INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN BIOMEDICAL ENGINEERING
Int. J. Numer. Meth. Biomed. Engng. 2015; **00**:1–16
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cnm

Improved Hybrid/GPU Algorithm for Solving Cardiac Electrophysiology Problems on Purkinje Networks

M. Lange[1], S. Palamara[2], T. Lassila[1], C. Vergara[2], A. Quarteroni[3], A.F. Frangi[1]

[1]*CISTIB, Department of Electronic and Electrical Engineering, The University of Sheffield, UK*
[2]*MOX, Dipartimento di Matematica, Politecnico di Milano, Italy*
[3]*CMCS, Mathematics Institute of Computational Science and Engineering, École Polytechnique Fédérale de Lausanne, Switzerland*

SUMMARY

Cardiac Purkinje fibres provide an important pathway to the coordinated contraction of the heart. We present a numerical algorithm for the solution of electrophysiology problems across the Purkinje network that is efficient enough to be used in *in-silico* studies on realistic Purkinje networks with physiologically detailed models of ion exchange at the cell membrane. The algorithm is based on operator splitting and is provided with three different implementations: pure CPU, hybrid CPU/GPU, and pure GPU. Compared to our previous work, we modify the explicit gap junction term at network bifurcations in order to improve its mathematical consistency. Due to this improved consistency of the model, we are able to perform an empirical convergence study against analytical solutions. The study verified that all three implementations produce equivalent convergence rates, which shows that the algorithm produces equivalent result across different hardware platforms. Finally, we compare the efficiency of all three implementations on Purkinje networks of increasing spatial resolution using membrane models of increasing complexity. Both hybrid and pure-GPU implementations outperform the pure-CPU implementation, but their relative performance difference depends on the size of the Purkinje network and the complexity of the membrane model used. Copyright © 2015 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: cardiac electrophysiology; Purkinje networks; graphics processing units; hybrid algorithms

1. INTRODUCTION

The Purkinje fibres form an extensive branching network of fast conducting cells within the ventricular sub-endocardium of the human heart. This network covers large areas of the ventricles and ensures their rhythmic contraction, in response to signals traversing from the atrioventricular (AV) node along the His bundle and its branches [1, 6]. The fibres consist of specialised cardiac muscle cells that conduct a travelling electrical potential wave (the "action potential") towards the ventricular myocardium (MC) at the Purkinje-muscle junctions (PMJs).

Even if the function of Purkinje fibres is well-known, their involvement in arrhythmia is less well understood. It is suspected that the Purkinje network (PN) can generate an action potential (AP) spontaneously [4], or be essential to the initiation of some ventricular tachycardias (VT) [19, 3]. The PN can cause VT if a unidirectional block in the main bundle is present. Assuming the left bundle

*Correspondence to: Centre for Computational Imaging & Simulation Technologies in Biomedicine (CISTIB), Department of Electronic and Electrical Engineering University of Sheffield, Pam Liversidge Building, Mappin Street, Sheffield S1 3JD, UK.

branch has a unidirectional block with no orthodromic conduction, but slow antidromic conduction. Then the AP would be conducted through the right bundle towards the ventricular septum, from where it travels in the direction of the left ventricular PN and enters in the antidromic direction. In antidromic direction the AP can pass through the unidirectional block and arrives at the previously excited PN. If the loop is long enough or the conduction slow enough, the cells in the previously excited PN have passed their absolute refractory period. Thus, a new excitation can be generated. Equally unidirectional blocks downstream in the PN are believed to form loops and consequently generate VT [19]. In summary, the PN is potentially involved in the generation of arrhythmias.

However, it is challenging to investigate the effect of the PN on arrhythmias in a clinical setting. This is because the PN is very thin and fragile, which makes it difficult to perform imaging or *in-vivo* electrophysiological measurements. However, in recent years *in-silico* experiments have become feasible, and can provide information about the PN which are difficult to obtain from clinical settings [27, 10]. It has even been suggested to include the PN in cardiac simulations designed to aid interventions [27].

In the past, different models for the PN have been suggested and reviewed by Ten Tusscher and Panfilov in 2008 [26]. More recent work can be found in [32, 9, 11, 10, 12]. In such simulations, two main types of models have been used. The simpler eikonal model and the monodomain model.

The eikonal model has several limitations, the most significant of which is the absence of an ionic model. This renders the eikonal model impractical for re-entry simulations, or simulations of ischemic processes in the heart because the absolute and the relative refractory period of the cell cannot be simulated. Another limitation is the inability to simulate the physiological delay of 3ms to 12 ms from the PN to the MC at the PMJs. This delay has to be included in the model manually. Another recently observed limitation of the eikonal model has been the assumption of constant conduction velocity, which is not correct due to the push and pull effect [30]. Therefore, the eikonal model has a limited range of applications. Some of this limitations can be overcome if the monodomain model is used.

The monodomain model describes the propagation of the AP based on the ion exchange over the cell membrane. For this model certain observations have been made in conjunction with the PN. Most importantly, it has been shown that coarse networks are not able to reproduce physiological activations [27, 12, 9, 22]. A further indication that the PN needs a certain density comes from the fact that a single PMJ might fail to conduct the AP from the PN to the MC. However, if a large enough density is given, other PMJs will take over. Thus, to obtain realistic activation patters dense networks are required.

High complexity of the network or ionic model will reflect on the computational costs. Currently, it is possible to solve the monodomain equation on the densest of PN within hours, as we will show. For single simulations this might not be a strong limitation. However, to obtain information which is valid for a population, the computer simulations should be performed in different configurations representing this population. The leads to experiments with a variety of different configurations in shape, size, and possibly dysfunction of the hear. The variation in parameters leads to large studies, as our previous study [10] with more than 700 simulations. We call these type of studies, where all experiments are performed by a computer simulation, *in-silico* studies. In this studies the time demand for simulation of the PN activation with the monodomain model will exceed feasible limits. Therefore, a reduction of the computational time for the numeric solver is required. A reduction of the computational time can be achieved by processing the problem in parallel. High performance computing facilities offer this parallel computing power. However, the sparsity structure of the linear system resulting from the Purkinje problem causes a high communication demand for a one dimensional problem. This reduces the speed-up from using multiple CPUs. To overcome the scalability issue shared memory systems can be used. For the current task the graphics processing unit (GPU) poses a promising choice.

There are two major advantages of the GPU when used with the PN problem. 1) The GPU is a highly parallel shared memory system and 2) the electrophysiology problem in the PN is small enough to fit on one GPU. The first advantage means that a substantial reduction in computational time is expected, as more than 100 cores are processing at the same time. The second advantage relates to the fact that memory is often a limitation when using the GPU, as the memory size is small in comparison to a desktop machine. However, for the Purkinje system the common amount of 2GB memory for a GPU is sufficient to store the whole problem. Furthermore, the ionic models can be processed independently of each other. This makes the electrophysiology problem of the PN a good candidate for acceleration with the GPU.

Using new computational devices, like the GPU, can also introduce new errors. Validation of the method would be the best, however all Purkinje models currently lack *in-vivo* validation due to difficulties in measuring APs in the subendocardial Purkinje network. As such, the importance of numerical verification first and foremost has been identified as a key stepping stone to the wider development and acceptance of numerical models and methods for simulating Purkinje network activation [9].

Hence, in this study we present a verification of the numerical solution of the monodomain solution in one dimension. Our method of verification is the same as typically used for numerical solvers: the difference between an analytical solution and the numerical solution is computed. The numerical method used to solve the electrophysiology problem in the PN is based on the work of Vigmond and Clemens [33], and has been implemented in three different hardware configurations: pure-CPU, CPU/GPU hybrid, and pure-GPU.

Once the accuracy of the solutions has been verified, a second comparison is made concerning the performance of the implementations. Therefore, we create four different Purkinje fibre networks, where the smallest one has 6251 nodes and the largest one has 43748 nodes. The electrophysiology for these networks is simulated with two different ionic models: The model developed by di Francesco and Noble [7] and with the model of Stewart et al. [24].

2. NUMERICAL SCHEME FOR COMPUTING ACTION POTENTIALS IN 1-D NETWORKS

2.1. *Definition and modification of the explicit gap junction model*

To solve the action potential in the Purkinje fibres, we improve our previously published algorithm [11] based on [33], which is described below. It is based on the one-dimensional *monodomain equation*

$$\partial_x\left(\sigma_i \partial_x V_m\right) = \beta(C_m \partial_t V_m + I_{\text{ion}}(V_m, \xi)), \tag{1}$$

where V_m is the transmembrane potential, I_{ion} the total current through the ionic channels on the cell membrane, ξ are the state variables of the membrane model, β is the surface-to-volume ratio of the cell membrane, C_m the cell membrane capacitance, σ_i the intracellular conductivity tensor, and x the local spatial coordinate.

The one-dimensional cable equation needs to be extended to describe the propagation of the potential at the branching points. Each Purkinje branch is modelled as a separate problem on a one-dimensional line segment, which is then coupled to other branches by interface conditions determined by the continuity of potential and Kirchhoff's current law. For the latter, the input and output currents at the branching points are needed, which can either be obtained from a numerical derivative of the potential, or by the use of a finite difference approximation for the derivative. The numerical problem in this work is solved with a finite element (FE) scheme using Hermite basis functions, such that the derivatives of the solutions are degrees of freedom (DOF) in the formulation.

To close the monodomain equations approximated using Hermite basis functions, the concept of explicit *gap junction* models is introduced. This involves writing Kirchhoff's laws at the bifurcations explicitly in terms of the currents and potentials. Gap junctions are specialised intercellular connections between two Purkinje cells at their ends (see Fig. 1). In the classical monodomain model for the myocardium, the effect of these gap junctions is conflated into the conductivity tensor by a homogenisation process [5]. In the types of models considered for the PN in this work, all gap-junctions between two cells are homogenised into a single gap-junction. This gap-junction is then modelled explicitly [33, 29, 11]. Also the bifurcation is modelled based on the homogenisation.

This allows us to model bifurcations with a single gap-junction connecting three cells, in order to correctly capture the "push-and-pull" effect.

Let us consider the case of two Purkinje cells connected by a gap junction (inset of Fig. 1). Then the DOFs of the problem are the intracellular potential V_g and the current I_g across the gap junction, where I_g relates to V_g by the derivative.

$$I_g = \pi\rho^2\sigma_i^*\frac{\partial V_g}{\partial l}, \tag{2}$$

where σ_i^* is the equivalent conductivity defined later, ρ the Purkinje cell radius and l the cell length. Both quantities I_g and V_g are formally located in the middle of the gap junction (in red in Fig. 1). The ionic channel current I_{ion} is calculated in the cells at the ghost nodes (in blue in Fig. 1), which means for each node point the cell membrane model needs to be evaluated twice.

The relation between the intracellular potential ϕ_i in the gap junction and the transmembrane potential in the two neighbouring cells $V_\pm$ is given by Ohm's law

$$V_\pm = \phi_i - \phi_e \mp \frac{I_g R_g}{2},$$

where R_g is the gap junction resistance and ϕ_e is the extracellular potential. The latter is assumed to be constant throughout this work. Furthermore, up to a multiplicative factor I_g represents the derivative of ϕ_i by Ohm's law. The difference between intra- and extra cellular potential is the transmembrane potential $V_g = \phi_i - \phi_e$.

The values ϕ_i, I_g at the branching point are repeated in order to allow each segment to be solved separately. The three endpoints of the segments are then assumed to connect in the gap junction of the three cells (see Fig. 2). In contrast to our previously published method [11], each of the points gets only one cell membrane model associated with it instead of two. The single cell membrane model will then be solved in the corresponding cell segment. The currents are given from each cell to the branching point as indicated in Fig. 2. This adjustment is necessary because in our previous work there are six ghost nodes at each branching point, but only three actual cells. As we will see, the more concise formulation will also result in a more accurate numerical solution in comparison with analytical solutions.

Once the explicit gap junction model is in place, we need to ensure the compatibility of the macroscopic conductivity tensor with the gap junction resistance. Therefore, we assume that σ_i is the intracellular conductivity without any effect of gap junctions and introduce the equivalent conductivity under the assumption of a cylindrical volume conductor $\sigma_i^* = (\sigma_i\ell)/(\ell + \sigma_i R_g \pi\rho^2)$, where ℓ is the length of the Purkinje cell and ρ its radius. Note that this assumes that any discretisation has a step length $h = Z\ell$, which is an integer multiple Z of the cell length ℓ. In this notation (1) becomes

$$\partial_x\sigma_i^*\partial_x V_\pm = \beta(C_m\partial_t V_\pm + I_{\text{ion}}(V_\pm, \xi_\pm)). \tag{3}$$

To approximate the solution of (3) in time we introduce a time discretisation, where the superscript n refers to the numerical solution computed at time t^n. The algorithm to obtain the solution at t^{n+1} from the solution at t^n has four steps (Fig. 1), and uses of an operator splitting scheme as follows

$$\begin{cases} \partial_t V + L_1(V) = 0 \\ \partial_t V + L_2(V) = 0 \end{cases}, \tag{4}$$

where $L_1 = I_{\text{ion}}$, $L_2 = \partial_x(\sigma_i^*\partial_x)$ and V being one of the unknown potentials. The first three steps address the first equation, starting with Ohm's law to obtain the transmembrane potential $V_\pm^n$ in the cells from the potential V_g^n and current I_g^n at the gap junctions

$$V_\pm^n = (\phi_i^n - \phi_e^n) \mp \frac{I_g^n R_g}{2}.$$

The algorithm then proceeds to the second step, in which the membrane models are solved

$$V_\pm^{n+1/2} = V_\pm^n - \frac{I_{ion}(V_\pm^n, \xi)}{C_m}\Delta t.$$

The third step generates from the transmembrane potentials $V_{\pm}^{n+1/2}$ the current and potential in the gap junction $I_g^{n+1/2}, V_g^{n+1/2}$. The last step solves the operator equation $\partial_t V + L_2(V) = 0$ with the FE model, therefore we use the relation

$$V_g = \phi_i^{n+1/2} - \phi_e^{n+1/2} = \frac{V_+^{n+1/2} + V_-^{n+1/2}}{2}$$

and since the equations (3) are linear in $V_\pm$ we obtain:

$$\beta C_m \frac{(\phi_i^{n+1} - \phi_e^{n+1}) - (\phi_i^{n+1/2} - \phi_e^{n})}{\Delta t} = \partial_x \sigma_i \partial_x \phi_i^{n+1}. \quad (5)$$

In the FEM this translates into the following matrix formulation of the problem:

$$(\kappa M + K)\tilde{\boldsymbol{\phi}}_i^{n+1} = \kappa M(\tilde{\boldsymbol{\phi}}_i^{n} + (\tilde{\boldsymbol{\phi}}_e^{n+1} - \tilde{\boldsymbol{\phi}}_e^{n})), \quad (6)$$

where $\kappa = \beta C_m/\Delta t$, M is the mass matrix, K is the stiffness matrix and $\tilde{\boldsymbol{\phi}}_i$ is the vector of unknowns, which contains the potential and the derivative of the potential at each node. This is due to the fact that we are using Hermite basis functions, which include as a DOF also the value of the derivative in each node.

In the stiffness matrix of the FE model, the triplicated branching points of line segments are used to enforce the interface conditions and thus couple together the solutions obtained from the different line segments. In the case that segment 1 branches into segments 2 and 3, we enforce the continuity of the potential $\phi_1 = \phi_2 = \phi_3$ and the conservation of current $I_1 = I_2 + I_3$. Our implementation covers the case where segments 1 and 2 join to form segment 3 and thus giving our algorithm the ability to solve physiological networks with loops. In this case the coupling condition of the currents is $I_1 = I_3 - I_2$. These interface conditions are introduced in the FEM stiffness matrix associated to (6) and the right hand side.

2.2. *Hardware implementation*

In the following section, we detail the different characteristics of the three implementations. They are all performed using the `LifeV` library[†], which provides methods to assemble the FE stiffness and mass matrices and the right hand side coming from boundary conditions, time discretization and force terms. Furthermore, linear solvers and preconditioners are provided through the `Trilinos`[‡] linear algebra library.

In all implementations the linear system is solved with the generalised minimal residual method (GMRES) with incomplete LU (ILU) factorisation for preconditioning of the linear system. The GMRES method was used as the system matrices are not symmetric due to the coupling condition enforcement at the junctions. In each iteration the preconditioner is applied to the linear system. This is done by solving first for the lower triangular system and then solving for the upper triangular system.

To advance the ionic cell models in time an explicit forward Euler method has been used. The more efficient Rush-Larsen method could not be used, due to problems with the numerical stability of the Stewart model. This is also reflected in the stiffness of the resulting ordinary differential equation (ODE) system from the ionic models. As a result of the high stiffness, a timestep of 0.002ms for the Euler method was needed. This timestep is a factor of ten smaller than the timestep for the linear system of the diffusion part. Therefore, in each global timestep the ionic model has been solved ten times.

The pure CPU implementation was parallelised with the help of the OpenMPI framework, which allows in the proposed algorithm to perform Steps 1 to 3 in a distributed way with linear partitioning.

[†]The `LifeV` (http://www.lifev.org) finite element library is the joint collaboration between four institutions: EPFL, Politecnico di Milano, INRIA, and Emory University.
[‡]http://www.trilinos.org

The linear system is solved with one OpenMPI process to eliminate communication between CPUs while solving the linear system. Furthermore, the computationally most expensive step in the algorithm is Step 2. This implies that all other processes need to send their data to the serial process and after solving the problem the solution needs to be redistributed (see Fig. 3 for the workflow).

In the CPU/GPU hybrid implementation the membrane models are solved on the GPU. Therefore, before Step 2 the transmembrane potential is copied to the GPU, then the membrane model variables are updated, and the transmembrane potential is copied back from the GPU to the CPU. These three tasks are generated and queued in a CUDA streams, which allow for asynchronous GPU tasks. After the CPU has scheduled all task groups, it waits for their completion, and subsequently returns to Steps 3 and 4 on the CPU.

The third implementation does all the computation on the GPU, thus there is no memory copy between the steps. Steps 1 and 3 use the same code on the GPU as on the CPU, and in Step 2 we reuse the code from the hybrid implementation, but without the memory copy. To solve the linear system in Step 4, the mass matrix and the stiffness matrix are built on the CPU with the `LifeV` framework, and the resulting sparse matrices are copied to the GPU. The same is done for the preconditioner, which is built on the CPU and then copied to the GPU. The GMRES method on the GPU is the same as on the CPU, but uses `cuSPARSE` and `cuBLAS` for the matrix operation. Solving for the upper and lower triangular matrix in the preconditioner is optimised with the CUDA framework, which provides a parallel implementation for the solution process [17].

There are two different hybrid and GPU implementations in the performance test, which correspond to using different levels of floating point precision. GPUs are designed for single precision and thus have much higher number of floating point operations in single precision than in double precision mode. Therefore, we implemented the same GPU code using both double precision, and selectively dropping down to single precision where numerical stability was verified not to be affected. This is referred to as *mixed-mode*.

All computations were performed with a Dell Precision-WorkStation-T7500 featuring two Intel(R) Xeon(R) CPUs E5620 at 2.40GHz and a NVIDIA Quadro 4000 GPU with 256 CUDA Cores.

3. VERIFICATION OF THE PROPOSED NUMERICAL METHOD

Two verification tests are performed. The first evaluates the accuracy of the solution in equilibrium against an analytical solutions, and the second uses a travelling pulse solutions to verify the dynamic solutions.

3.1. Numerical error and convergence in equilibrium

In the first experiment we seek an equilibrium solution for the monodomain equation (3), of a simplified cell membrane model [28] given by

$$\partial_t V = pV, \tag{7}$$

where V is the transmembrane potential and p is a model parameter. Depending on p the solution is stable ($p < 0$), or it is exponentially unstable if ($p > 0$). The cell membrane model is then applied to two different test geometries. The first geometry is a finite one-dimensional line segment (Fig. 4, left) $D_1 = [-M, M]$, $M > 1$, which is divided in three subparts $D_{1,1} = [-M, -1]$, $D_{1,2} = [-1, 1]$, and $D_{1,3} = [1, M]$. In $D_{1,1}$ and $D_{1,3}$ the cell membrane model is chosen to be stable while in $D_{1,2}$ it is unstable

$$p(x) = \begin{cases} p_1 & \text{for } x \in D_{1,2} \\ -p_2 & \text{else where} \end{cases},$$

where $p_i > 0$. The simplified cell membrane model is then introduced in the monodomain equation (1). Letting $\delta = \sigma_i/\beta$ and, assuming that the conductivity σ_i has no spatial dependency, the problem to be solved becomes

$$\begin{array}{rclcrcl} & & C_m\partial_t V = \delta\partial_x^2 V - p(x)V, & & & & \\ V_1(-1) & = & V_2(-1) & , & V_2(1) & = & V_3(1), \\ V_1'(-1) & = & V_2'(-1) & , & V_2'(1) & = & V_3'(1), \\ V_1(-M) & = & 0 & , & V_3(M) & = & 0. \end{array} \tag{8}$$

The solution can be deduced with the canonical *ansatz* $V_i(x) = c_1 \exp(kx) + c_2 \exp(kx)$, as shown by Artebrant *et al.* [2]. For a line segment the solution is:

$$V(x) = \begin{cases} \sinh(\kappa(M+x)) & , \quad x \in D_{1,1} \\ d\cos(kx) & , \quad x \in D_{1,2} \\ \sinh(\kappa(M-x)) & , \quad x \in D_{1,3} \end{cases}, \tag{9}$$

where $d = \sinh(\kappa(M-1))/cos(k)$ and parameters $\kappa = \sqrt{p_1/\delta}$ and $k = \sqrt{p_2/\delta}$. To satisfy the differentiability conditions $V_1'(x)|_{x=-1} = V_2'(x)|_{x=-1}$, $V_2'(x)|_{x=1} = V_3'(x)|_{x=1}$, in (8) the relation

$$k\tan(k) = \frac{\kappa}{\tanh(\kappa(M-1))}$$

must hold.

The second problem is a symmetric domain D_2 with a branching and joining point (Fig. 4, right). The domain consists of five line segments, the first two, $D_{2,1} = [-M,-1]$ and $D_{2,2} = [-M,-1]$, join the segment $D_{2,3} = [-1,1]$, which branches into two further segments $D_{2,4} = [1,M]$ and $D_{2,5} = [1,M]$. As in the first domain, the middle segment $D_{2,3}$ has unstable cells while the outer branches $D_{2,1}, D_{2,2}, D_{2,4}$, and $D_{2,5}$ are stable. The problem is symmetric at zero, so we will look at the negative domain only. Furthermore, $D_{2,1}$ and $D_{2,2}$ are equal, thus it is sufficient to find the solution on one of them. This means we need to solve the following problem

$$\begin{array}{c} \delta V_1'' - p_1 V_1 \quad = 0 \quad \forall\, x \in D_{2,1} \\ \delta V_3'' + p_2 V_3 \quad = 0 \quad \forall\, x \in D_{2,3} \\ V_1(-1) = V_3(-1),\ 2V_1'(x)|_{x=-1} = V_3\prime(x)|_{x=-1},\ V_1(-M) = 0 \end{array},$$

where the first two equations are due to Kirchoff's current law, which states that the current sum of the first and second branch need to equal the third branch.

Following an exponential *ansatz*, we constrain the solution to be unique by choosing the maximum amplitude $V(0) = 1$, which leads to

$$\begin{aligned} V_{1,2} &= c_1 \sinh(\kappa(M+x)), \\ V_3 &= \cos(kx), \\ V_{4,5} &= c_1 \sinh(\kappa(M-x)), \end{aligned} \tag{10}$$

where $c_1 = \cos(-k)/\sinh(\lambda_1(M-1))$ and the relation between κ and k changes to

$$k\tan(k) = \frac{2\kappa}{\tanh(\kappa(M-1))}.$$

3.1.1. Numerical solution in equilibrium For the numerical solution we choose $\delta = 1$, which imposes the condition $1 = \sigma^*/\beta$. With a physiological cell length of $\ell = 62.5$ μm, diameter 16.0 μm, and chosen gap-junction resistance $R = 0.1$ kΩ, the intracellular conductivity becomes 1967.5 kS/cm. Furthermore, the spatial step size h is chosen as an integer multiples of the cell length ℓ.

On the line D_1 we choose parameters $p_2 = 0.0946441$, $M = 20$, and the capacitance of the cell membrane $C_m = 1$ μF. For the branching domain D_2 we choose $M = 10$, $p_2 = 1$, and $\delta = 1$.

The resulting error distribution over the line and the branching domain is shown in Fig. 5, where the largest contribution of the error comes from the passive cell region. The convergence test shows that with decreasing spatial step size the L_2-error reduces faster than linearly for the single-interval domain D_1 (Fig. 5) and linearly for the example in the branching domain D_2. Note that without the modification introduced in Sect. 2.1 only sub-linear convergence behaviour was obtained for the branching domain case (compare with Fig. 3 from [11]), indicating that the modification is required for the accuracy of the numerical method.

3.2. Analytical solution for a travelling pulse

In this section the convergence of the dynamic solution is investigated. Therefore, an analytic solution for a travelling wave of the linearised FitzHugh-Nagumo (FHN) equation [20, 15] is constructed and then solved numerically.

For the construction of the solution to the FHN model we follow the work of Rinzel and Keller [20]. They used the two variable model, with the transmembrane potential V and the recovery variable w

$$\begin{aligned} \partial_t V = I_{ion} &= f(V) + w \\ \partial_t w &= bV \\ f &= V(a-V)(1-V) \end{aligned} \quad . \tag{11}$$

f can be linearised to $f = V - H(V-a)$, with $0 \le a \le 1/2$ and H is the Heaviside function. The solution of the linearised problem (11) on an infinite line is well-known [8, 20].

We consider the monodomain equation (3) over an infinite line under the assumption that σ_i^* does not depend on x and couple it to the linearised FHN cell membrane model

$$\begin{aligned} C_m \partial_t V &= \tfrac{\sigma_i^*}{\beta}\partial_x^2 V - f(V) - w \\ \partial_t w &= bV && , b \ge 0 \\ f(V) &= V - H(V-a) && , 0 \le a \le 1/2 \end{aligned} \quad . \tag{12}$$

To be in the condition of the approach in [20] we assume in the following $\sigma_i^*/\beta = 1$ and $C_m = 1$.

By differentiating the first equation in (12) with respect to time, the system can be rewritten in one equation

$$\begin{aligned} \partial_t^2 V &= \partial_t \partial_x^2 V - \partial_t f(V) - \partial_t w, \\ \Rightarrow \partial_t^2 V &= \partial_t \partial_x^2 V - \partial_t f(V) - bV. \end{aligned} \tag{13}$$

To solve this problem the travelling wave *ansatz* $V(x,t) = v_c(z)$ where $z = x + ct$ with $c > 0$ is introduced. Furthermore, we assume that $v_c(0) = a$ and $\lim_{|z|\to\infty} v_c(z) \to 0$, and from the intermediate value theorem follows the existence of a $z_1 \neq 0$ with $v_c(z_1) = a$. The system to be solved can be rewritten as

$$\begin{aligned} c^2 v_c'' &= c v_c''' - c f'(v_c) v_c' - bv \\ 0 &= v_c''' - c v_c'' - f'(v_c) v_c' - (b/c) v \\ 0 &= \begin{cases} v_c''' - c v_c'' - v_c' - (b/c) v & \forall z \in \mathbb{R}\backslash\{0, z_1\} \\ v_c''' - c v_c'' - (b/c) v & z \in \{0, z_1\} \end{cases} \end{aligned} \quad , \tag{14}$$

with boundary condition $\lim_{|z|\to\infty} v_c(z) \to 0$ and where $'$ indicates a derivative with respect to z. The solution can be obtained in the three regions $z < 0$, $0 \le z \le z_1$ and $z > z_1$. Following an exponential ansatz for the differential equation we need to find the roots of the cubic polynomial

$$p(\lambda) = \lambda^3 - \lambda^2 - \lambda - (b/c). \tag{15}$$

If the discriminant is non-negative there are three distinct real solutions, while for a negative discriminate two of the solutions are complex. Let λ_1 be the positive real solution while λ_2 and λ_3 are the possible complex solutions. Then, the solution to the differential equation (14) is [8]

$$v_c = \begin{cases} a\exp(\lambda_1 x) & z < 0 \\ (a - p'(\lambda_1))^{-1}\exp(\lambda_1 x) - p'(\lambda_2)^{-1}\exp(\lambda_2 x) - p'(\lambda_3)^{-1}\exp(\lambda_3 x) & 0 \le z \le z_1 \\ p'(\lambda_2)^{-1}(\exp(-\lambda_2 z_1) - 1)\exp(\lambda_2 x) + p'(\lambda_3)^{-1}(\exp(-\lambda_3 z_1) - 1)\exp(\lambda_3 x) & z > z_1. \end{cases} \tag{16}$$

In the sequel, we show that v_c is real, even with complex eigenvalues λ_2, λ_3. We use that $\mathrm{Re}(\lambda_2) = \mathrm{Re}(\lambda_3)$ and $\mathrm{Im}(\lambda_2) = -\mathrm{Im}(\lambda_3)$, where $i = \sqrt{-1}$.

$$\Rightarrow v_c = \begin{cases} a\exp(\lambda_1 x) & z < 0 \\ \frac{\exp(\lambda_1 x)}{(a-p'(\lambda_1))} - (\frac{e^{(i\mathrm{Im}(\lambda_2)x)}}{(p'(\mathrm{Re}(\lambda_2)+i\mathrm{Im}(\lambda_2))} + \frac{e^{(-i\mathrm{Im}(\lambda_2)x)}}{p'(\mathrm{Re}(\lambda_2)-i\mathrm{Im}(\lambda_2))})e^{(\mathrm{Re}(\lambda_2)x)} & 0 \le z \le z_1 \\ \frac{e^{\lambda_2(x-z_1)}-e^{(\lambda_2)x}}{p'(\lambda_2)} + \frac{e^{\lambda_3(x-z_1)}-e^{\lambda_3 x}}{p'(\lambda_3)} & z > z_1 \end{cases}, \tag{17}$$

$$\Rightarrow v_c = \begin{cases} a\exp(\lambda_1 x) & z < 0 \\ \frac{\exp(\lambda_1 x)}{(a-p'(\lambda_1))} - (\frac{\alpha-i\beta}{\alpha^2+\beta^2}e^{(i\mathrm{Im}(\lambda_2)x)} + \frac{\alpha+i\beta}{\alpha^2+\beta^2}e^{(-i\mathrm{Im}(\lambda_2)x)})e^{(\mathrm{Re}(\lambda_2)x)} & 0 \le z \le z_1 \\ \frac{e^{\lambda_2(x-z_1)}}{p'(\lambda_2)} + \frac{e^{\lambda_3(x-z_1)}}{p'(\lambda_3)} - \left(\frac{e^{(\lambda_2)x}}{p'(\lambda_2)} + \frac{e^{\lambda_3 x}}{p'(\lambda_3)}\right) & z > z_1 \end{cases}, \tag{18}$$

$$\alpha = 3(\mathrm{Re}(\lambda_2)^2 - (\mathrm{Im}(\lambda_2)^2) - 2\mathrm{Re}(\lambda_2) - 1,$$
$$\beta = 6(\mathrm{Re}(\lambda_2))(\mathrm{Im}(\lambda_2)) - 2(\mathrm{Im}(\lambda_2)).$$

Note that $\alpha, \beta \in \mathbb{R}$. Then applying Euler's formula

$$(a+bi)e^{-ci} + (a-bi)e^{ci} = 2(a\cos(c) + b\sin(c)) \qquad a, b, c \in \mathbb{R}. \tag{19}$$

to write the formula in the region $0 \le z \le z_1$ as real expression

$$\Rightarrow v_c = \begin{cases} a\exp(\lambda_1 x) & z < 0 \\ \frac{\exp(\lambda_1 x)}{(a-p'(\lambda_1))} - 2(\frac{\alpha}{\alpha^2+\beta^2}\cos(\mathrm{Im}(\lambda_2)x) + \frac{\beta}{\alpha^2+\beta^2}\sin(\mathrm{Im}(\lambda_2)x))e^{(\mathrm{Re}(\lambda_2)x)} & 0 \le z \le z_1 \\ \frac{e^{\lambda_2(x-z_1)}}{p'(\lambda_2)} + \frac{e^{\lambda_3(x-z_1)}}{p'(\lambda_3)} - \left(\frac{e^{(\lambda_2)x}}{p'(\lambda_2)} + \frac{e^{\lambda_3 x}}{p'(\lambda_3)}\right) & z > z_1 \end{cases}, \tag{20}$$

$$\alpha = 3(\mathrm{Re}(\lambda_2)^2 - (\mathrm{Im}(\lambda_2)^2) - 2\mathrm{Re}(\lambda_2) - 1,$$
$$\beta = 6(\mathrm{Re}(\lambda_2))(\mathrm{Im}(\lambda_2)) - 2(\mathrm{Im}(\lambda_2)).$$

For the term in $z > z_1$ we apply twice the steps we used in the region $0 \le z \le z_1$, which results in the real expression

$$v_c = \begin{cases} a\exp(\lambda_1 x) & z < 0 \\ \frac{\exp(\lambda_1 x)}{(a-p'(\lambda_1))} - 2(\frac{\alpha}{\alpha^2+\beta^2}\cos(\mathrm{Im}(\lambda_2)x) + \frac{\beta}{\alpha^2+\beta^2}\sin(\mathrm{Im}(\lambda_2)x))e^{(\mathrm{Re}(\lambda_2)x)} & 0 \le z \le z_1 \\ 2(\frac{\alpha}{\alpha^2+\beta^2}\cos(\mathrm{Im}(\lambda_2)(x-z_1)) + \frac{\beta}{\alpha^2+\beta^2}\sin(\mathrm{Im}(\lambda_2)(x-z_1)))e^{(\mathrm{Re}(\lambda_2)(x-z_1))} - & z > z_1 \\ \quad -2(\frac{\alpha}{\alpha^2+\beta^2}\cos(\mathrm{Im}(\lambda_2)x) + \frac{\beta}{\alpha^2+\beta^2}\sin(\mathrm{Im}(\lambda_2)x))e^{(\mathrm{Re}(\lambda_2)x)} & \end{cases} \tag{21}$$

Rintzel and Keller showed that the above solution (16) holds true only if the parameter a satisfies the relation with the parameters b and c, which we outline in the following. The relation assumes we know the eigenvalues λ_i $i = 1, 2, 3$ for given b, c, which then define the function

$$f(s) := 2 - s + \frac{p'(\lambda_1)}{p'(\lambda_2)}s^{(-\lambda_2/\lambda_1)} + \frac{p'(\lambda_1)}{p'(\lambda_3)}s^{(-\lambda_3/\lambda_1)}.$$

The root s_0 of the function f defines

$$a = \frac{1-s_0}{p'(\lambda_1)}. \tag{22}$$

This relation can be satisfied for any b with at most two c_i, where $c_1 \le c_2$. The slow pulse c_1 is an unstable solution, while c_2 is a stable solution [8, 20]. To obtain the value of z_1 for the given set of parameter a, b, c the following equation needs to be solved

$$\exp(-\lambda_1 z_1 s_0) = 1 - ap'(\lambda_1). \tag{23}$$

3.2.1. Numerical simulation of the travelling wave For the verification of the dynamic solution we use the solution for $a = 0.2250646$ mV, $c = 1.2$ cm/ms , $b = 0.2$ and $z_1 = 6.63395$ cm. In the numerical problem the parameter $\sigma_i = 1967.5, \beta = 1$, $R = 0.1$ kΩ and $C_m = 1$ μF are used. We use the solution (16) to initialise our numerical solution at the time 0 ms on a line of length 160 cm, and origin at 85 cm. With these values the wave exits the domain at 50 ms. The final time is chosen such that the wave in the numerical simulation stabilises in shape and then propagates for about 20 ms. All simulation use a temporal time step of 0.001 ms.

The first experiment was performed for spatial resolution of 0.00625 mm, where the L^2-error was calculated at each time step and plotted against the time for all three solvers (Fig. 6, top, left)). For the first time steps the error increases slower compared to the error increase after about 20 ms. Thereafter, a linear increase of the error is observed. This can be explained with the plot of the L^2-norm of the solution (Fig. 6, top, right), which is changing until 20 ms and thereafter can be considered as constant. The changes are due to the fact that the maximal amplitude of the wave is changing. The stable, slightly larger pulse, has after 20 ms a higher conduction velocity of about $c = 1.20132$ m/s, which is responsible for the linearly increasing error over time.

We conclude the verification with a convergence test in the L_2 error and the conduction velocity for the dynamic simulation. Therefore, the L_2 error and the conduction velocity after 40 ms have been evaluated for different step discretisations $h = \{1$ mm, 0.5 mm, 0.25 mm, 0.125 mm, 0.0625 mm$\}$. The L_2 error converges superlinearly (Fig. 6, bottom, left). More importantly, the conduction velocity approaches the theoretical value of 1.2 m/s at a step size of 0.0625 mm (Fig. 6, bottom, right). Again, the improved method of Sect. 2.1 exhibits proper convergence to the exact conduction velocity.

4. COMPUTATIONAL EFFICIENCY

We next evaluate the computational performance of the different solver implementations on problems of varying size and complexity. The number of DOFs is therefore varied either by increasing the complexity of the Purkinje fibre network (spatial complexity), or by the switching to a more complex cell membrane model (model complexity).

Four different Purkinje networks of varying levels of detail are considered, all of which are generated with the fractal rule presented in [23]. In order of increasing complexity, the first Purkinje network consists of the main Purkinje branches only, the second one has another level of branching giving a physiological covering of the LV, and the third network has another level of Purkinje branches added to increase the density of the end-junctions, resulting in a physiological network for the left ventricle. The fourth case is a dense Purkinje network for both the left and right ventricles. All Purkinje networks are discretised with a spatial resolution of $0.1\,\mathrm{mm}$ and are generated without loops for compatibility with other solvers (see Fig. 7).

Two different cell membrane models were used to test the influence of model complexity. The first and simpler Di Francesco-Noble model [7], which has been used in previous works [33], has 15 state variables. The model has been obtained from the CellML database and used without modification to the initial states or constants. The second membrane model used here has been published by Stewart et al. [24], and is based on modifications to the ten Tusscher-Panfilov model, and has 20 state variables. The model was obtained from the CellML repository, although the initial conditions were set to the values stated in Appendix Tab. II. The change in initial conditions was made to avoid the early self-excitation that is present in Purkinje cells but should not manifest itself under physiological conditions. For both membrane models a cell length of 0.01 mm, cell radius of 0.005mm, and an intracellular conductivity of 40 Ω^{-1}cm^{-1} were assumed, where the last two values were chosen to obtain realistic conduction velocities in the range between 3 m/s and 4 m/s. The gap junction resistance was chosen as 500 kΩ. For the simulation a temporal step size of 0.01 ms has been used and the simulation was run for 50 ms, after which all networks were fully depolarised. The meshes corresponding to the Purkinje networks can be downloaded as supplementary material, including the local activation times as supplemental material.

In the pure CPU implementation eight processes are run in parallel, while the hybrid and GPU implementations are run using one CPU process. The simulations were run in two different configurations on the GPU, the first in double precision, while the second was in single precision. For all simulations, we measure the time spent on setting up the problem, which includes reading the mesh and assembling the matrices and preconditioners. We report the time needed for solving the ionic membrane models and the diffusion equation in Fig. 7.

As expected the hybrid and pure GPU implementations are faster than the pure CPU implementation (Fig. 7), and a further speed-up is observed moving from double precision to single precision in the pure GPU implementation. The reason for the speed-up with the single precision in the pure GPU implementation is that the particular GPU used has roughly twice the number of floating point operations in single precision than it has in double precision. The reduction of computational time achieved with the hybrid implementation was limited. One possible reason for this might be that the particular GPU used is able to handle the entire double precision problem without full occupation. The second reason might be that the transmembrane potential needs to be converted from double precision to single precision, which is done in serial on the CPU.

The amount of time needed to solve the reaction part of the problem varies considerably between the pure CPU, hybrid, and pure GPU implementations. The pure CPU implementation is always the slowest, but the hybrid implementation performs more favourably on less complex membrane models, while the pure GPU implementation performs better with more complex membrane models. This is much more evident in the single precision versions. A possible reason for this can be found in the workflow of the hybrid and pure GPU implementations (Fig. 3), where a memory copy from the GPU to the CPU takes place in each time step of the hybrid implementation. In the pure GPU implementation this is unnecessary because values are used on the GPU only. This explains why the GPU implementation performs better with increasing complexity of the membrane model.

Solving the diffusion step with the pure GPU implementation is nearly always the slowest. We note that the hybrid implementation is faster than the pure CPU, as in the CPU implementation the transmembrane potential and the current need to be sent from all the OpenMPI nodes to the master node and the results communicated back. The linear system itself is solved in the same way in the pure CPU and the GPU/CPU hybrid cases. While in the pure GPU implementation the same algorithm is used, but the matrix operations are performed on the GPU. For small Purkinje systems, meaning very sparse and small matrices, the performance of the pure GPU implementation is behind the pure CPU and hybrid implementations. With increasing spatial complexity the pure GPU performance becomes better compared to the pure CPU performance, which likely is related to the size of the problem. Due to the overhead introduced by each CUDA operation, for very small problem sizes the benefits of GPU parallelism are lost.

After comparing between our various implementations, we compare our results with previously published studies (Tab. I). We survey the speed-up for the reaction part only, because no comparable speed-up was studied for solutions on 1-D networks. Most studies compare the speed-up of a single GPU against a single CPU core. Therefore, we performed our simulation of the bi-ventricular PN on one core with the di Francesco-Nobel model. It took 13497s to solve the reaction part. This means that the pure-GPU formulation was 123 times faster in mixed precision mode and in double precision mode it was 30 times faster. We compare our results against four different studies, which use an explicit solving schema and report their speed-ups. The first study is conducted by Mena and Rodrigues [16] and investigates the speed-up for different number of nodes in the mesh. Their results are based on the ionic model by Ten Tusscher and Panfilov [25], which is integrated with the Rush-Larson method. The best speed-up achieved was 120, where a single CPU core is compared against one NVIDIA Tesla M2090 GPU with 512 CUDA cores. This is the highest speed-up reported in our comparison, and our single precision implementation performs slightly better. The double precision mode is out performed by the study of Mena and Rodrigues, as well as by the study of Rocha et al. [21]. In their study the Luo and Rudy [13] model was solved by a Euler forward method. The usage of a GPU with 480 CUDA cores resulted in a speed-up of 51 over one CPU. The remaining two studies reported smaller speed-ups than our double precision implementation achieved. Vigmond et al. [31] compared the speed-up for the cell model by Mahajan et al. [14]. For

Study	# CUDA cores	Speed-up	Degrees of freedom	Ionic model	Temporal step size [ms]
Mena [16]	512	120	1 900 000	TP06	0.020
Rocha[21]	240	51	824 328	LR	0.010
Vigmond [31]	240	11	27 000 600	Mahajan	0.025
Neic [18]*	448	12	116 100 000	Mahajan	0.025
Hybrid	256	123	1 387 410	DFN	0.002
Full GPU	256	30	1 387 410	DFN	0.002

Table I. Comparison of the speed-up in solving the ionic model with previous studies. The speed-up are based on one CPU core, against one GPU. * obtained using 6 CPUs and 6 GPUs. **TP06** model by Ten Tusscher and Panfilov [25], **LR** Model by Luo and Rudy [13], **DFN** model by di Francesco and Noble [7].

integration the Rush-Larsen method was employed where possible and otherwise the Euler forward or Runge-Kutta method. This allowed them to reduce the computational time by a factor of about 11, when using a single GPU with 240 CUDA cores. In the work of Neic et al. [18] a larger whole ventricular simulation with the Mahajan ionic model is performed. Therefore, they employed a minimum of six GPU with 448 CUDA cores each, and compared it against an implementation with six CPU. The reported timings for the time spent solving the ODE system shows a reduction by a factor of 12. Overall, our single precision implementation has the best speed-up of all studies, but also our double precision implementation shows an average improvement for the speed-up.

5. CONCLUSION

We have presented an improved parallel algorithm for solving the monodomain cardiac electrophysiology equations on one-dimensional branching Purkinje fibre networks that is suitable for simulating activation on realistic Purkinje fibre networks in human-size hearts. We then developed a verification scheme of the numerical solution, which was applied to our three different implementations: pure-CPU, pure-GPU, and hybrid. Finally, we compared the performance of the implementations.

The verification of our first implementation [11] of the original proposed algorithm by Vigmond and Clemens [33] showed sublinear convergence in the L_2-error for branching fibres. To improve the convergence in this work, we have described a new explicit gap junction formulation, which is also more consistent in terms of the connection between multiple Purkinje fibre endpoints. Furthermore, a modification to the effective conductivity tensor was needed to ensure mathematical compatibility with the new formulation. Both improvements together led to a linear convergence as demonstrated in our verification study. Our results also showed convergence of the conduction velocity in the numeric solution towards its theoretical value. Importantly, there were no notable differences in convergence rates between the three different implementations.

After establishing that all three implementation achieved the same accuracy, the selection of an implementation is based on the relative computational performances of each implementation. Our first observation was that GPU based methods outperformed the implementation of eight parallel CPUs. The largest benefit of the parallel pure-GPU implementation was obtained either when a fully detailed biventricular (spatially complex) network was used, or when sufficiently complex membrane models were used, such as the model proposed by Stewart et al. 2009 considered in this study. For simpler LV-only models, or when using simpler membrane models, such as the Di Francesco-Noble model considered in this study, the hybrid implementation may be more attractive. In either case, the benefits of GPU-accelerated computation of action potentials in the fast conduction system have been demonstrated.

The implementation proposed in this study, results in a substantial reduction of computational time over conventional multi-processor implementations. Moreover, it relies on GPU hardware which is widely available. Thus, there is no need to use less available or expensive HPC facilities.

The speed-up gained by the GPU implementation enables running more simulations in less time. Therefore, it is more feasible to perform *in-silico* simulation, where large numbers of experiments are required.

The PN model can be combined with different models of the myocardium to conduct simulations of the ventricular activation. Currently, the feasible model for in-silico studies is the eikonal model of activation time. However, our model of the PN can also be coupled with monodomain or bidomain approach for the myocardium [30].

Future work could involve incorporating similar algorithms for the bidomain equations to explore e.g. the effect of defibrillation on Purkinje fibres. Currently ongoing work aims at extending the Purkinje fibre model to account for the presence of ischemia in the heart. We hope to have convinced the cardiac modelling community that large-scale numerical simulation of action potentials in the Purkinje fibre system is both feasible and important.

ACKNOWLEDGEMENT

Matthias Lange is funded by a PhD scholarship from the Electrical and Electronic Engineering Department at the University of Sheffield. Simone Palamara has been funded by “Fondazione Cassa di Risparmio di Trento e Rovereto” (CARITRO) within the project “Numerical modelling of the electrical activity of the heart for the study of the ventricular dyssynchrony”. Christian Vergara and Alfio Quarteroni has been partially supported by the Italian MIUR PRIN09 project no. 2009Y4RC3B_001. The research of Alfio Quarteroni was also supported by the Swiss National Science Foundation (SNF), project no. 140184.

APPENDIX

Table II. Initial conditions used for the Stewart et al. 2009 model

V	Transmembrane potential [mV]	-75.6095
K_i	Potassium dynamics [mMol]	136.757
Na_i	Sodium dynamics [mMol]	0.80211
Ca$_i$	Intracellular calcium [mMol]	1.47164e-4
y	y gate	0.00780153
X_{r1}	Rapid time dependent potassium current	0.382558
X_{r2}	Rapid time dependent potassium current	0.37373
X_s	Slow time dependent potassium current	3.85284e-2
m	m gate	1.24135e-2
h	h gate	0.361832
j	j gate	0.102063
Ca$_{ss}$	Calcium dynamics [mMol]	5.49319e-4
d	L type Ca current d	1.21585e-4
f	L type Ca current f	0.611603
f_2	L type Ca current $f2$	0.861484
f_{cass}	L-type Ca current	0.985735
s	Transient outward current s	0.925862
r	Transient outward current r	6.46602e-4
Ca$_{SR}$	Calcium in sarcoplasmic reticulum [mMol]	3.17519
R_{prime}	Calcium dynamics	0.851882

REFERENCES

1. A. Ansari, S.Y. Ho, and R.H. Anderson. Distribution of the Purkinje fibres in the sheep heart. *Anat. Rec.*, 254(1):92–97, 1999.
2. R. Artebrant, A. Tveito, and G.T Lines. A method for analyzing the stability of the resting state for a model of pacemaker cells surrounded by stable cells. *Math. Biosci. Eng.*, 7(3):505–526, 2010.
3. F. Bogun, E. Good, S. Reich, D. Elmouchi, P. Igic, D. Tschopp, S. Dey, A. Wimmer, K. Jongnarangsin, H. Oral, A. Chugh, F. Pelosi, and F. Morady. Role of Purkinje fibers in post-infarction ventricular tachycardia. *J. Am. Coll. Cardiol.*, 48(12):2500–2507, 2006.
4. .e a. Boyden, C. Barbhaiya, T. Lee, and H. E D J Ter Keurs. Nonuniform Ca2+ transients in arrhythmogenic Purkinje cells that survive in the infarcted canine heart. *Cardiovasc. Res.*, 57:681–693, 2003.
5. P Colli Franzone, L.F. Pavarino, and S. Scacchi. *Mathematical cardiac electrophysiology*, volume 13 of *Modeling, Simulation & Applications*. Springer Cham Heidelberg New York Dordrecht London, 2014.
6. L.L. Cooper, K.E. Odening, M.-S. Hwang, L. Chaves, L. Schofield, C.A. Taylor, A.S. Gemignani, G.F. Mitchell, J.R. Forder, B.-R. Choi, and G. Koren. Electromechanical and structural alterations in the aging rabbit heart and aorta. *Am. J. Physiol. Heart Circ. Physiol.*, 302(8):H1625–H1635, 2012.
7. D. DiFrancesco and D. Noble. A model of cardiac electrical activity incorporating ionic pumps and concentration changes. *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, 307(1133):353–398, 1985.
8. J. Keener and J. Sneyd. *Mathematical Physiology I: Cellular Physiology*, volume 8(I) of *Interdisciplinary Applied Mathematics*. Springer New York, New York, NY, 2nd edition, 2009.
9. S Krishnamoorthi, L. E. Perotti, N. P. Borgstrom, O. A. Ajijola, A. Frid, A. V. Ponnaluri, J. N. Weiss, Z. Qu, W. S. Klug, D. B. Ennis, and A. Garfinkel. Simulation Methods and Validation Criteria for Modeling Cardiac Ventricular Electrophysiology. *PLoS One*, 9(12):e114494, 2014.
10. M. Lange, L. Y. Di Marco, K. Lekadir, T. Lassila, and A. F. Frangi. Protective Role of False Tendon in Subjects with Left Bundle Branch Block: A Virtual Population Study. *PLoS One*, 11(1):e0146477, jan 2016.
11. M. Lange, S. Palamara, T. Lassila, C. Vergara, A. Quarteroni, and A.F. Frangi. Efficient numerical schemes for computing cardiac electrical activation over realistic purkinje networks: method and verification. *In: H. van Assen, P. Bovendeerd, T. Delhaas (Eds.) Proc. 8th Int. Conf. Functional Imag. Model. Heart (FIMH 2015), June 25–27, Maastricht*, 2015.
12. T. Lassila, M. Lange, A. R. Porras Perez, K. Lekadir, X. Albà, G. Piella, and A. F. Frangi. Electrophysiology Model for a Human Heart with Ischemic Scar and Realistic Purkinje Network. pages 90–97. 2016.
13. C H Luo and Y Rudy. A dynamic model of the cardiac ventricular action potential. I. Simulations of ionic currents and concentration changes. *Circ. Res.*, 74(6):1071–1096, 1994.
14. A. Mahajan, Y. Shiferaw, D.e Sato, A. Baher, R.o Olcese, L-H Xie, M-J Yang, P-S Chen, J. G. Restrepo, A. Karma, A. Garfinkel, Z. Qu, and J. N. Weiss. A rabbit ventricular action potential model replicating cardiac dynamics at rapid heart rates. *Biophys. J.*, 94(2):392–410, 2008.
15. H.P McKean Jr. Nagumo's equation. *Adv. Math.*, 4(3):209–223, 1970.
16. A. Mena and J.F. Rodriguez. Using graphic processor units for the study of electric propagation in realistic heart models. In *IEEE Computing in Cardiology*, volume 39, pages 37–40, 2012.
17. M. Naumov. Parallel solution of sparse triangular linear systems in the preconditioned iterative methods on the GPU. Technical Report NVR-2011-001, NVIDIA, 2011.
18. A Neic, M Liebmann, E Hoetzl, L Mitchell, E J Vigmond, G Haase, and G Plank. Accelerating cardiac bidomain simulations using graphics processing units. *IEEE transactions on bio-medical engineering*, 59(8):2281–90, 2012.
19. A. Nogami. Purkinje-related arrhythmias part I: Monomorphic ventricular tachycardias. *Pacing Clin. Electrophysiol.*, 34(5):624–650, 2011.
20. J. Rinzel and J.B. Keller. Traveling wave solutions of a nerve conduction equation. *Biophys. J.*, 13(12):1313–37, 1973.
21. Bernardo M. Rocha, Fernando O. Campos, Gernot Plank, Rodrigo W. Dos Santos, Manfred Liebmann, and Gundolf Haase. Simulations of the electrical activity in the heart with graphic processing units. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6067 LNCS, pages 439–448, 2010.
22. F. Sahli Costabal, D. E. Hurtado, and E. Kuhl. Generating Purkinje networks in the human heart. *J. Biomech.*, dec 2015.
23. R. Sebastian, V. Zimmerman, D. Romero, and A.F. Frangi. Construction of a computational anatomical model of the peripheral cardiac conduction system. *IEEE Trans. Biomed. Eng.*, 58(12):3479–82, 2011.
24. P. Stewart, O.V. Aslanidi, D. Noble, P.J. Noble, M.R. Boyett, and H. Zhang. Mathematical models of the electrical action potential of Purkinje fibre cells. *Philos. Trans. A. Math. Phys. Eng. Sci.*, 367(1896):2225–55, 2009.
25. K H W J ten Tusscher and a V Panfilov. Alternans and spiral breakup in a human ventricular tissue model. *Am. J. Physiol. Heart Circ. Physiol.*, 291(3):H1088–100, sep 2006.
26. K.H.W.J. Ten Tusscher and A.V. Panfilov. Modelling of the ventricular conduction system. *Prog. Biophys. Mol. Biol.*, 96(1):152–170, 2008.
27. N. A. Trayanova and P. M. Boyle. Advances in modeling ventricular arrhythmias: From mechanisms to the clinic. *Wiley Interdiscip. Rev. Syst. Biol. Med.*, 6(2):209–224, 2014.
28. A. Tveito and G.T. Lines. A condition for setting off ectopic waves in computational models of excitable cells. *Math. Biosci.*, 213(2):141–50, 2008.
29. C. Vergara, M. Lange, S. Palamara, T. Lassila, A.F. Frangi, and A Quarteroni. A coupled 3D-1D numerical monodomain solver for cardiac electrical activation in the myocardium with detailed Purkinje network. *J Comput Phys*, 308:218–238, 2016.
30. C. Vergara, M. Lange, S. Palamara, T. Lassila, A.F. Frangi, and A. Quarteroni. A coupled 3D–1D numerical monodomain solver for cardiac electrical activation in the myocardium with detailed Purkinje network. *J. Comput. Phys.*, 308:218–238, mar 2016.

31. Edward J. Vigmond, Patrick M. Boyle, L. Joshua Leon, and Gernot Plank. Near-real-time simulations of biolelectric activity in small mammalian hearts using graphical processing units. In *Proceedings of the 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society: Engineering the Future of Biomedicine, EMBC 2009*, pages 3290–3293, 2009.
32. Edward J Vigmond and Bruno D Stuyvers. Modeling our understanding of the His-Purkinje system. *Prog. Biophys. Mol. Biol.*, 120(1-3):179–88, jan 2016.
33. E.J. Vigmond and C. Clements. Construction of a computer model to investigate sawtooth effects in the Purkinje system. *IEEE Trans. Biomed. Eng.*, 54(3):389–99, 2007.