An eikonal-diffusion solver and its application to the interpolation and the simulation of reentrant cardiac activations

Vincent Jacquemet[a,b]

[a]*Institut de Génie Biomédical & Département de Physiologie, Université de Montréal*
[b]*Hôpital du Sacré-Coeur de Montréal, Centre de Recherche, 5400 boul. Gouin Ouest, Montréal (QC) H4J 1C5, Canada*
phone: +1 514-338-2222 ext. 2522
`vincent.jacquemet@umontreal.ca`

Published in: Comput. Methods Programs Biomed. 2012, vol. 108, no. 2, pp. 548-558

Abstract

Electrical propagation of the cardiac impulse in the myocardium can be described by the eikonal-diffusion equation. This equation governs the field of activation times in a domain where conduction properties are specified. This approach has been applied to knowledge-based interpolation of sparse measurements of activation times and to the creation of initial conditions for detailed ionic models of cardiac propagation. This paper presents the mathematical basis, matrix formulation, and compact Matlab implementation of an iterative finite-element solver (triangular meshes) for the eikonal-diffusion equation extended to reentrant activations, which automatically identifies the period of reentry and computes the resulting isochrones. An iterative algorithm is designed to perform Laplacian interpolation of reentrant activation maps to be used as initial estimate for the eikonal-diffusion solver. The performance of the algorithm is analyzed in test-case geometries (ventricular slice and simplified atrial surface model).

Keywords: cardiac propagation, reentry, eikonal-diffusion equation, interpolation, numerical methods

1. Introduction

Atrial arrhythmias are rhythm disorders frequently encountered in clinical practice. Current therapies include pharmacological control of the ventricular rate, electrical cardioversion (defibrillation) and catheter ablation (creation of lesions in the atrial tissue using radio frequency or cryo catheter electrodes). Catheter ablation involves exploration of atrial endocardium with intracardiac electrodes recording electrical signal (electroanatomical mapping). Local activation time can usually be extracted from these intracardiac electric signals. In combination with cardiac imaging data, this procedure can provide a description of the dynamics of the arrhythmia through activation maps. Spatial resolution is, however, often limited.

To investigate the basic mechanisms of atrial arrhythmias and guide the development of diagnostic and therapeutic tools, computer models of atrial electrophysiology have been developed [1–6]. In these models, propagation of the electrical impulse in the myocardium is governed by a reaction-diffusion equation [7]. To improve the clinical relevance of model results, patient-specific information needs to be incorporated. This information can be local (cell electrophysiology, cell-to-cell coupling) or global (dynamics of the arrhythmia, pathways of reentry). Local, microscale data are natural parameters in the bottom-up approach typically used in cardiac modeling. Global, macroscale data such as activation maps are often easier to obtain but more difficult to integrate in the model.

Lines *et al.* proposed to add a non-local forcing term to the reaction-diffusion equation to synchronize a reentrant activity with experimental or synthetic signals recorded at sparse locations during atrial flutter [8]. More recently, we developed a method for creating an initial condition for the reaction-diffusion system from a reentrant pathway [9]. This method was based on the eikonal-diffusion equation [10–12], a partial differential equation for the activation time. Activation maps computed using this approach showed good correspondence with those simulated using the reaction-diffusion system, at a much lower computational cost. In addition, the initial condition created from an activation map enabled the simulation of reentries along prescribed anatomical or functional reentrant pathways in the reaction-diffusion model [9].

In this paper, an efficient iterative solver for the eikonal-diffusion equation applied to reentrant activity is described. Numerical methods from [9] are reformulated to facilitate and optimize Matlab implementation. An alternative, more robust algorithm for the interpolation of activation times (used as initial estimate for the iterative solver) is proposed. Compact Matlab code is provided and explained. A theoretical analysis of the algorithm is presented that enables automatic computation of the period of the reentry, thus reducing the number of required input parameters. Performance and accuracy is assessed using test case problems.

2. Background

2.1. Problem statement

Activation time is the time $t_a(\mathbf{x})$ at which an electrical impulse (cardiac wave front) passes through the point $\mathbf{x}$. The field $t_a(\mathbf{x})$ forms an activation map. Stable reentry consists in single or multiple self-sustained activation waves propagating periodically in the cardiac tissue. To emphasize the periodic nature of a reentrant activity and exhibit its topological features, the scaled activation time τ is defined as

$$\tau(\mathbf{x}) = 2\pi\, t_a(\mathbf{x})/T \mod 2\pi, \tag{1}$$

where T is the period of the reentry. For the moment, T is assumed to be known. Later, methods will be presented to derive its value from conduction properties (Subsects. 2.3 and 4.5).

Two problems will be considered in this paper:

1. *Interpolation*: From a set of known scaled activation times $\tau(\mathbf{x}_i)$, $\mathbf{x}_i \in \Gamma$, interpolate an activation map $\tau(\mathbf{x})$ while taking into account its periodic nature. The set Γ can be a discrete set of points (interpolation from a finite number of measurements, for example catheter electrodes) or closed curves describing observed pathways of reentry.

2. *Simulation*: Reconstruct activation maps using *a priori* knowledge about wave front propagation (local curvature-dependent conduction velocity). Adjust activation maps obtained from problem 1 to satisfy hypothesized conduction properties of the tissue substrate.

Both problems will be solved using a partial differential equation based on the eikonal-diffusion equation for the field τ.

2.2. The eikonal-diffusion equation for a reentry

Derived from the monodomain propagation equations [7] using singular perturbation techniques [12], the eikonal-diffusion equation in the domain Ω (with boundary $\partial\Omega$) governs the shape of activation wave fronts [9–12]:

$$\|\mathbf{c}\nabla\tau\| = 1 + \nabla\cdot(\mathbf{D}\nabla\tau) \qquad \mathbf{x}\in\Omega, \tag{2}$$

$$\mathbf{n}\cdot\mathbf{D}\nabla\tau = 0 \qquad \mathbf{x}\in\partial\Omega. \tag{3}$$

where the tensors $\mathbf{c}$ (scaled propagation velocity in cm/rad, which means conduction velocity in cm/s $\times\, T/2\pi$) and $\mathbf{D}$ (scaled diffusion tensor in cm^2) are symmetric positive definite, $\|\cdot\|$ is the euclidean norm and $\mathbf{n}$ is a unit vector normal to the boundary $\partial\Omega$. In the isotropic case with $D \to 0$, it reduces to the eikonal equation $c\,\|\nabla\tau\| = 1$ stating that the propagation velocity of the wave fronts is constant [13]. Diffusion of activation times introduces wave front curvature-dependent propagation velocity [10]. In the purely diffusive limit $\mathbf{D} = \lambda\hat{\mathbf{D}}$ with $\lambda \to +\infty$, the equation becomes the diffusion equation

$$\nabla\cdot(\hat{\mathbf{D}}\nabla\tau) = 0 \qquad \mathbf{x}\in\Omega\setminus\Gamma, \tag{4}$$

$$\mathbf{n}\cdot\hat{\mathbf{D}}\nabla\tau = 0 \qquad \mathbf{x}\in\partial\Omega\setminus\Gamma, \tag{5}$$

$$\tau(\mathbf{x}) = \tau_0(\mathbf{x}) \qquad \mathbf{x}\in\Gamma. \tag{6}$$

The Dirichlet boundary condition on Γ (6) was added to formulate a Laplacian interpolation problem [14]. Interpolation of activation times is therefore a limit case of the eikonal-diffusion problem.

Note, however, that τ may contain 2π jumps anywhere, which makes it more difficult in this formulation to numerically compute the gradient [15]. To handle this phase unwrapping problem, a phase function transform $\phi = \exp(i\tau)$ is applied. The transformed eikonal-diffusion equation reads [9]:

$$\|\mathbf{c}\nabla\phi\| = 1 + \mathrm{Im}\,\nabla\cdot(\phi^*\mathbf{D}\nabla\phi) \qquad \mathbf{x}\in\Omega, \tag{7}$$

$$|\phi| = 1 \qquad \mathbf{x}\in\Omega, \tag{8}$$

$$\mathbf{n}\cdot\mathbf{D}\nabla\phi = 0 \qquad \mathbf{x}\in\partial\Omega, \tag{9}$$

where the symbol 'Im' denotes the imaginary part and the star $(*)$ means the conjugate (when applicable) transposed vector/tensor.

2.3. Parameter identification

The parameters **c** and **D** will be selected to reproduce activation patterns that would be observed in a monodomain model with conductivity tensor $\boldsymbol{\sigma}$ (mS/cm), membrane surface-to-volume ratio β (cm^{-1}), and membrane capacitance per unit membrane area C_m (μF/cm^2). The derivation of the eikonal-diffusion equation from the monodomain model [12] establishes the following relations [9, 10]:

$$\mathbf{c}^*\mathbf{c} = \frac{T^2 k_m}{4\pi^2\beta C_m}\,\boldsymbol{\sigma} \qquad \text{and} \qquad \mathbf{D} = \frac{T}{2\pi\beta C_m}\,\boldsymbol{\sigma}\ . \tag{10}$$

The membrane model-dependent parameter k_m is such that the conduction velocity (cm/ms) of a plane wave is $CV = \sqrt{k_m\sigma/\beta C_m}$. At this point, the period T of the reentry is assumed to be known. Numerical methods will be presented that automatically identify the appropriate value of T from the solution computed with an arbitrary value $\tilde{T}$ (Subsect. 4.5), for example $\tilde{T} = 1000$ ms. In some applications, only the target conduction velocity CV (for plane wave propagation) is known or is determined experimentally, for example on the basis of endocardial activation mapping [16, 17]. In such a case, the parameters **c** and **D** are set to:

$$\mathbf{c} = \frac{\tilde{T}}{2\pi}\,\mathbf{CV} \qquad \text{and} \qquad \mathbf{D} = \frac{\tilde{T}}{2\pi k_m}\,\mathbf{CV}^2\ , \tag{11}$$

where the membrane-dependent parameter k_m can be set to 2.0833 ms^{-1} [10] or to a value specifically adjusted to a given membrane model.

3. Design considerations

The goal is to implement in Matlab an efficient eikonal-diffusion solver for reentrant activation maps defined on (coarse) triangular meshes of 2D geometries and 3D surfaces. A previously-developed method [9] will be extended and reformulated to fully use sparse matrix operation capabilities in Matlab and facilitate its implementation. A faster and more robust iterative algorithm will be proposed for the diffusive case ($D \to \infty$). This alternative approach will serve both for the interpolation problem and as initial estimate for the eikonal-diffusion solver. Another goal is to automatically determine the period T of the reentry to reduce the number of adjustable parameters needed to generate activation maps. Computed activation maps will be compared to those simulated in the framework of the monodomain model [9].

4. Methods

4.1. An iterative eikonal-diffusion solver

To numerically solve Eq. (7), a linearized eikonal approach [18] will be used. This method is equivalent to the Newton root finding method applied at the level of the partial differential equation [9, 19]. Assuming that an estimate ϕ^0 of the solution to (7) satisfying the constraint (8) and the boundary condition (9) is available, an iterative scheme is obtained by applying a space-varying phase shift $\phi^{s+1} = \phi^s \exp(i\theta^s)$ at step s, such that ϕ^{s+1} is the solution to (7) up to first order in $\nabla\theta^s$. This process is then iterated to eventually converge toward the solution. Analytical calculations using a Taylor expansion of (7) lead to the following linear equation for the correction θ^s [9]:

$$\|\mathbf{c}\nabla\phi^s\| - \mathrm{Im}\ \nabla \cdot (\phi^{s*}\mathbf{D}\nabla\phi^s) - 1 = \|\mathbf{c}\nabla\phi^s\|^{-1}\ \mathrm{Im}\ (\phi^s\nabla\phi^{s*}\mathbf{c}^*\mathbf{c}\nabla\theta^s) + \nabla \cdot (\mathbf{D}\nabla\theta^s) \quad (12)$$

This is a steady-state convection-diffusion equation in θ^s with coefficients that depend on the known field ϕ^s. The correction θ^s must also verify the boundary condition $\mathbf{n} \cdot \mathbf{D}\nabla\theta^s = 0$ so that the same boundary condition will hold for ϕ^{s+1}.

The two-dimensional domain Ω is discretized using a triangular mesh composed of n_t triangles and n_v vertices. The set of triangles is denoted by $\mathcal{T}$ and the set of vertices by $\mathcal{V}$. Vertex $m \in \mathcal{V}$ is located at position $\mathbf{x}_m$. The triangle $(ijk) \in \mathcal{T}$ is denoted by T_{ijk} and its area by Ω_{ijk}. The area associated with node $m \in \mathcal{V}$ is $\Omega_m = \frac{1}{3}\sum_{(ijk)\ni m} \Omega_{ijk}$ where the sum runs over the neighboring triangles. The vectors $\boldsymbol{\phi}^s$ and $\boldsymbol{\theta}^s$, both of size n_v by 1, are formed by the values ϕ_i^s of the field ϕ^s (respectively the values θ_i^s of the field θ^s) at vertex $i \in V$. The fields ϕ and θ are reconstructed by interpolation using linear shape functions N_i ($N_i(\mathbf{x}_j) = \delta_{ij}$, the Kronecker symbol):

$$\phi^s = \sum_{i\in\mathcal{V}} \phi_i^s N_i \qquad \text{and} \qquad \theta^s = \sum_{i\in\mathcal{V}} \theta_i^s N_i \ . \tag{13}$$

The application of the finite element method to the linearized eikonal-diffusion equation (12) leads to a linear system

$$\mathbf{A}(\boldsymbol{\phi}^s)\, \boldsymbol{\theta}^s = \mathbf{f}(\boldsymbol{\phi}^s) \tag{14}$$

Explicit expressions for the matrix $\mathbf{A}$ and the right hand side $\mathbf{f}$ have been shown to be given by [9] (here both A_{mn} and f_m have been normalized by Ω_m):

$$\begin{aligned} A_{mn}(\phi) = & -\sum_{(ijk)\in\mathcal{T}} \frac{\Omega_{ijk}}{\Omega_m} \nabla_{ijk} N_m \cdot \mathbf{D}_{ijk} \nabla_{ijk} N_n \\ & + \sum_{(ijk)\in\mathcal{T}} \frac{\Omega_{ijk}}{3\Omega_m} \|\mathbf{c}_{ijk}\nabla_{ijk}\phi\|^{-1} \ \mathrm{Im}\ \frac{\phi_i+\phi_j+\phi_k+\phi_m}{4} \left(\mathbf{c}_{ijk}\nabla_{ijk}\phi\right)^* \cdot \left(\mathbf{c}_{ijk}\nabla_{ijk}N_n\right) \end{aligned} \tag{15}$$

$$f_m(\phi) = \sum_{\substack{(ijk)\in\mathcal{T}\\ m\in\{ijk\}}} \frac{\Omega_{ijk}}{3\Omega_m} \left(\|\mathbf{c}_{ijk}\nabla_{ijk}\phi\| - 1 + 3\ \nabla_{ijk} N_m \cdot \mathrm{Im} \frac{\phi_i^* + \phi_j^* + \phi_k^*}{3} \mathbf{D}_{ijk}\nabla_{ijk}\phi \right) \ . \tag{16}$$

The index ijk in $\mathbf{c}_{ijk}$, $\mathbf{D}_{ijk}$ and ∇_{ijk} indicates that these fields are evaluated at the center of gravity of the triangle (ijk). The outline of the eikonal-solver is described by the following algorithm:

1. Generate an initial estimate $\boldsymbol{\phi}^0$
2. Iterate for $s = 0, 1, 2, \ldots$ until $\|\boldsymbol{\theta}^s\| < tol$
    (a) Compute $\mathbf{A}(\boldsymbol{\phi}^s)$ [Eq. (15)] and $\mathbf{f}(\boldsymbol{\phi}^s)$ [Eq. (16)]

(b) Solve $\mathbf{A}(\boldsymbol{\phi}^s)\,\boldsymbol{\theta}^s = \mathbf{f}(\boldsymbol{\phi}^s)$

(c) Mean subtraction: $\boldsymbol{\theta}^s := \boldsymbol{\theta}^s - \mathrm{mean}(\boldsymbol{\theta}^s)$

(d) Under-relaxation: $\boldsymbol{\theta}^s := \boldsymbol{\theta}^s \cdot \min\big(1, \theta_{max}/\max(|\theta^s|)\big)$, where θ_{max} is a fixed parameter

(e) Set $\boldsymbol{\phi}^{s+1} := \boldsymbol{\phi}^s \exp(i\boldsymbol{\theta}^s)$

The tolerance *tol* was set to 10^{-10} and the under-relaxation threshold θ_{max} to 0.1 [9]. The reason for step 2(c) will be explained in Subsect. 4.5. The next subsection will cover the construction of the initial estimate $\boldsymbol{\phi}^0$. The following subsections will present the derivation of new expressions for the matrix $\mathbf{A}$ and the vector $\mathbf{f}$ in terms of sparse matrices. This will facilitate the implementation of a compact and efficient code in Matlab taking advantage of sparse matrix operations.

4.2. Generating the initial estimate

The initial estimate ϕ^0 is constructed by solving the eikonal-diffusion in the diffusion-dominant limit ($\mathbf{D} = \lambda\hat{\mathbf{D}}$ with $\lambda \to +\infty$). In order to make the initial activation map estimate follow a prescribed pathway, an additional boundary condition will be defined on set Γ (typically a curve describing the pathway): $\phi = \phi_0$ on Γ. Then, the algorithm of the previous subsection can be viewed (and can be implemented) as a continuation procedure [19] in which D is progressively reduced from $+\infty$ to a finite value.

In the diffusion-dominant limit, the system reduces to

$$\nabla \cdot (\phi^* \hat{\mathbf{D}} \nabla \phi) = 0 \tag{17}$$

with the constraint $|\phi| = 1$ and the boundary condition $\phi = \phi_0$ on Γ and $\mathbf{n} \cdot \hat{\mathbf{D}}\nabla\phi = 0$ on $\partial\Omega \setminus \Gamma$. If $\phi = \exp(i\tau)$ is substituted to satisfy $|\phi| = 1$, the equation becomes $\nabla \cdot (\hat{\mathbf{D}}\nabla\tau) = 0$, that is, it reduces to Laplacian interpolation [14] of an orientation field from values defined on a set Γ. Note that the 'Im' symbol has been dropped since the left hand side is purely imaginary when $|\phi| = 1$.

This nonlinear equation can be solved using a variant of fixed point iterations [19].

Starting from $\phi^0 = 1$ on $\Omega \setminus \Gamma$, ϕ^{s+1} is computed from

$$\nabla \cdot \phi^{s*} \hat{\mathbf{D}} \nabla \phi^{s+1} = 0 \tag{18}$$

with boundary condition $\phi^{s+1} = \phi_0$ on Γ and $\mathbf{n} \cdot \hat{\mathbf{D}} \nabla \phi^{s+1} = 0$ on the remaining part of the boundary $\partial\Omega \setminus \Gamma$. Note that the constraint $|\phi| = 1$ is released. As a result, the solution ϕ^{s+1} has to be normalized by dividing it by its module.

Similarly to the previous subsection, this equation can be discretized as a linear system $\mathbf{A}^\infty(\boldsymbol{\phi}^s)\,\boldsymbol{\phi}^{s+1} = \mathbf{f}^\infty(\boldsymbol{\phi}^s)$. The matrix $\mathbf{A}^\infty$ [compare with the third term of Eq. (16)] is given by

$$A^\infty_{mn}(\phi) = -\sum_{(ijk)\in\mathcal{T}} \frac{\Omega_{ijk}}{\Omega_m} \nabla_{ijk} N_m \cdot \frac{\phi_i^* + \phi_j^* + \phi_k^*}{3} \,\hat{\mathbf{D}}_{ijk} \nabla_{ijk} N_n \ . \tag{19}$$

The boundary condition is introduced by setting $\mathbf{f}^\infty$ to zero and then replacing the i-th row of the linear system, for each $i \in \Gamma$, by the equation $\phi_i^{s+1} = \phi_i^s = \phi_{0,i}$. The outline of the algorithm for generating the initial estimate is therefore:

1. Set $\boldsymbol{\phi}^0 = 1$
2. Iterate for $s = 0, 1, 2, \ldots$ until $\left\|\boldsymbol{\phi}^{s+1} - \boldsymbol{\phi}^s\right\| < tol$
   (a) Compute $\mathbf{A}^\infty(\boldsymbol{\phi}^s)$ and $\mathbf{f}^\infty(\boldsymbol{\phi}^s)$ [Eq. (19)], taking into account the boundary condition
   (b) Solve $\mathbf{A}^\infty(\boldsymbol{\phi}^s)\,\boldsymbol{\phi}^{s+1} = \mathbf{f}^\infty(\boldsymbol{\phi}^s)$
   (c) Normalize: $\boldsymbol{\phi}^{s+1} := \boldsymbol{\phi}^{s+1} / \left|\boldsymbol{\phi}^{s+1}\right|$

The first iteration consists in solving the linear diffusion equation $\nabla \cdot \hat{\mathbf{D}} \nabla \phi^1 = 0$, which is the approach proposed in [9] as initial estimate for the eikonal-diffusion solver. The tolerance *tol* was set to 10^{-10}.

4.3. Sparse matrix formulation

Domain geometry is described by the list of vertices $\mathcal{V}$, numbered from $m = 1$ to n_v, and the list of triangles $\mathcal{T}$, numbered from $ijk = 1$ to n_t. Indices μ and ν will run over the three spatial components x, y and z of a vector. Building-block matrices are defined as follows.

To convert a vertex-centered field into an element-centered field, values are averaged over the 3 vertices of the triangle. This transform defines the matrix $\mathbf{T}^{v\to t}$ of size n_t by n_v:

$$\left(\mathbf{T}^{v\to t}\right)_{ijk,m} = \frac{1}{3} \quad \text{if } m \in \{i,j,k\} \tag{20}$$

and 0 otherwise. Conversion from an element-centered field to a vertex-centered field is performed using the matrix $\mathbf{T}^{t\to v}$, of size n_v by n_t:

$$\left(\mathbf{T}^{t\to v}\right)_{m,ijk} = \frac{\Omega_{ijk}}{3\Omega_m} \quad \text{if } m \in \{i,j,k\} \tag{21}$$

and 0 otherwise. The matrix $\mathbf{Grad}_\mu$ (component μ of the gradient), of size n_t by n_v, is defined for each $\mu = x, y, z$ by

$$(\mathbf{Grad}_\mu)_{ijk,m} = \partial_\mu N_m \quad \text{if } m \in \{i,j,k\} \tag{22}$$

and 0 otherwise. The partial derivative along the coordinate μ, $\partial_\mu N_m$, can be easily computed using the dual basis of the triangle [20]. The matrix $\mathbf{Div}_\mu$, of size n_v by n_t, is defined for each $\mu = x, y, z$ by

$$(\mathbf{Div}_\mu)_{m,ijk} = -\frac{\Omega_{ijk}}{\Omega_m}\partial_\mu N_m \quad \text{if } m \in \{i,j,k\} \tag{23}$$

and 0 otherwise.

The multiplication of the gradient by a tensor is obtained by summing over the x, y and z components. The discretized version of the component μ of the operator $\mathbf{D}\nabla$ is written as

$$(\mathbf{DGrad}_\mu)_{ijk,m} = \sum_{\nu=x,y,z} D^{\mu\nu}_{ijk}\, \partial_\nu N_m \quad \text{if } m \in \{i,j,k\} \tag{24}$$

and 0 otherwise, where $D^{\mu\nu}_{ijk}$ is the $\mu\nu$ component of the 3-by-3 tensor $\mathbf{D}_{ijk}$. If the vector $\mathbf{D}^{\mu\nu}$, of size n_t by 1, is constructed such that its entry at row ijk is given by $D^{\mu\nu}_{ijk}$, the matrix $\mathbf{DGrad}_\mu$ can be expressed as

$$\mathbf{DGrad}_\mu = \sum_{\nu=x,y,z} \mathbf{diag}\left(\mathbf{D}^{\mu\nu}\right)\mathbf{Grad}_\nu\ , \tag{25}$$

where $\mathbf{diag}(\mathbf{x})$ is a diagonal matrix containing the components of $\mathbf{x}$ along its diagonal. $\mathbf{CGrad}_\mu$ is defined similarly as a function of $\mathbf{c}_{ijk}$. None of these matrices depend on ϕ; they can be precomputed.

With all these definitions, the matrix $\mathbf{A}(\boldsymbol{\phi})$ can be written as

$$\begin{aligned}\mathbf{A} = \mathbf{Diff} &+ \frac{3}{4}\,\mathbf{T}^{t\to v}\cdot\mathbf{Norm}^{-1}\cdot\mathrm{Im}\left(\mathbf{diag}\left(\mathbf{T}^{v\to t}\boldsymbol{\phi}\right)\cdot\mathbf{B}\right)\\ &+ \frac{1}{4}\,\mathrm{Im}\left(\mathbf{diag}\left(\boldsymbol{\phi}\right)\cdot\mathbf{T}^{t\to v}\cdot\mathbf{Norm}^{-1}\cdot\mathbf{B}\right)\ ,\end{aligned}\tag{26}$$

where the n_v-by-n_v matrices **Diff** and **B** are defined as

$$\mathbf{Diff} = \sum_{\mu=x,y,z}\mathbf{Div}_\mu\cdot\mathbf{DGrad}_\mu \tag{27}$$

$$\mathbf{B} = \sum_{\mu=x,y,z}\mathbf{diag}\left(\mathbf{CGrad}_\mu\cdot\boldsymbol{\phi}\right)^*\cdot\mathbf{CGrad}_\mu\ , \tag{28}$$

and the norm of the gradient is represented as a n_t-by-n_t diagonal matrix

$$\mathbf{Norm} = \left(\sum_{\mu=x,y,z}\mathbf{diag}\left(\mathbf{CGrad}_\mu\cdot\boldsymbol{\phi}\right)^2\right)^{1/2}\ , \tag{29}$$

where the power 1/2 is computed component-wise along the diagonal. The right hand side $\mathbf{f}$ (n_v by 1) can be written as

$$\mathbf{f} = \mathbf{T}^{t\to v}\cdot\left(\mathbf{Norm}\cdot\mathbf{e}-\mathbf{e}\right)-\mathrm{Im}\sum_{\mu=x,y,z}\mathbf{Div}_\mu\cdot\mathbf{diag}\left(\mathbf{T}^{v\to t}\boldsymbol{\phi}^*\right)\cdot\mathbf{DGrad}_\mu\cdot\boldsymbol{\phi} \tag{30}$$

where $\mathbf{e}$ is the n_t-by-1 vector containing 1 in each entry. Similarly, a comparison between (16) and (19) shows that

$$\mathbf{A}^\infty = \sum_{\mu=x,y,z}\mathbf{Div}_\mu\cdot\mathbf{diag}\left(\mathbf{T}^{v\to t}\boldsymbol{\phi}^*\right)\cdot\mathbf{DGrad}_\mu\ . \tag{31}$$

4.4. Matlab implementation

The algorithm sketched at the end of Subsect. 4.1 was implemented in Matlab. The list of vertices is represented by a n_v-by-3 matrix `VER`, one row for the xyz components of each vertex, and the list of triangles by a n_t-by-3 matrix `TRI` containing the indices of the 3 vertices of each triangles. Table 1 gives a piece of Matlab code (initialization and input-output omitted) to compute the building-block matrices $\mathbf{Grad}_\mu$, $\mathbf{Div}_\mu$, $\mathbf{T}^{t\to v}$ and $\mathbf{T}^{v\to t}$. The formula for the gradient originates from finite volume or finite element discretization scheme for reaction-diffusion equations [5, 20–23]. Note that this code could also be used to compute the stiffness matrix of reaction-diffusion systems. The function `sparse` automatically adds together elements with duplicate values of row/column. The matrices $\mathbf{DGrad}_\mu$ and $\mathbf{CGrad}_\mu$ (variables `DGrad.`μ and `CGrad.`μ) are then computed using Eq. (25).

Implementation of the algorithm for the case $D \to \infty$ or for generating the initial condition is straightforward. Matlab code for assembling the linear system is given in Table 2. The linear system (complex non-symmetric) is solved with the backslash (\) operator. In the iterations of the eikonal-diffusion solver, the linear system is assembled using the piece of Matlab code shown in Table 3. Note, however, that the matrix $\mathbf{A}$ is singular ($\mathbf{Ae} = \mathbf{0}$, where $\mathbf{e}$ is now a n_v-by-1 vector with 1 in each entry). It stems from the fact that Eq. (7) is invariant to global phase shift $\phi \mapsto \phi \exp(i\bar{\theta})$ where $\bar{\theta}$ is a constant. The solution at step s is obtained by solving the system (deflation method [24])

$$\left(\mathbf{A}(\boldsymbol{\phi}^s) + \frac{1}{n_v}\, \mathbf{e}\,\mathbf{e}^* \right) \boldsymbol{\theta}^s = \mathbf{f}(\boldsymbol{\phi}^s) \ . \tag{32}$$

Since $\mathbf{A}$ is real non-symmetric, a biconjugate stabilized gradient method (BiCGstab) with incomplete LU preconditioner was used and implemented in Matlab as

```
[L,U] = luinc(A,1e-6);
Adefl = @(x)(A*x+mean(x)*ones(nv,1));
theta = bicgstab(Adefl,f,1e-10,100,L,U);
alpha = mean(theta);
theta = theta - alpha;
```

in order to avoid the allocation of full matrices. The preconditioner served to accelerate convergence. In the test cases considered (2D meshes and atrial surfaces), no ill-conditioned system was encountered. The next subsection will explain the meaning of the variable `alpha` (α).

4.5. Determination of the period T of the reentry

The eikonal-diffusion problem requires to specify the geometry and the conduction properties. The definition of $\mathbf{c}$ and $\mathbf{D}$ also assumes that the period T of the reentry is *a priori* known. For a given reentrant pathway (or initial condition ϕ^0), however, there is only one value of T for which a solution to the eikonal-diffusion equation exists. For example, if a reentry propagates at a speed CV along a pathway of length L with a period T, the relation $CV \cdot T = L$ must hold. We are going to show that the algorithm is not affected by an incorrect choice of T for the definition of $\mathbf{c}$ and $\mathbf{D}$. Moreover, the analysis will enable us to *determine* the period T once convergence is reached.

Both $\mathbf{c}$ and $\mathbf{D}$ are proportional to the period T [Eq. (10)]. To analyze the effect of T, assume both $\mathbf{c}$ and $\mathbf{D}$ are scaled by the same factor $\lambda > 0$, i.e., $\mathbf{c} \mapsto \lambda\mathbf{c}$ and $\mathbf{D} \mapsto \lambda\mathbf{D}$. From the definitions of $\mathbf{A}$ [Eq. (26)] and $\mathbf{f}$ [Eq. (30)], the linear system (32) becomes :

$$\left(\lambda\mathbf{A} + \frac{1}{n_v}\,\mathbf{e}\,\mathbf{e}^*\right)\boldsymbol{\theta}_\lambda = \lambda\mathbf{f} + (\lambda - 1)\,\mathbf{e}\ . \tag{33}$$

If $\boldsymbol{\theta}_1$ is the solution for $\lambda = 1$, the general solution $\boldsymbol{\theta}_\lambda$ is given by $\boldsymbol{\theta}_1 + g(\lambda)\mathbf{e}$, where $g(\lambda) = (\lambda - 1)(\mathrm{mean}(\theta_1) - 1)$, as shown by direct substitution. Therefore, if the mean is subtracted from the correction θ at each iteration (as done in the Matlab code in the previous paragraph), the iterative scheme is invariant by scaling of $\mathbf{c}$ and $\mathbf{D}$, *i.e.*, $\theta_\lambda - \mathrm{mean}(\theta_\lambda) = \theta_1 - \mathrm{mean}(\theta_1)$. Note that if the conduction property $(\sigma/\beta C_m)$ was scaled by λ, we would have a different transformation, namely $\mathbf{c} \mapsto \sqrt{\lambda}\,\mathbf{c}$ and $\mathbf{D} \mapsto \lambda\mathbf{D}$.

Since the mean of θ^s is subtracted after each iteration in our implementation, the algorithm converges if and only if θ^s tends to a constant field $\alpha\,\mathbf{e}$. Unless $\alpha = 0$, $\mathbf{f}(\boldsymbol{\phi}^s)$ does not tend to zero, but rather to $\alpha\,\mathbf{e}$ [Eq. (32)]. This means that the equation has no solution for this parameter set (geometry, $\mathbf{c}$ and $\mathbf{D}$) because the estimated period $\tilde{T}$ was inexact. However, there exists a scaling factor λ such that $\mathbf{f}(\boldsymbol{\phi}^s)$ tends to zero with the exact same sequence $\boldsymbol{\phi}^s$. This factor is obtained by setting the right hand side of (33) to zero, leading to $\lambda = (1+\alpha)^{-1}$. The corrected period of the reentry is therefore

$$T = \frac{\tilde{T}}{1+\alpha}\ , \tag{34}$$

and the effective diffusion tensor is $(1+\alpha)^{-1}\mathbf{D}$.

5. Status report

5.1. Reentrant activation maps

To illustrate the capabilities of the eikonal-diffusion solver, reentrant activation maps were computed in a model of ventricular slice (5,050 nodes, $\Delta x = 0.82$ mm) with uniform isotropic conduction properties. Scaled activation time τ was set to $2\pi\ell/L$ along the boundary corresponding to the left ventricle (circuit indicated in Fig. 1A), where L is the circuit length and ℓ is the curvilinear coordinate along the circuit. Figure 1A shows the activation map resulting from Laplacian interpolation of known activation times. Activation maps in Figs. 1B–F were computed using the eikonal-diffusion equation with scaled conduction velocity $c = L/2\pi$ and decreasing values of the diffusion coefficient D from 1 cm^2 down to 0.15 cm^2. With lower diffusion of activation times, local isochrone curvature can be higher and the space constant of boundary effects is shorter. Previous works [9] have demonstrated that these reentrant activation maps correspond well to those computed with a monodomain model when Eq. (10) is used to specify eikonal-diffusion parameters, even in the presence of anisotropy or in a geometry with more complex topology.

5.2. Period of reentry

The new formula for estimating the period of the reentry was tested by comparing the eikonal-diffusion model to a monodomain reaction-diffusion model using the Luo-Rudy membrane model [25], a membrane capacitance of $C_m = 1$ μF/cm^2 and a surface-to-volume ratio of $\beta = 2000$ cm^{-1}. A finer mesh (32,820 nodes, $\Delta x = 0.32$ mm) of the ventricular slice model (Fig. 1) and finite volume discretization [20] were used for monodomain simulations. Tissue conductivity σ was varied so that plane wave conduction velocity ranged from 34.1 to 54.5 cm/s. In each case, a counter-clockwise reentry was initiated around the left ventricle [9] (like the activation patterns in Fig. 1) and the period of reentry was reported (after stabilization of the reentry). Eikonal-diffusion parameters were set according to Eq. (11) with $k_m = 2.0833$ ms^{-1} and $\tilde{T} = 1000$ ms. Figure 2 compares the periods obtained in the monodomain model and in the eikonal-diffusion model [Eq. (34)]. Another estimate is given by the length L of reentrant pathway divided by plane wave conduction velocity CV. The results demonstrate the effect of curvature-dependent conduction velocity (prolonging the period by about 15 ms) captured by the eikonal-diffusion model. At slow conduction velocity, estimated period is in agreement with the monodomain model (the error is about 2–3 ms or 1%). At faster conduction velocity, however, conduction velocity restitution is involved because of the shorter cycle length. Since the eikonal approach ignores any repolarization or restitution effect, the period of reentry is underestimated.

5.3. Interpolation of Activation times

Lines *et al.* stated the problem of synchronizing a simulation with signals recorded at sparse measurement sites [8]. The eikonal-diffusion equation provides another approach to tackle this problem. A computer model of macroreentrant atrial arrhythmia will serve as test case to demonstrate the potential of the eikonal approach for this application.

A stable reentry was simulated in a simplified 3D computer model (800,000 cubic elements) of the human atria [6, 9] using the Courtemanche *et al.* membrane model [26]. Conduction properties were isotropic and uniform. The epicardium was meshed with coarse triangular elements (13,798 nodes, $\Delta x = 1.2$ mm). Activation times (defined with a threshold at -60 mV) were extracted at each node of the triangular mesh. The resulting activation map served as reference. The problem will be to recover this activation map with activation times known at a limited number (N_e) of locations.

In a first step, $N_e = 60$ electrodes were uniformly distributed over the atrial surface. Activation times were interpolated using Eq. (17) where Γ is the set of N_e electrode positions (Laplacian interpolation). Figure 3A displays both the interpolated and the reference activation map. The reconstructed map was qualitatively correct and the 2π jump due to the periodic nature of the reentry was appropriately handled. The root mean square (RMS) error in activation time was 6.8 ms (the period was $T = 197$ ms). This map served as initial condition for the eikonal-diffusion solver, assuming that conduction properties are known. Eikonal-diffusion parameters were set according to Eq. (10). The resulting activation map is shown in Fig. 3B. Eikonal-diffusion isochrones (dashed lines) match those obtained in the monodomain model (solid lines), even close to sites of wavefront collisions. RMS error in activation time was 0.72 ms.

To estimate the number of electrodes necessary to reach a sufficient level of accuracy in the reconstructed map, N_e electrodes were randomly distributed throughout the atrial surface, with N_e varying from 10 to 200. For each number of electrodes N_e, 100 random electrode configurations were generated. In each case, activation maps were interpolated from those N_e values using both Laplacian interpolation and the eikonal-diffusion solver. Qualitative correspondence between reconstructed and reference activation maps was assessed by comparing the winding numbers, defined as $(2\pi)^{-1} \oint \nabla\tau \cdot d\mathbf{s}$ (= integer), where the contour integral is calculated around each anatomical obstacle. Qualitative correspondence was said to be correct when the winding number around each of the 8 anatomical obstacles matched those of the reference activation map. Quantitative correspondence was measured using RMS error in activation time. The percentage of qualitatively incorrect reconstructed maps was 85%, 34% and 18% for $N_e = 10$, 20 and 30, and was $\leq 3\%$ for $N_e > 35$. Reconstruction was qualitatively incorrect when the location of the electrodes was too far from the anatomical reentrant pathway (here the valves) to enable its identification. For example, if no electrode is located between the tricuspid valve and the inferior vena cava, it may not be possible to differentiate between a reentry around the tricuspid valve and a reentry around the inferior vena cava. These cases were excluded from subsequent quantitative analysis. For qualitatively correct cases of Laplacian interpolation, RMS error decreased when more electrodes were used, following approximately a power law with an exponent of -0.6, as shown in Fig. 4. By using *a priori* information about conduction properties, the eikonal-diffusion approach was able to make all (qualitatively correct) interpolated maps converge to exactly the same activation map, in fact also the same as that of Fig. 3B. The RMS error was always 0.72 ms (0.37% of the period).

5.4. Performance of the algorithm and its implementation

To measure the computational cost of the proposed algorithms, test cases were run and profiled using Matlab 7.9 on a Linux computer with Intel Core i7-950 CPU (3.07 GHz). Finite element matrices (Matlab code in Table 1) were constructed in about 0.65 s for

100,000 triangles. Because the eikonal approach does not require a fine mesh (typically 10,000 nodes are sufficient), preprocessing time is essentially negligible (0.08 s for the atrial mesh).

Interpolation with fixed point iterations (Subsect. 4.2) was tested on the atrial activation time interpolation problem of the previous subsection (13,798 nodes). The number of iterations needed and the CPU time depended on the number of measurement sites N_e. The more data available, the faster the convergence (34±22 iter./5.4±3.5 s CPU time for $N_e = 20$; 15±3 iter./2.5±0.5 s for $N_e = 60$; 11±1 iter./2.2±0.2 s for $N_e = 100$). More than 75% of CPU time was spent solving complex linear systems (intrinsic Matlab operator).

Iterations of the eikonal-diffusion solver are more computational expensive since the BiCGstab algorithm has been implemented in Matlab. One iteration on the 13,798-nodes atrial mesh took 1.3±0.1 s, depending on the number of sub-iterations in the linear solver. Typically 15 to 30 eikonal-diffusion iterations were needed (20 to 40 s CPU time). About 95% of this CPU time was spent in the preconditioner and in the linear solver. An optimized, parallelized compiled code might enhance the performance in this case since the bottleneck is clearly linear system solving.

6. Lessons learned

In a previous paper, the eikonal-diffusion equation was generalized to reentrant activation patterns in two-dimensional domains or three-dimensional surfaces [9]. The resulting activation maps were shown to reproduce those obtained in the framework of the monodomain model. These eikonal-based activation maps were also used to generate initial conditions for the monodomain model. This provided a new approach for creating a library of simulated arrhythmias with different pathways of reentry and for running simulations reproducing sparse experimental measurements of activation times.

Here, the finite-element-based numerical methods presented in [9] were reformulated in order to improve and facilitate computer implementation in a compact and efficient Matlab code. The crucial importance of deflation for solving the singular linear system (a method chosen empirically in [9]) was revealed by a theoretical argument. Thank to a scaling property, the eikonal-diffusion solver is now able to automatically identify the period of the reentry for which a reentrant solution exists. This also reduces the number of input parameters needed to generate reentrant activation maps. Another way to look at it is to consider that the period is fixed and that the solver finds the appropriate conduction velocity.

Convergence of the eikonal-solver depends on the quality of the initial estimate. This paper uses Laplacian interpolation of known activation times as initial estimate, generalizing previous works [9]. This form of interpolation is a special case of the eikonal-diffusion equation with the diffusion coefficient $D \to \infty$. Instead of applying the eikonal-diffusion solver in this case, another, more robust, algorithm based on fixed point iterations was developed. These iterations converge from a wider range of initial estimates, even white noise or a constant field. In contrast, Newton iterations tend to preserve phase singularities since corrections are small and smooth [9]. This property is desirable for final adjustment of the activation map, but possibly incorrect phase singularities present at the initial stage may not be eliminated. Convergence of fixed point iterations is asymptotically slower since the method is of first order as compared to the second order Newton method. However, CPU time per iteration was found to be shorter. If necessary, it could be possible to switch from fixed point to Newton iterations once close enough to the solution. Note that the Laplacian interpolation algorithm applies to any type of orientation field or angular data (phase, fiber angle, electric/magnetic dipole orientation).

The problem of reconstructing an activation map based on sparse measurements of activation times was considered. A large number (> 100) of measurements was necessary to accurately interpolate an activation map, in agreement with Lines *et al.* [8]. When tissue conduction properties are known, however, the activation pattern can be accurately reproduced from a limited number of electrodes using the eikonal-diffusion approach. Then the resulting activation map can serve as initial condition for simulating the reentry in a monodomain model [9] or in a bidomain model after a straightforward extension. Note that this could form the basis for solving the inverse problem of extracting conduction properties from reentrant activation maps.

A fundamental limitation of the eikonal approach is that repolarization is ignored. This has several consequences. First, the period of reentry is underestimated when it is close to the effective refractory period due to conduction velocity restitution. Second, reentrant circuits may not be stable when simulated in a monodomain model due to action potential duration restitution. They may even self-terminate by conduction blocks if action potential durations are too long. On the other hand, these limitations may be used as features to initiate different types of arrhythmias with various pathways of reentry and dynamical regimes.

7. Future plans

The eikonal-diffusion equation provides at low computational cost an accurate representation of the isochrones of a reentry. Since today's computer power enables us to simulate reentries in large scale reaction-diffusion models with detailed description of membrane kinetics, the goal is not to reduce the complexity of the system but rather to develop new tools for initiating clinically relevant arrhythmias in computer models. Moreover, activation maps computed in a coarse eikonal-diffusion model can be used as initial condition [9] in more structurally-realistic volumetric models incorporating anisotropic fiber bundles and anatomical structures [27, 28]. This will facilitate the creation of a database of simulated episodes of arrhythmias including activation maps and electrical signals (electrograms, electrocardiograms) that could be used to evaluate signal processing or diagnostic tools and guide therapeutic interventions. During a clinical intervention, full propagation map may be inferred from a set of sparse measurements to guide toward critical points in the circuits.

Acknowledgment

This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). The author thanks Alain Vinet (Université de Montréal) for helpful discussions and comments, and the Lausanne Heart Group (Ecole Polytechnique Fédérale de Lausanne) for sharing simulation tools and the atrial geometry.

References

[1] V. Jacquemet, L. Kappenberger, C. S. Henriquez, Modeling atrial arrhythmias: Impact on clinical diagnosis and therapies, IEEE Rev Biomed Eng 1 (2008) 94–114.

[2] D. Harrild, C. Henriquez, A computer model of normal conduction in the human atria, Circ Res 87 (2000) E25–36.

[3] G. Seemann, C. Hoper, F. B. Sachse, O. Dossel, A. V. Holden, H. Zhang, Heterogeneous three-dimensional anatomical and electrophysiological model of human atria, Philos Transact A Math Phys Eng Sci 364 (2006) 1465–81.

[4] E. J. Vigmond, R. Ruckdeschel, N. Trayanova, Reentry in a morphologically realistic atrial model, J Cardiovasc Electrophysiol 12 (2001) 1046–54.

[5] L. Wieser, H. E. Richter, G. Plank, B. Pfeifer, B. Tilg, C. N. Nowak, G. Fischer, A finite element formulation for atrial tissue monolayer, Methods Inf Med 47 (2008) 131–9.

[6] V. Jacquemet, A. van Oosterom, J. M. Vesin, L. Kappenberger, Analysis of electrocardiograms during atrial fibrillation. a biophysical model approach, IEEE Eng Med Biol Mag 25 (2006) 79–88.

[7] R. Plonsey, R. C. Barr, Bioelectricity: A Quantitative Approach, Kluwer Academic Plenum Publishers, 2000.

[8] G. T. Lines, M. C. MacLachlan, S. Linge, A. Tveito, Synchronizing computer simulations with measurement data for a case of atrial flutter, Ann Biomed Eng 37 (2009) 1287–93.

[9] V. Jacquemet, An eikonal approach for the initiation of reentrant cardiac propagation in reaction-diffusion models, IEEE Trans Biomed Eng 57 (2010) 2090–2098.

[10] K. A. Tomlinson, P. J. Hunter, A. J. Pullan, A finite element method for an eikonal equation model of myocardial excitation wavefront propagation, SIAM J Appl Math 63 (2002) 324–350.

[11] M. Sermesant, Y. Coudiere, V. Moreau-Villeger, K. S. Rhode, D. L. G. Hill, R. S. Razavi, A fast-marching approach to cardiac electrophysiology simulation for XMR interventional imaging, Proc. MICCAI 3750 (2005) 607–615.

[12] P. C. Franzone, L. Guerri, S. Rovida, Wave-front propagation in an activation model of the anisotropic cardiac tissue – Asymptotic analysis and numerical simulations, J Math Biol 28 (1990) 121–176.

[13] J. P. Keener, J. Sneyd, Mathematical Physiology, Interdisciplinary applied mathematics ; v. 8, Springer, New York, 2nd edition, 2001.

[14] T. Oostendorp, A. van Oosterom, G. Huiskamp, Interpolation on a triangulated 3D surface, J Comput Phys 80 (1989) 331–343.

[15] T. Cecil, S. Osher, L. Vese, Numerical methods for minimization problems constrained to S_1 and S_2, J Comput Phys 198 (2004) 567–579.

[16] P. Chinchapatnam, K. S. Rhode, M. Ginks, C. A. Rinaldi, P. Lambiase, R. Razavi, S. Arridge, M. Serme-

sant, Model-based imaging of cardiac apparent conductivity and local conduction velocity for diagnosis and planning of therapy, IEEE Trans Med Imaging 27 (2008) 1631–42.

[17] J. Relan, P. Chinchapatnam, M. Sermesant, K. Rhode, H. Delingette, R. Razavi, N. Ayache, Coupled personalisation of electrophysiology models for simulation of induced ischemic ventricular tachycardia, Med Image Comput Comput Assist Interv 13 (2010) 420–8.

[18] T. Alkhalifah, Traveltime computation with the linearized eikonal equation for anisotropic media, Geophys Prospect 50 (2002) 373–382.

[19] H. P. Langtangen, Computational Partial Differential Equations – Numerical Methods and Diffpack Programming, volume 1 of *Texts in Computational Science and Engineering*, Springer, 2003.

[20] V. Jacquemet, C. S. Henriquez, Finite volume stiffness matrix for solving anisotropic cardiac propagation in 2-D and 3-D unstructured meshes, IEEE Trans Biomed Eng 52 (2005) 1490–2.

[21] D. J. Rose, H. Shao, C. S. Henriquez, Discretization of anisotropic convection-diffusion equations, convective M-matrices and their iterative solution, VLSI Design 10 (2000) 485–529.

[22] H. Shao, K. J. Sampson, J. B. Pormann, D. J. Rose, C. S. Henriquez, A resistor interpretation of general anisotropic cardiac tissue, Math Biosci 187 (2004) 155–174.

[23] S. Zozor, O. Blanc, V. Jacquemet, N. Virag, J.-M. Vesin, E. Pruvot, L. Kappenberger, C. S. Henriquez, A numerical scheme for modeling wavefront propagation on a monolayer of arbitrary geometry, IEEE Trans Biomed Eng 50 (2003) 412–420.

[24] M. S. Lynn, W. P. Timlake, The use of multiple deflations in the numerical solution of singular systems of equations to potential theory, SIAM. J. Numer. Anal. 5 (1968) 303–322.

[25] C.-H. Luo, Y. Rudy, A model of the ventricular cardiac action potential, Circ. Res. 68 (1991) 1501–1526.

[26] M. Courtemanche, R. J. Ramirez, S. Nattel, Ionic mechanisms underlying human atrial action potential properties: insights from a mathematical model, Am J Physiol 275 (1998) H301–21.

[27] M. W. Krueger, F. M. Weber, G. Seemann, O. Dössel, Semi-automatic segmentation of sinus node, Bachmann's bundle and terminal crest for patient specific atrial models, in: World Congress on Medical Physics and Biomedical Engineering, volume 25/4 of *IFMBE Proceedings*, 1996, pp. 673–676.

[28] M. W. Krueger, K. Rhode, F. M. Weber, D. U. J. Keller, D. Caulfield, G. Seemann, B. R. Knowles, R. Razavi, O. Dössel, Patient-specific volumetric atrial models with electrophysiological components: A comparison of simulations and measurements, Biomed Technik 55 (2010) 54–57.

Table 1: Matlab code for computing the building-block matrices $\mathbf{Grad}_\mu$ (`Grad.`μ), $\mathbf{Div}_\mu$ (`Div.`μ), $\mathbf{T}^{t\to v}$ (`Tt2v`) and $\mathbf{T}^{v\to t}$ (`Tv2t`) for a triangular mesh defined by its `nv` vertices (`VER`) and `nt` triangles (`TRI`)

```
u = VER(TRI(:,2),:)-VER(TRI(:,1),:);                   % (u,v) is the basis of the triangle
v = VER(TRI(:,3),:)-VER(TRI(:,1),:);
u2 = sum(u.^2,2); v2 = sum(v.^2,2); uv = sum(u.*v,2);  % compute scalar products
delta = u2.*v2 - uv.^2;                                % determinant
e = [1 1 1];
us = (v2./delta)*e .* u - (uv./delta)*e .* v;         % (us,vs) is the dual basis of (u,v)
vs = (u2./delta)*e .* v - (uv./delta)*e .* u;
Gx = [-us(:,1)-vs(:,1) us(:,1) vs(:,1)];               % components of the gradient matrix
Gy = [-us(:,2)-vs(:,2) us(:,2) vs(:,2)];
Gz = [-us(:,3)-vs(:,3) us(:,3) vs(:,3)];
I = TRI; J = repmat(1:nt,1,3);                         % indices in the sparse matrices
Grad.x = sparse(J,I,Gx,nt,nv);                         % create the sparse matrices
Grad.y = sparse(J,I,Gy,nt,nv);
Grad.z = sparse(J,I,Gz,nt,nv);

St = repmat(sqrt(delta)/2,1,3);                        % area of the triangles
Tt2v = sparse(I,J,St,nv,nt);                           % temporary matrix
Sv = sum(Tt2v,2)/3;                                    % area associated with each vertex
invS = spdiags(1./Sv,0,nv,nv);                         % inverse of vertex area
Tt2v = invS/3 * Tt2v;                                  % interpolation triangle to vertex
Tv2t = sparse(J,I,1/3,nt,nv);                          % interpolation vertex to triangle

Div.x = invS*sparse(I,J,-St.*Gx,nv,nt);                % create the sparse matrices
Div.y = invS*sparse(I,J,-St.*Gy,nv,nt);
Div.z = invS*sparse(I,J,-St.*Gz,nv,nt);
```

Table 2: Matlab code for computing the linear system $\mathbf{A}^\infty$ (`Ainf`) and $\mathbf{f}^\infty$(`finf`) as a function of ϕ (`phi`) for Dirichlet boundary condition on Γ (`Gamma`)

```
phit = spdiags(Tv2t*conj(phi),0,nt,nt);          % value of phi on the triangles
Ainf = Div.x * phit * DGrad.x ...                % matrix A_inf
     + Div.y * phit * DGrad.y ...
     + Div.z * phit * DGrad.z;
Ainf(Gamma,:) = 0;                               % replace row i with row i of the identity matrix
Ainf(sub2ind([nv nv],Gamma,Gamma)) = 1;
finf = zeros(nv,1); finf(Gamma) = phi(Gamma);   % right hand side f_inf
```

Table 3: Matlab code for computing the linear system $\mathbf{A}$ (`A`) and $\mathbf{f}$ (`f`) as a function of ϕ (`phi`)

```
Diff = Div.x * DGrad.x + Div.y * DGrad.y + Div.z * DGrad.z;   % diffusion operator
phis = conj(phi);  phit = Tv2t*phis;
Norm = sqrt( abs(CGrad.x * phi).^2 + abs(CGrad.y * phi).^2... % norm of the gradient
     + abs(CGrad.z * phi).^2 );
cv = 1./Norm;                                                 % propagation velocity
B = spdiags(CGrad.x*phis,0,nt,nt) * CGrad.x ...               % matrix B
  + spdiags(CGrad.y*phis,0,nt,nt) * CGrad.y ...
  + spdiags(CGrad.z*phis,0,nt,nt) * CGrad.z;
A = Diff + 3/4 * Tt2v * (spdiags(cv,0,nt,nt) * ...            % matrix A
    imag(spdiags(Tv2t*phi,0,nt,nt) * B )) ...
  + 1/4 * imag(spdiags(phi,0,nv,nv) * ...
    Tt2v * (spdiags(cv,0,nt,nt) * B) );
f = Tt2v * (Norm - 1) ...                                     % right hand side f
  - Div.x * imag( phit .* (DGrad.x * phi) ) ...
  - Div.y * imag( phit .* (DGrad.y * phi) ) ...
  - Div.z * imag( phit .* (DGrad.z * phi) );
```

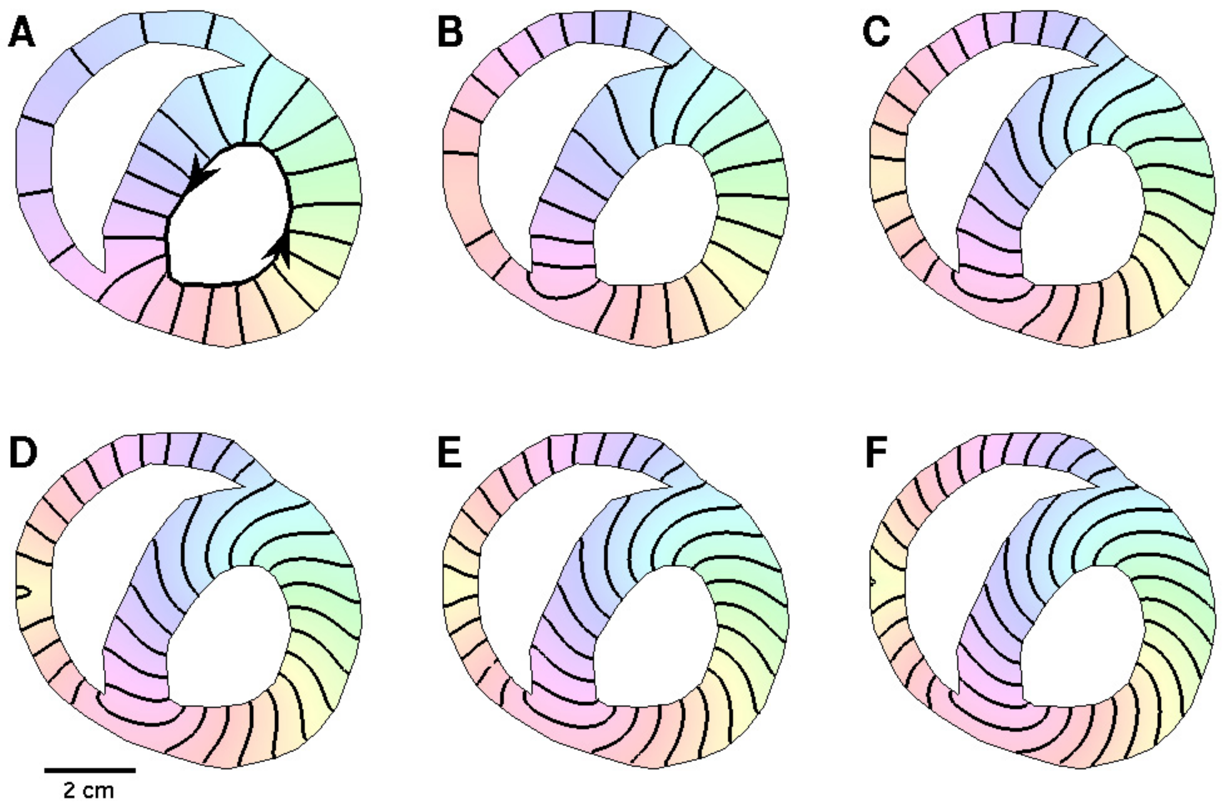

Figure 1: Activation maps computed using the eikonal-diffusion equation in a model of ventricular slice. Activation time is color-coded. Isochrones (black curves) are displayed every $T/20$ where T is the period. (A) Laplacian interpolation ($D = \infty$). The circuit serving as boundary condition is shown as a thick oriented curve; (B)–(F) Solution to the eikonal-diffusion equation with $D = 1$, 0.5, 0.4, 0.3 and 0.15 cm^2 from (B) to (F) respectively.

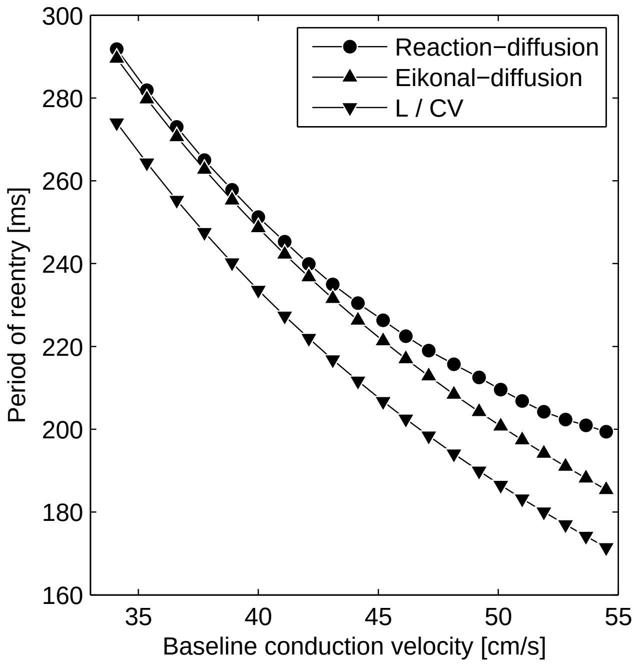

Figure 2: Period of reentry as a function of plane wave (baseline) conduction velocity in a ventricular slice model. *Circles*: period of reentry simulated in a monodomain model (reference value); *Upward triangles*: estimate of the period based on the eikonal-diffusion equation [Eq. (34)]; *Downward triangles*: estimate of the period obtained by dividing the pathway length (L) by the plane wave conduction velocity (CV).

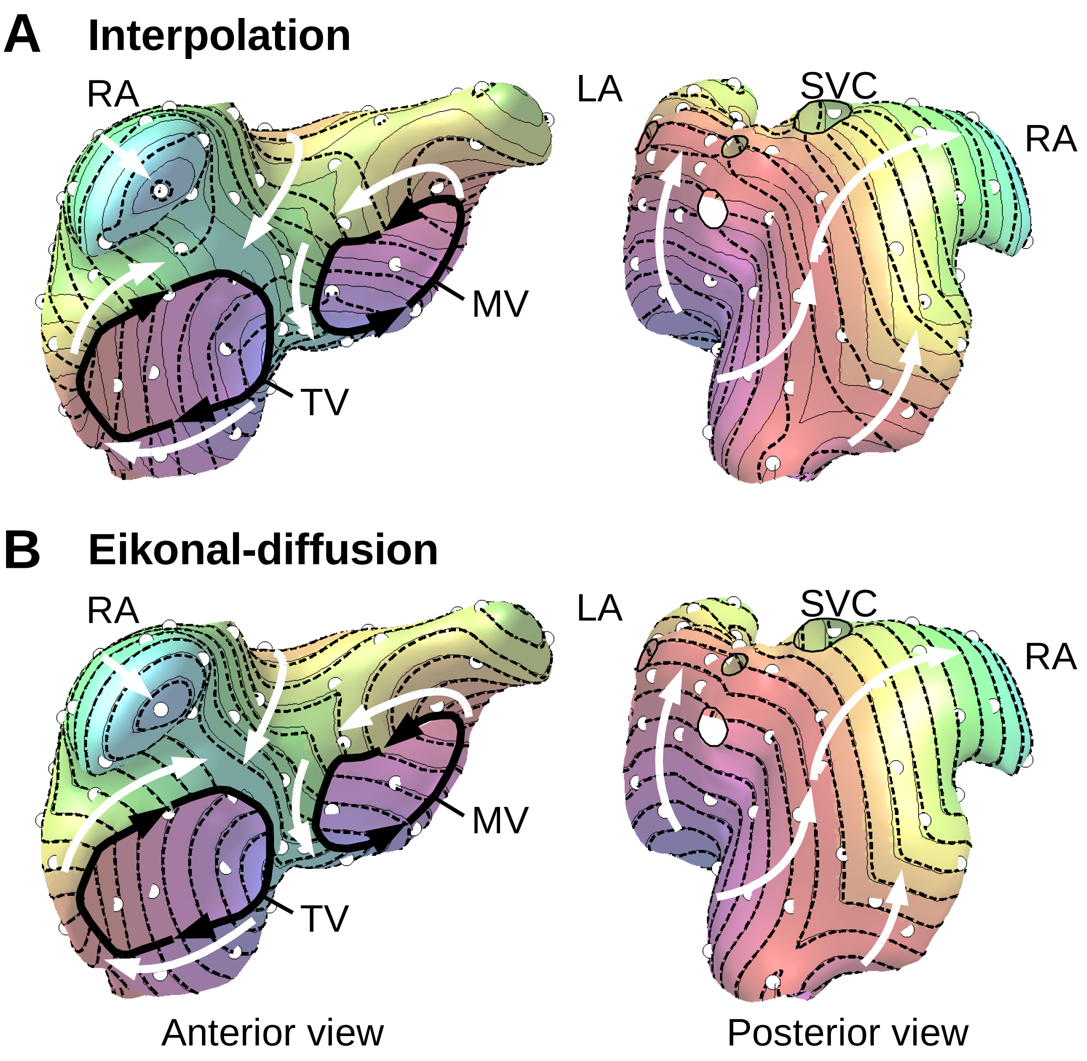

Figure 3: Reconstruction of activation maps from sparse measurement data in geometry of the atria. Activation time is color-coded. Isochrones (solid lines: reference; dashed lines: reconstructed) are displayed every $T/20$ where T is the period. Arrows indicate the direction of wavefront propagation. White circles denote the position of the $N_e = 60$ electrodes. (A) Laplacian interpolation. (B) Solution to the eikonal-diffusion. RA: right atrium; LA: left atrium; TV: tricuspid valve; MV: mitral valve; SVC: superior vena cava.

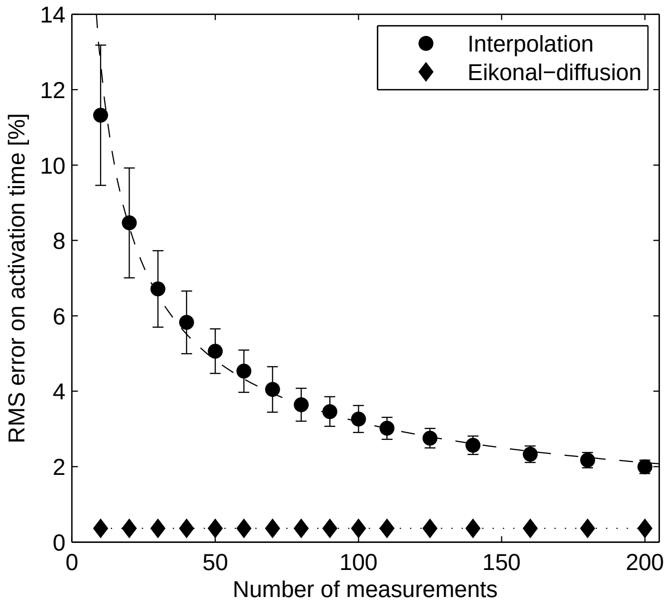

Figure 4: Root mean square (RMS) error between reconstructed (*Circles*: Laplacian interpolation; *Diamonds*: eikonal-diffusion approach) and reference activation maps (monodomain simulations) in percentage of the period $T = 197$ ms, as a function of the number N_e of measurement electrodes. RMS error is averaged over 100 electrodes configurations (mean±standard deviation is shown as error bars). The dashed line is a power law function fitted to data points.