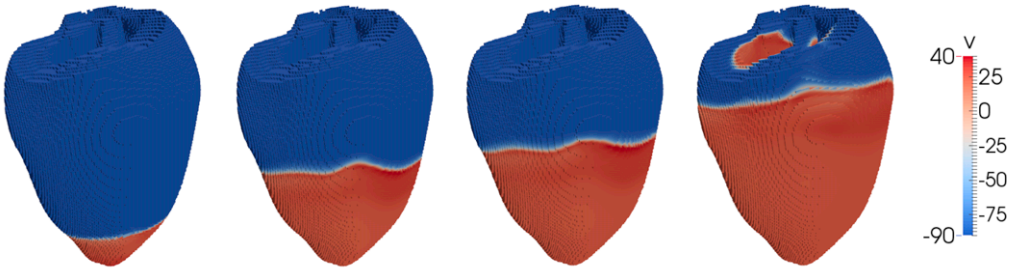


Fig. 9. Simulations of electrical activity in the rabbit ventricles.

In this test case since the fiber is aligned with the x-axis, the resulting conductivity tensor is diagonal. We note that the implementation of the non-reaction term of the collision given in Eq. (15) does not perform matrix–matrix multiplication, rather it is hand-coded in order to optimize numerical computations. Here, since the conductivity tensor is diagonal, we considered two implementations for the LBM solver: one that exploits this fact in the code and another, more general, that takes the full conductivity tensor, without exploiting its diagonal structure for this test case.

We present the performance results of the LBM solver in terms of MLUPS using the GPU for the general and diagonal implementations in Fig. 8(a) and (b), respectively. The simulations were performed using both single and double precision floating point arithmetic and with a different number of threads per block. Since the LBM is only used to solve the parabolic PDE, the results for both cell models LRI and TT2 are very similar. The differences in the ODEs do not affect the LBM and thus we only report the results for the TT2.

First, it is clear that the diagonal implementation had an overall better performance than the general implementation, since its computations are simplified due to the diagonal structure of the conductivity tensor and thus it performs less floating point operations. The general implementation obtained good performances of about 320 MLUPS with 128 and 256 threads per block in the finer grids of 224^3 nodes. The best performance achieved by the diagonal implementation on the 192^3 grid was about 400 MLUPS using 256 threads per block in single precision.

Although the diagonal implementation obtained a better performance, we note that it is not the general case and when complex geometries with physiological fiber orientations, such as the one from Fig. 3, are used, the general implementation has to be used. We only report on its performance here to show the impact of considering the anisotropy of cardiac tissue in the MRT collision operator of LBM solver.

5.5. Rabbit biventricular geometry

Fig. 9 presents snapshots of the simulation of the electrical activity in the rabbit bi-ventricular geometry at different time steps. In this case we apply an external stimulus at the apex to initiate the electrical activity which was simulated for 1 s.

The results for the rabbit bi-ventricular geometry are shown in Table 2, where the total elapsed time, speedup and MLUPS obtained for the LBM solver to complete 1 s of simulation using the MSH cell model are detailed.

Table 2
Execution time in seconds and speedup obtained for the rabbit geometry.

Δx (μm)	T_{CPU}	T_{GPU}	Speedup	MLUPS_{GPU}
250	24 500	85	287	265
125	217 000	635	341	281

For the finest grid, a 341-fold speedup was achieved by the GPU code in comparison to the single core CPU version for the entire problem. This clearly shows a good performance improvement when compared to the CPU implementation. In this test case 281 MLUPS were obtained with the LBM solver for the solution of the parabolic PDE.

6. Conclusions

In this work we have demonstrated the usage of the lattice Boltzmann method for the numerical solution of cardiac electrophysiology equations in 3D domains with regular and complex geometries. Here we discussed an optimized implementation of the lattice Boltzmann method with improvements in the streaming step using the swap algorithm, which has less memory requirements and allows to step through memory only once. Also a sparse representation of the LBM grid was introduced in order to reduce memory usage.

One of the main attractive features of the LBM is the high locality of its operations, therefore making it well suited for parallel implementation using modern GPUs. We have described a parallel implementation of the monodomain equation with the LBM using the CUDA environment. Then, the performance gains obtained with such approach was shown through several numerical experiments. Acceleration factors as high as 500× and 419 MLUPS were obtained and a near real time performance was achieved in a single computer equipped with a modern GPU, which makes this approach very suitable for application in a clinical workflow. The simulations were performed using both single and double precision floating point arithmetic and with a different number of threads per block.

6.1. Related works

Here we discuss some related works with focus on performance of cardiac electrophysiology simulations using accelerators and also the lattice Boltzmann method within this context.

Bartocci et al. presented in [12] two-dimensional simulations of cardiac dynamics using the monodomain model implemented with CUDA. Five different cell models, with increasing complexity were used, for the simulations in grids with variable sizes to assess their implementation. The results showed a near real-time performance. In this work we also present simulations with near real-time performance, however we present a different method for the numerical solution of the monodomain model and we focus on 3D simulations with regular and irregular geometries. To make a comparison with their results on a 2D grid of 1536^2 nodes, we consider the same cell model (TT2) and the same simulation time of 1 s. We note that the GPU we used is similar to the one used in their experiments, which was a NVIDIA Tesla C2070. In this case, their best implementation, which takes advantage of their 2D grid using texture memory and also other optimizations for the cell model, took about 300 s to simulate 1 s of cardiac dynamics in single precision. Our experiment with a regular cubic domain with 128^3 nodes, slightly less than their grid, took 287 s to complete and a speedup of 380× was obtained. It should be noted, that our implementation is generic since it uses a data structure with indirect addressing that supports complex 3D domains. We remark also that their implementation took advantage of some CUDA features optimized for spatial locality in 2D simulations.

Recently, Wang et al. have presented in [14] the performance results of an implementation of the monodomain equation on modern accelerators using different implementations with OpenMP, CUDA, OpenCL and also an implementation using the OpenACC auto-parallelization API. In this work the performance results were presented in terms of speedup and only for 2D simulations. The cell model used in [14] was a 8 variables model, which in terms of complexity is comparable to the LRI used in this work. In their best scenario in a grid with 2048^2 nodes using double precision, an acceleration factor of 200× was obtained with respect to a single core CPU implementation and 30× with respect to their OpenMP parallel code. In terms of speedup only, our best case using double precision was 300× faster than our single core CPU code. Note that, their best results were obtained with a GPU card Tesla K20, whose peak performance is much higher than the GPU we used in our experiments.

In terms of related works using the lattice Boltzmann method for simulating cardiac electrophysiology using the monodomain model, to the best of our knowledge, Rapaka et al. in [33] was the only work to report its usage in a single core CPU implementation. We used the same D3Q7 lattice model with the MRT for the collision operator as used in their work. Here, however, we described an optimized implementation of the LBM using the swap algorithm and a sparse representation for the computational grid. We also provided a detailed description of a parallel LBM GPU implementation and its performance for a reaction–diffusion problem with the objective to explore the high locality of its operation to obtain fast and near real-time simulations, which was demonstrated through our results.

It is also important to relate the present work with other works of our group [3,13] with focus on fast simulations of cardiac electrophysiology. In [13] 2D simulations of cardiac activity were carried out on the GPU with CUDA using the finite

element method for spatial discretization and the Crank–Nicolson implicit time scheme. Thus, a system of linear equations had to be solved at each time step of the solution algorithm, in contrast to the explicit nature of the LBM used in the present work. In [3] an adaptive mesh algorithm with the finite volume method was proposed for fast numerical solution of the monodomain equation using a parallel implementation with OpenMP. Due to the data structure used for adaptive mesh refinement an efficient GPU implementation still remains a challenge, however, we note that other strategies like the one proposed in [6] for mesh refinement with the LBM could be combined with this work for GPU simulations.

Acknowledgments

This work was partially funded by FAPEMIG, CAPES, UFJF and CNPq.

References

[1] B.M. Rocha, F. Kickinger, A.J. Prassl, G. Haase, E.J. Vigmond, R.W. dos Santos, S. Zaglmayr, G. Plank, A macro finite-element formulation for cardiac electrophysiology simulations using hybrid unstructured grids, IEEE Trans. Biomed. Eng. 58 (2011) 1055–1065.
[2] R.H. Clayton, A.V. Panfilov, A guide to modelling cardiac electrical activity in anatomically detailed ventricles, Prog. Biophys. Mol. Biol. 96 (2008) 19–43.
[3] R.S. Oliveira, B.M. Rocha, D. Burgarelli, W. Meira, R. Weber do Santos, A parallel accelerated adaptive mesh algorithm for the solution of electrical models of the heart, Int. J. High Perform. Syst. Archit. 4 (2) (2012).
[4] R. Bordas, B. Carpentieri, G. Fotia, F. Maggio, R. Nobes, J. Pitt-Francis, J. Southern, Simulation of cardiac electrophysiology on next-generation high-performance computers, Philos. Trans. A Math. Phys. Eng. Sci. 367 (2009) 1951–1969.
[5] S. Succi, The Lattice Boltzmann Equation: for Fluid Dynamics and Beyond, Oxford Science Publications, 2013.
[6] P. Neumann, T. Neckel, A dynamic mesh refinement technique for lattice Boltzmann simulations on octree-like grids, Comput. Mech. 51 (2013) 237–253.
[7] A.M. Artoli, A.G. Hoekstra, P.M.A. Sloot, Mesoscopic simulations of systolic flow in the human abdominal aorta, J. Biomech. 39 (2006) 873–884.
[8] S.P. Dawson, S. Chen, G.D. Doolen, Lattice Boltzmann computations for reaction–diffusion equations, J. Chem. Phys. 98 (1992) 1514–1523.
[9] R. Blaak, P.M.A. Sloot, Lattice dependence of reaction–diffusion in lattice Boltzmann modeling, Comput. Phys. Comm. 129 (2000) 256–266.
[10] R.M. Amorim, R.W. dos Santos, Solving the cardiac bidomain equations using graphics processing units, J. Comput. Sci. 4 (2013) 370–376.
[11] B.G. de Barros, R.S. Oliveira, W. Meira Jr., M. Lobosco, R.W. dos Santos, Simulations of complex and microscopic models of cardiac electrophysiology powered by multi-GPU platforms, Comput. Math. Methods Med. (2012).
[12] E. Bartocci, E.M. Cherry, J. Glimm, R. Grosu, S.A. Smolka, F.H. Fenton, Toward real-time simulation of cardiac dynamics, in: CMSB11: Proceedings of the 9th International Conference on Computational Methods in Systems Biology, 2011, pp. 103–112.
[13] B.M. Rocha, F.O. Campos, R.M. Amorim, G. Plank, R.W. dos Santos, M. Liebmann, G. Haase, Accelerating cardiac excitation spread simulations using graphics processing units, Concurr. Comput.: Pract. Exper. 23 (2011) 708–720.
[14] W. Wang, L. Xu, J. Cavazos, H.H. Huang, M. Kay, Fast acceleration of 2D wave propagation simulations using modern computational accelerators, PLoS One 9 (1) (2014) e86484.
[15] M. Schreiber, P. Neumann, S. Zimmer, H.-J. Bungartz, Free-surface lattice-Boltzmann simulation on many-core architectures, Procedia Comput. Sci. 4 (2011) 984–993.
[16] M. Geveler, D. Ribbrock, S. Mallach, D. Göddeke, A simulation suite for lattice-Boltzmann based real-time CFD applications exploiting multi-level parallelism on modern multi- and many-core architectures, J. Comput. Sci. 2 (2011) 113–123.
[17] S.A. Niederer, E. Kerfoot, A.P. Benson, M.O. Bernabeu, O. Bernus, C. Bradley, E.M. Cherry, R. Clayton, F.H. Fenton, A. Garny, E. Heidenreich, S. Land, M. Maleckar, P. Pathmanathan, G. Plank, J.F. Rodríguez, I. Roy, F.B. Sachse, G. Seemann, O. Skavhaug, N.P. Smith, Verification of cardiac tissue electrophysiology simulators using an N-version benchmark, Philos. Trans. A Math. Phys. Eng. Sci. 13 (2011) 4331–4351.
[18] J. Keener, J. Sneyd, Mathematical Physiology, Springer, 1998.
[19] C. Luo, Y. Rudy, A model of the ventricular cardiac action potential. depolarization, repolarization, and their interaction, Circ. Res. 68 (6) (1991) 1501–1526.
[20] K.H.W.J. ten Tusscher, A.V. Panfilov, Alternans and spiral breakup in a human ventricular tissue model, Am. J. Physiol. Heart Circ. Physiol. 291 (2006) H1088–H1100.
[21] A. Mahajan, Y. Shiferaw, D. Sato, A. Baher, R. Olcese, L.-H. Xie, M.-J. Yang, P.-S. Chen, J.G. Restrepo, A. Karma, A. Garfinkel, Z. Qu, J.N. Weiss, A rabbit ventricular action potential model replicating cardiac dynamics at rapid heart rates, Biophys. J. 94 (2008) 392–410.
[22] P.L. Bhatnagar, E.P. Gross, M. Krook, A model for collisional processes in gases I: small amplitude processes in charged and in neutral one-component systems, Phys. Rev. 94 (1954) 511.
[23] H. Yoshida, M. Nagaoka, Multiple-relaxation-time lattice Boltzmann model for the convection and anisotropic diffusion equation, J. Comput. Phys. 229 (2010) 7774–7795.
[24] X. Zhang, A.G. Bengough, J.W. Crawford, I.M. Young, A lattice BGK model for advection and anisotropic dispersion equation, Adv. Water Resour. 25 (2002) 1–8.
[25] I. Ginzburg, Equilibrium-type and link-type lattice Boltzmann models for generic advection and anisotropic-dispersion equation, Adv. Water Resour. (2005).
[26] S. Rush, H. Larsen, A practical algorithm for solving dynamic membrane equations, IEEE Trans. Biomed. Eng. 25 (1978) 389–392.
[27] J. Latt, How to implement your DdQq dynamics with only q variables per node (instead of 2q), Tech. Rep., Tufts University, 2007.
[28] K. Mattila, J. Hyvaluoma, T. Rossi, M. Aspnas, J. Westerholm, An efficient swap algorithm for the lattice Boltzmann method, Comput. Phys. Comm. 176 (2007) 200–210.
[29] M. Bernaschi, M. Fatica, S. Melchionna, S. Succi, E. Kaxiras, A flexible high-performance lattice Boltzmann GPU code for the simulations of fluid flows in complex geometries, Concurr. Comput.: Pract. Exper. 22 (1) (2010) 1–14.
[30] F.J. Vetter, A.D. McCulloch, Three-dimensional analysis of regional cardiac function: a model of rabbit ventricular anatomy, Prog. Biophys. Mol. Biol. 69 (1998) 157–183.
[31] E. Vigmond, M.H.G. Plank, L. Leon, Computational tools for modeling electrical activity in cardiac tissue, J. Electrocardiol. 36 (2003) 69–74.
[32] G.R. Mirams, C.J. Arthurs, M.O. Bernabeu, R. Bordas, J. Cooper, A. Corrias, Y. Davit, S.-J. Dunn, A.G. Fletcher, D.G. Harvey, M.E. Marsh, J.M. Osborne, P. Pathmanathan, J. Pitt-Francis, J. Southern, N. Zemzemi, D.J. Gavaghan, Chaste: an open source C++ library for computational physiology and biology, PLoS Comput. Biol. (2013).
[33] S. Rapaka, T. Mansi, B. Georgescu, M. Pop, G.A. Wright, A. Kamen, D. Comaniciu, LBM-EP: lattice-Boltzmann method for fast cardiac electrophysiology simulation from 3D images, in: Lecture Notes in Computer Science, MICCAI, 2012.