
Proceedings Paper:
Lange, M., Palamara, S., Lassila, T. et al. (3 more authors) (2015) Efficient Numerical Schemes for Computing Cardiac Electrical Activation over Realistic Purkinje Networks: Method and Verification. In: Functional Imaging and Modeling of the Heart. 8th International Conference, FIMH 2015, June 25-27, 2015, Maastricht, The Netherlands. Lecture Notes in Computer Science, 9126 . Springer International Publishing . ISBN 978-3-319-20308-9

https://doi.org/10.1007/978-3-319-20309-6_49

Efficient Numerical Schemes for Computing Cardiac Electrical Activation over Realistic Purkinje Networks: Method and Verification

Matthias Lange[1], Simone Palamara[2], Toni Lassila[1],
Christian Vergara[2], Alfio Quarteroni[3], and Alejandro F. Frangi[1]

[1] CISTIB, Department of Electronic and Electrical Engineering,
The University of Sheffield, United Kingdom
{m.lange,t.lassila,a.frangi}@sheffield.ac.uk
[2] MOX, Dipartimento di Matematica, Politecnico di Milano, Italy
{simone.palamara,christian.vergara}@polimi.it
[3] Chair of Modelling and Scientific Computing,
École Polytechnique Fédérale de Lausanne, Switzerland
alfio.quarteroni@epfl.ch

Abstract. We present a numerical solver for the fast conduction system in the heart using both a CPU and a hybrid CPU/GPU implementation. To verify both implementations, we construct analytical solutions and show that the L^2-error is similar in both implementations and decreases linearly with the spatial step size. Finally, we test the performance of the implementations with networks of varying complexity, where the hybrid implementation is, on average, 5.8 times faster.

1 Introduction

The cardiac Purkinje fibre network is an important contributor to the coordinated contraction of the heart as it can provide a fast conduction system reaching out to large areas of the sub-endocardium. The Purkinje fibres form an extensively branching and rejoining network, which is important for the reliability and fault-tolerance of the propagation of the action potential [1][2]. The ability to simulate propagation in physiological Purkinje networks is essential for studies of the healthy heart to obtain realistic activation patterns [3, 4].It is equally important in the simulation of pathological hearts, where disturbances in the conduction system can alter the activation pattern greatly, e.g. bundle branch blocks and long duration ventricular tachycardia [5].

Typically the simulation of the action potential propagation in a Purkinje network is based on the bidomain equations [6], or on the cable equation with a reaction term [7]. The approach of Vigmond *et al.* [7] treats the Purkinje conduction system as a branching tree of conducting segments without loops. Our approach also allows current loops in the Purkinje network, which are observed in realistic Purkinje networks [2].

We present first briefly the approach of Vigmond *et al.*, and then explain its implementation on the CPU and on a CPU/GPU hybrid platform. Then, we

present a simple model with exact known solution and use it for verification of both solvers. Finally, we compare the performance of both implementations.

2 Methods

2.1 Mathematical Model and Solution Method

The electrophysiology of cardiac tissue can be described either by the bidomain or the monodomain model. The former assumes an extracellular and intracellular space with different anisotropic conductivity tensors; if these tensors are linearly dependent the model reduces to the monodomain model [7].

The monodomain equation is considered in one dimension, because the Purkinje network can be approximated by a network of 1-D line segments. Here we assume that the extracellular space is not affected by the Purkinje network, and ignore it in the following. The monodomain equation reads

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

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

To derive a coupling condition between two or more line segments, needed to complete (1), we follow the idea of Vigmond *et al.* [7]. The equations on each line segment are coupled together by a boundary condition resulting from the enforcement of continuity of the potential and the conservation of charges (Kirchhoff's law). To satisfy the boundary conditions, the transmembrane potential, V_M, and the current, I, are needed. Since $I = \sigma \partial_x V_m$, the spatial derivatives of the potential need to be computed. The system is discretized using a cubic Hermite finite element method (FEM), which allows the current I to be recovered as a continuous quantity.

In view of the numerical discretization with the Finite Element Method, each node of the mesh is assumed to be located in the gap junction between two cells, where the unknowns are the intracellular potential ϕ_i and the current I_g through the gap junction. Two ghost nodes are created on both sides of the gap junction, where the transmembrane potential $V_\pm$, and ionic channel current I_{ion} are defined. The advantage of the ghost nodes is that with the gap junction modelled as a resistor R, the current I_g can be obtained from Ohm's law

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

where ϕ_e is the extracellular potential, taken constant in this work.

To correct for the introduced gap junction resistance, we use the equivalent conductivity $\sigma^* = (\sigma_i l)/(l + \sigma R \pi \rho^2)$, where l is the length of the Purkinje cell

and ρ the radius. This means, that σ_i is the conductivity in the cell only, while σ^* is the conductivity of the cell and the gap junction. In this notation (1) becomes

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

where $\phi_i^{\pm}$ is the intracellular potential in the ghost nodes. Furthermore, we apply an operator splitting technique to (3):

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

where $L_1 = I_{\text{ion}}$ is part of the differential operator that represents the nonlinear term of (3), whereas $L_2 = \partial_x(\sigma^*\partial_x)$ represent the diffusion term of (3). A fractional-step method with a discretization of the temporal derivatives by a first-order approximation is introduced, where the superscript n refers to the numerical solution computed at time t^n:

$$\frac{V^{n+1/2} - V^n}{\Delta t} = -L_1(V^n), \quad \frac{V^{n+1} - V^{n+1/2}}{\Delta t} = -L_2(V^{n+1}). \tag{5}$$

Algorithm 1 to Solve the Cable Equation with a Splitting Scheme

Step 1. Recover the transmembrane potential $V_{\pm}^n$ with (2) from I_g^n, ϕ_i^n, ϕ_e^n.

Step 2. Solve the first equation of the (5), which is the update of the ionic current in the ghost nodes

$$V_{\pm}^{n+1/2} = V_{\pm}^n - \frac{I_{\text{ion}}(V_{\pm}^n, \xi)}{C_m}\Delta t \ . \tag{6}$$

Step 3. Compute ϕ_i and I_g with the new values of $V_{\pm}^{n+1/2}$ in the real node:

$$\phi_i^{n+1/2} = \frac{V_+^{n+1/2} + V_-^{n+1/2}}{2} + \phi_e^n \ , \ I_g^{n+1/2} = \frac{V_+^{n+1/2} - V_-^{n+1/2}}{R} \ . \tag{7}$$

Step 4. Use the FEM for the second stage of the operator splitting. By noticing $\phi_i = \frac{\phi_i^+ + \phi_i^-}{2}$ and using the linearity of L_2, we find:

$$\beta C_m \partial_t(\phi_i - \phi_e) = \partial_x(\sigma^*\partial_x\phi_i) \ . \tag{8}$$

Introducing a discretization in time results in:

$$\beta C_m \frac{(\phi_i^{n+1} - \phi_e^{n+1}) - (\phi_i^{n+1/2} - \phi_e^n)}{\Delta t} = \partial_x\sigma^*\partial_x\phi_i^{n+1} \ , \tag{9}$$

which is solved with the FEM with 1-D cubic Hermite shape functions.

Now the cable equation can be solved in four steps (Algorithm 1). To handle branching and joining of segments in the network, the node where the three segments join is triplicated. The triplicated point is used to enforce the boundary conditions, and thus couple together the solutions obtained for the different

line segments. In the case that segment 1 bifurcates into segments 2 and 3, we enforce the continuity of the potential $\phi_1 = \phi_2 = \phi_3$ and the conservation of current $I_1 = I_2 + I_3$. In contrast to Vigmond *et al.*, our implementation covers the case where segments 1 and 2 join to from segment 3, in which case the coupling condition of the currents is $I_1 = I_3 - I_2$. These boundary conditions are introduced in the FEM system matrix associated to (9) and the right hand side.

2.2 Hardware Implementation

We now outline the CPU and the CPU/GPU hybrid implementations. The solver for the cable equation used the FEM in Step 4, and was implemented using the `LifeV` library (http://www.lifev.org) . Parallelism was achieved using OpenMPI. We parallelized only Steps 1-3 of the algorithm and solve the linear system in Step 4 serially. The reason for this is that calculating the ionic model can be done without knowing the mesh geometry and is computationally intensive. On the other hand, it is less trivial to parallelize the solving of the linear system. The resulting computational workflow is shown in Fig. 1.

The Steps 1, 3, and 4 were always implemented on the CPU, only Step 2 is run on the GPU. In the hybrid implementation, between Steps 1 and 2, an additional copy of the transmembrane potential $V_\pm$ from the CPU to the GPU is made. To minimize the time spent copying the data, CUDA streams are used, which allow asynchronous tasks to be queued to the CPU. All computations were performed with Dell a Precision-WorkStation-T7500 featuring two Intel(R) Xeon(R) CPUs E5620 at 2.40GHz and NVIDIA Quadro 4000 GPU with 256 CUDA Cores.

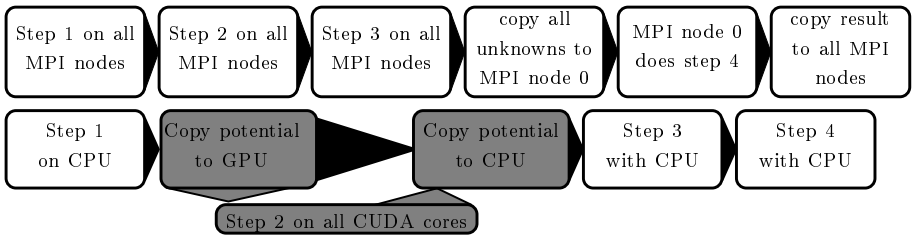

Fig. 1. Workflow for the CPU (above), and CPU/GPU hybrid (below) implementation. The CPU implementation needs to copy the potential in the gap junctions and the current, while the hybrid implementations needs to copy the potential of the ghost nodes. White boxes represent CPU tasks, and grey GPU tasks.

3 Numerical Experiments

To verify the correct and efficient implementation of the solvers, two numerical experiments are performed. The first experiment uses an analytical solution to

estimate the absolute error and then to carry out a convergence test. The second experiment compares the performs of the CPU and CPU/GPU hybrid algorithm.

3.1 Numerical Error and Convergence

We first introduce a simplified model and develop two test problems with analytical solutions. The non-physiological model is [8]

$$\partial_t V = pV, \tag{10}$$

where V is the transmembrane potential and p is a model parameter. Depending on the sign of p the cells are stable ($p < 0$) and return exponentially to 0, or are unstable ($p > 0$) and the transmembrane potential increases exponentially.

Next, we introduce two different test cases and derive their analytical solutions. For the first case the domain D_1 considered is an infinite line, which is composed of three subintervals $D_{1,1} = (-\infty, -a)$, $D_{1,2} = [-a, a]$, and $D_{1,3} = (a, \infty)$. In $D_{1,2}$ unstable cells are assumed, while in the surrounding regions $D_{1,1}$, $D_{1,3}$ the cells are stable, with results in a spatial depend parameter of the simplified model

$$p(x) = \begin{cases} p_2 & \text{for } x \in D_{1,2} \\ -p_1 & \text{otherwise} \end{cases}, \tag{11}$$

where $p_1, p_2 > 0$. Inserting the cell model in (1), we need to solve

$$\begin{array}{c} C_m \partial_t V = \delta \partial_x^2 V - p(x) V \\ \begin{array}{ccc} V_1(-a) = V_2(-a)\ , & V_2(a) & = V_3(a) \\ V_1'(x)|_{x=-a} = V_2'(x)|_{x=-a}\ , & V_2'(x)|_{x=a} & = V_3'(x)|_{x=a} \\ V_1(-\infty) = 0\ , & V_3(\infty) & = 0, \end{array} \end{array} \tag{12}$$

with $\delta = \sigma^*/\beta$. The solution presented by Artebrant *et al.* [8] is

$$V = \begin{cases} c_1 e^{\sqrt{p_1/\delta} x} & x < -a \\ \cos(\sqrt{p_2/\delta} x) & \|x\| \le a \\ c_1 e^{-\sqrt{p_1/\delta} x} & x > a \end{cases} \quad \text{with} \ , \begin{array}{l} p_1 = p_2 \tan^2(\sqrt{p_2/\delta} a), \\ c_1 = \cos(-\sqrt{p_2/\delta} a) e^{\sqrt{p_1/\delta} a} \end{array} \ , \tag{13}$$

where a and p_2 are the model parameters.

In the second test case, the domain D_2 is a double-bifurcation with an analytical solution. The domain consist of two rays, $D_{2,1} = (-\infty, -a)$ and $D_{2,2} = (-\infty, -a)$ joining to form a line segment $D_{2,3} = [-a, a]$ in the middle, which then splits again into two rays $D_{2,4} = (a, \infty), D_{2,5} = (a, \infty)$, resulting in a domain of five subintervals in total. The line segment $D_{2,3}$ consists of active cells, while all the rays consist of passive cells. The problem is symmetric with respect to zero, so we will look at the negative domain only. Furthermore, the rays $D_{2,1}$ and $D_{2,2}$ are identical, thus it suffices to solve the following problem for only one of them:

$$\begin{array}{c} \delta V_1'' - p_1 V_1 = 0, \qquad \forall\ x \in D_{2,1} \\ \delta V_3'' + p_2 V_3 = 0, \qquad \forall\ x \in D_{2,3} \\ V_1(-a) = V_3(-a)\ , \quad 2\ V_1'(x)|_{-a} = V_3(x)'|_{-a}\ , V_1(-\infty) = 0\ , \end{array} \tag{14}$$

where the two in the derivatives is a result of Kirchhoff's current law. The solution is very similar to the problem on one infinite line, with the ansatz functions $V_1 = c_1 e^{k_1 x}$, $V_3 = c_3 \cos(k_2 x)$ the constant c_1 is still given by (13). A relation between p_1 and p_2 follows from

$$\begin{aligned} 2V_1'(-a) &= V_3'(-a) \\ \overset{(13)}{\Rightarrow} 2k_1(c_3\cos(-k_2 a)e^{k_1 a})e^{-k_1 a} &= -k_2 c_3 \sin(-k_2 a) \\ \Rightarrow p_1 &= \tfrac{p_2}{4}\tan^2(\sqrt{p_2/\delta}a). \end{aligned} \tag{15}$$

Again, we need to fix V_3 at one point to get a unique solution.

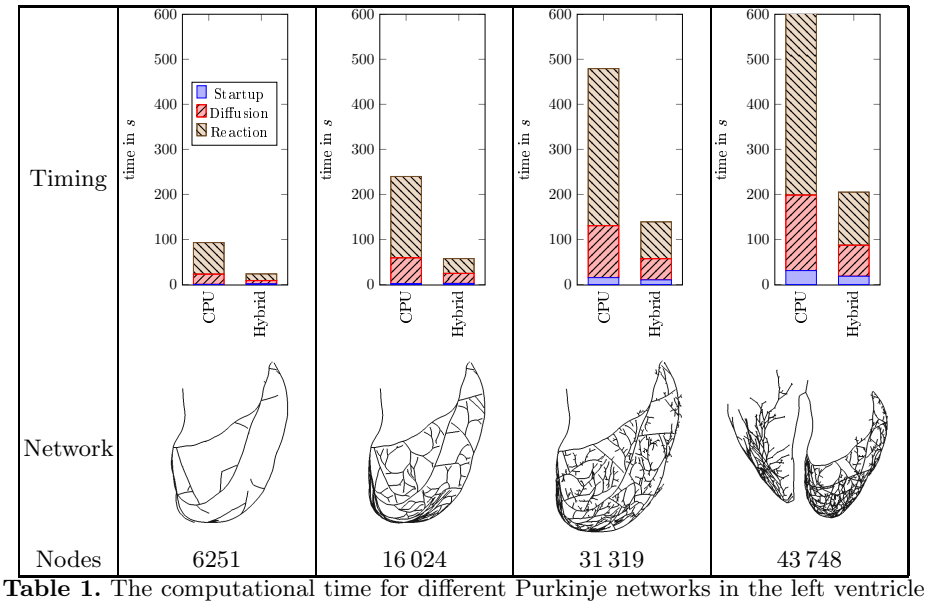

Table 1. The computational time for different Purkinje networks in the left ventricle (LV) and right ventricle (RV).

Comparison of the Absolute Error: For numerical simulations, we used the parameters $p_2 = 1\ \mu F/ms$, $a = 1cm$, $C_m = 1\mu F$, and $c_2 = 1$. The cell length has been chosen to $l = 62.5\ \mu m$, and a radius of $\rho = 16.0\ \mu m$, which is within the physiological limits [9]. Furthermore, we make the arbitrary choice $\delta = 1\ [kS/cm^2]$, $R = 0.1\ k\Omega$ and recall, that $\delta = \sigma^*/\beta$, from which we find the conductivity $\sigma_i = 1967\ [kS/cm]$. The spatial discretisation step h is then chosen to be a integer multiple of l, i.e. $h = nl$, $n \in \mathbb{N}$, which means that each finite element contains $n-1$ virtual gap junctions and only the nth gap junction is explicitly included.

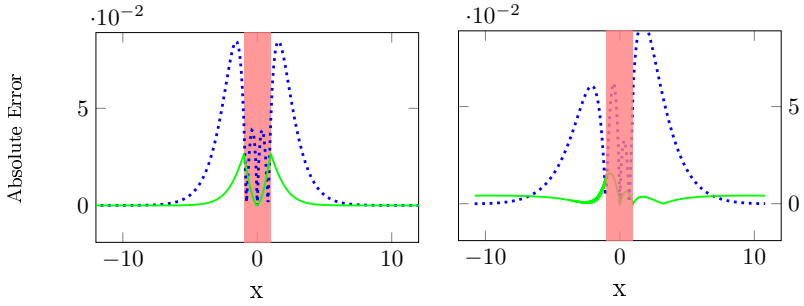

Fig. 2. The absolute error between the analytic solution of the potential and the numerical solution. For the test case on an infinite line (left), and for the simple branching network (right), where the dotted line has a step size of 0.1 cm, while the solid is 0.00625 cm and the error is multiplied by 10. The active cells are in the mark region.

Convergence Test: For the error convergence test, we ran the simulation with the same parameters as before for $n = \{1, 2, 3, 4, 5\}$ in the spatial discretisation and calculated the L^2-Error for each step size (Fig. 3). The CPU and CPU/GPU hybrid implementation give the same linear convergence of the error. We conjecture that the fourth order of convergence, which is expected for Hermit bases, is not reached because of the step 1 and 3.

3.2 Performance Comparison

To compare the efficiency of the two implementations, four Purkinje fibre networks were generated with the method presented in [10]. The last two networks are realistic networks for the left ventricle and for both ventricles, respectively. The simulation was performed with a spatial resolution of 0.1 mm and a temporal step size of 0.02 ms. The duration of 45 ms was chosen because all networks were fully depolarised by that time. The membrane model of Di Francesco-Noble was used [11]. The CPU code was run with eight parallel processes, while the hybrid code was run with one CPU. Table 1 shows both the networks and the total computational time spent obtaining the respective solutions. Furthermore, the same figure shows the time spent solving the diffusion problem and the reaction problem separately. In the pure CPU implementation, the majority of the time is used to solve the ionic models. This is due to the fact that a detailed ionic model with 15 state variables was used, while the linear system for the diffusion step is comparably simple to solve, as the moving activation front is limited to the vicinity of a few node points. For the hybrid implementation the situation changes, and the time for solving the reaction and diffusion steps are the roughly the same, because the GPU offers a larger number of parallel cores. As a result the solution of the reaction step is ca. 4.7 times faster with the GPU. We also notice a decrease in the time spent solving the diffusion step. This can be a result of several factors, including that there is no memory copy between the CPUs, and the CPU can be used in turbo mode, because of CPU core switches.

4 Conclusion

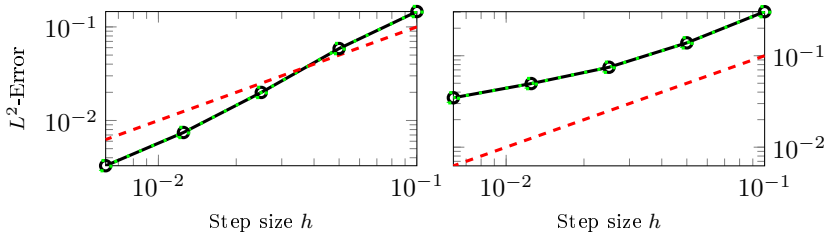

Fig. 3. Linear convergence in h (dashed line) and the convergence rates of the potential computed with the CPU (dotted line), and from the CPU/GPU hybrid (solid line). Results are for the single line case (left) and for the simple branching network (right).

We presented an extension of the work of Vigmond *et al.* to solve more realistic Purkinje networks, and implemented it both on a CPU and in a hybrid CPU/GPU architecture. To evaluate the accuracy of both we performed a convergence test of the L^2-Error, and showed that the solver converges linearly with the step size. The branching points introduce a small additional error in the numerical solution. Both implementations had equivalent numerical accuracy.

The performance test indicated that the hybrid implementation using 256 CUDA cores and 1 CPU was in average 5.8 times faster than the CPU implementation run with 8 CPUs. This motivates our future work on developing an implementation, which performs all the remaining steps of the algorithm on the GPU to realize even greater performance gains.

Acknowledgements

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 has been partially supported by the Italian MIUR PRIN09 project no. 2009Y4RC3B_001.

References

[1] Cooper, L.L., Odening, K.E., Hwang, M.-S., Chaves, L., Schofield, L., Taylor, C., Gemignani, A.S., Mitchell, G.F., Forder, J.R., Choi, B.-R., Koren, G.: Electromechanical and structural alterations in the aging rabbit heart and aorta. Am. J. Physiol. Heart Circ. Physiol. 302, H1625–H1635 (2012)

[2] Ansari, A., Ho, S. Y., Anderson,R. H.: Distribution of the Purkinje fibres in the sheep heart. Anat. Rec., 254, 92-97 (1999).

[3] Vergara, C., Palamara, S., Catanzariti, D., Nobile, F., Faggiano, E., Pangrazzi, C. Maurizio Centonze, Massimiliano Maines, Alfio Quarteroni, Giuseppe Vergara : Patient-specific generation of the Purkinje network driven by clinical measurements of a normal propagation. Med Biol Eng Comput, 52(10), 813–826, (2014)

[4] Palamara S., Vergara C., Catanzariti D., Faggiano E., Centonze M., Pangrazzi C., Maines M., Quarteroni A.: Computational generation of the Purkinje network driven by clinical measurements: The case of pathological propagations. Int. J. Num. Meth. Biomed. Eng., 30(12), 1558–1577, (2014)

[5] Bogun, F., Good, E., Reich, S., Elmouchi, D., Igic, P., Tschopp, D., Dey S., Wimmer, A., Jongnarangsin, K., Oral, H., Chugh, A., Pelosi, F., Morady, F.: Role of Purkinje Fibers in Post-Infarction Ventricular Tachycardia. J Am Coll Cardiol. 48(12), 2500–2507 (2006)

[6] Bordas, R.M., Gillow, K., Gavaghan, D., Rodriguez, B., Kay, D.: A Bidomain model of the ventricular specialized conduction system of the heart. SIAM J. Appl. Math. 72, 1618–1643 (2012)

[7] Vigmond, E.J., Clements, C: Construction of a computer model to investigate sawtooth effects in the Purkinje system. IEEE Trans. Biomed. Eng. 54, 389–99 (2007)

[8] Artebrant, R., Tveito, A., Lines, G.T.: A method for analyzing the stability of the resting state for a model of pacemaker cells surrounded by stable cells. Math. Biosci. Eng. 7, 505–526 (2010)

[9] Stankovicová T, Bito V, Heinzel F, Mubagwa K, Sipido KR.: Isolation and Morphology of Single Purkinje Cells from the Porcine Heart. Gen Physiol Biophys. 22(3), 329–340 (2003)

[10] Sebastian, R., Zimmerman, V., Romero, D., Frangi, A.F.: Construction of a computational anatomical model of the peripheral cardiac conduction system. IEEE Trans. Biomed. Eng. 58(12), 3479–3482 (2011)

[11] DiFrancesco, D. and Noble, D.: 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)