BeatBox — HPC Simulation Environment for Biophysically and Anatomically Realistic Cardiac Electrophysiology

Mario Antonioletti
EPCC
Edinburgh, UK
m.antonioletti@epcc.ac.uk

Vadim N. Biktashev *
University of Exeter
Harrison Building, North Park
Road
Exeter EX4 4QF, UK
v.n.biktashev@exeter.ac.uk

Adrian Jackson
EPCC
Edinburgh, UK
a.jackson@epcc.ac.uk

Sanjay R. Kharche
Harrison Building, North Park
Road
Exeter EX4 4QF, UK
s.r.kharche@exeter.ac.uk

Tomas Stary
University of Exeter
Harrison Building, North Park
Road
Exeter EX4 4QF, UK
t.stary@exeter.ac.uk

Irina V. Biktasheva †
University of Liverpool
Ashton building, Ashton street
Liverpool L69 3BX, UK
ivb@liverpool.ac.uk

ABSTRACT
BeatBox combines flexible script language user interface with robust solvers to set up cardiac electrophysiology in-silico experiments without re-compilation. It is free C software run on Unix platforms sequentially or in MPI, for multiscale modelling from single cell to whole heart simulations with run-time local/global measurements, cardiac re-entry tip/filament tracing, etc. Extensible solvers, cell and anatomy repositories provide open framework for new developments in the field.

Keywords
Cardiac simulations; realistic anatomy; reaction-diffusion system; finite difference solver; irregular domain; MPI parallelisation.

1. BACKGROUND
Cardiovascular disease (CVD) is the main cause of death in Europe, accounting for 47% of all deaths [30]. Cardiac arrhythmias, where the electrical activity of the heart responsible for its pumping action is disturbed, are among the most serious CVDs. Despite over a century of study, the circumstances from which such fatal cardiac arrhythmias arise are still poorly understood. Although several advancements have been made in linking genetic mutations to arrhythmogenic CVD [13, 32, 41], these do not explain the resultant mechanisms by which arrhythmia and fibrillation emerge and sustain at the whole heart level, for the position of the heart in torso makes in vivo measurement awkward and invasive, prohibitively so for study in humans. Thus, for some genetic cardiac diseases, the first presenting symptom is death with understandably limited opportunity to make even superficial examinations in vivo. The most modern experimental methods do not provide sufficient temporal and spatial resolution to trace down the multi-scale fine details of fibrillation development in samples of cardiac tissue, not to mention the heart in vivo.

Combination of mathematical modelling [8, 4, 5, 9, 11] and the latest realistic computer simulations of electrical activity in the heart have much advanced our understanding of heart fibrillation and sudden cardiac death[10, 6, 22, 23], and the impact of in-silico modelling, or indeed in-silico "testing", is expected to increase significantly as we approach the ultimate goal of the whole-heart modelling. With the vast amount of quantitative experimental data on cardiac myocytes action potential and the underlying transmembrane ionic currents ready for inclusion into the in-silico modeling, and the recent advance in high-resolution DT-MRI provision of detail anatomy models, the biophysically and anatomically realistic computer simulations allow unimpeded access to the whole heart with greater spatial and temporal resolutions than in a wet experiment, and allow to synthesise such elusive phenomena for closer study, hence improving prospects of their treatment and prevention.

The biophysically and anatomically realistic simulation of cardiac action potential propagation through the heart is computationally expensive due to the huge number of equations per cell and the vast spacial and temporal scales required. Complexity of realistic cardiac simulations spans multi-physical scales to include greater detail at cellular level, tissue heterogeneity, complex geometry and anisotropy of the heart. Due to huge number of strongly nonlinear equations to be solved on the vast temporal and spatial scales determined by the high-resolution DT-MRI anatomy mod-
els, its timely running relies on use of parallel processors - High Performance Computing (HPC).

To address the intrinsically modular cardiac electrophysiology in silico modelling, we developed modular software package BeatBox [27], with a built-in simulation script interpreter, extendable repositories of cell and tissue/anatomy models, capable to run both sequentially and in parallel on distributed (MPI) memory architecture, fig.1. The Beatbox cardiac simulation environment allows setup of complicated numerical experiments without re-coding at low-level, so that cell excitation, tissue and anatomy models, stimulation protocols may be included into a script, and BeatBox simulation run either sequentially or in parallel without re-compilation. Importantly, the BeatBox modular paradigm provides an open framework for new developments in the field, for the open source BeatBox solvers, and cell and tissue/anatomy repositories are extended via robust and flexible interfaces. This paper gives an overview of the current state of BeatBox cardiac electrophysiology simulator together with description of the main computational methods and MPI parallelisation approaches.

2. CARDIAC TISSUE MODELS

Computer simulation of cardiac muscle requires a mathematical model, describing the relevant biophysical and electrophysiological processes. The bidomain model considers intracellular and extracellular spaces in the syncytium of cardiac myocytes. Those two domains are separated from each other by cellular membranes, the conductivity through which is controlled by ionic channels. This situation is described by a system of partial (PDE) and ordinary (ODE) differential equations of the form:

\[
C_m \frac{\partial V}{\partial t} = -I_{ion}(V, g) + \frac{1}{\chi} \nabla \cdot \bar{\sigma}_1 \nabla \Phi_1, \\
\nabla \cdot (\bar{\sigma}_1 + \bar{\sigma}_e) \nabla \Phi_1 = \nabla \cdot \bar{\sigma}_e \nabla V - I_{ext}, \\
\frac{\partial g}{\partial t} = f(g, V, \bar{r}),
\]

(1)

where \( V \) is the transmembrane voltage, \( \Phi_1 \) is the intracellular electrostatic potential (so \( \Phi_1 = V \) is the extracellular potential), \( \bar{\sigma}_1 \) and \( \bar{\sigma}_e \) are the anisotropic conductance tensors of the intra- and extracellular domains respectively, \( C_m \) is the specific capacitance of the membrane and \( \chi \) is the average surface to volume ratio of the cells. The transmembrane ion currents \( I_{ion} \) are controlled by gating variables and ionic concentrations, represented by the vector \( g \). The kinetic rates are expressed in terms of the vector-function \( f \). The term \( I_{ext} \) designates the external electric current, say from experimental or defibrillation electrodes. In the system (1), the first equation is parabolic, the second is elliptic and the third effectively is a system of ODEs at every point of the tissue characterised by its location \( \bar{r} \). If the intracellular conductances are proportional, i.e. \( \bar{\sigma}_e = \nu \bar{\sigma}_1 \) for a scalar \( \nu \), then \( \Phi_1, V \) and \( g \) are proportional to each other, and the system (1) simplifies to a monodomain model:

\[
C_m \frac{\partial V}{\partial t} = -I_{ion}(V, g) + \frac{1}{\chi} \nabla \cdot \bar{\sigma}_{eff} \nabla V - I_{eff}(\bar{r}, t), \\
\frac{\partial g}{\partial t} = f(g, V, \bar{r}),
\]

(2)

where \( \bar{\sigma}_{eff} = \nu \bar{\sigma}_1 \) and \( I_{eff} = \frac{1}{\nu(1+\nu)} I_{ext} \). System (2) belongs to the class of reaction-diffusion systems, used for modelling of a large variety of natural and artificial nonlinear dissipative systems [31].

Computationally, the bidomain description is dramatically more challenging than the monodomain, as the elliptic equation has to be solved at every time step. Practice shows that unless an external electric field is involved, the bidomain models give results that differ only slightly from corresponding appropriately chosen monodomain models, which, together with the fact that experimental data on the intra- and extracellular conductivity tensors are scarce, means that in practice the monodomain simulations are used more widely.

The complexity of cardiac electrophysiology simulations is further increased as these expand to span multiple physical scales to include greater detail at the cellular level, such as cell metabolism, and greater integration with the surrounding biological systems, such as vascular fluid dynamics. In this context, it is not surprising that the timely completion of simulations relies on modern high performance computing hardware.

Use of HPC facilities, although essential, is severely limited by specialized software development skills required, so a separation of the low-level coding from the processes of formulating and solving research problems is highly desirable. The BeatBox project seeks to overcome these difficulties by providing a computational environment that could serve as a unifying paradigm for all in silico cardiac electrophysiology research, and for research in similar phenomena involving reaction-diffusion systems outside the cardiology domain.

3. LOGICAL STRUCTURE AND USER INTERFACE

The fundamental paradigm used by BeatBox is to represent a simulation as a ring of “devices”, i.e. individual modules that perform specific computational, input/output or control tasks. Module of each type can be used more than once in the ring, thus providing more than one device instance. This ring of devices, fig. 2, is constructed at startup, based on the instructions given in an input script. The script describes the sequence of devices used in a particular simulation and their parameters, using a domain-specific
to be required for a typical cardiac dynamics simulations, in the state they are currently available in at the time of writing this paper.

4. AN OVERVIEW OF COMPUTATIONAL ALGORITHMS

We believe that the main features of BeatBox are the flexibility of its user interface, and the fact that any new computational features can always be added as a new device. At the same time, it is clear that its utility at present depends on specific computational capabilities. In the next sections, we describe the components that are most likely

4.1 Splitting the problem into parts

4.1.1 Computation of intermediate expressions

“Divide and conquer” is a popular and successful strategy for evolution-type problems. The idea is to split the right-hand sides of complex evolution equations to simpler components, implement solvers corresponding to each of these components, and then coordinate the work of the solvers so together they solve the whole problem. The modular structure of a BeatBox job makes this approach particularly easy to implement. One way of doing so is by computing different parts of the right-hand side by different devices, and then allow the time-stepping device to use the results of these computations. We illustrate this by using a simple example, with reference to the sample script illustrated by fig. 2 and listed in in the Appendix. Ignoring the effect of an external electric field for now, the mathematical problem solved by the script is:

\[
\begin{align*}
\frac{\partial u}{\partial t} &= \frac{1}{\epsilon} (u - u^2/3 - v) + \nabla D \nabla u, \\
\frac{\partial v}{\partial t} &= \epsilon (u + \beta - \gamma v). 
\end{align*}
\]  

(3)

The script implements a forward Euler timestep (see Section 4.2.1 below) for this equation, using two devices: 

```
diff v0=[u] v1=[i] Dpar=D Dtrans=D/4 h=xhx;
euler v0=[u] v1=[v] ht=ht ode=fhncub
par={eps=eps bet=@[b] gam=gam Iu=@[i]};
```

In this fragment, the macro \([u]\) expands to 0, that is the very first layer of the grid allocated to the \(u\) field, \([v]\) expands 1, standing for the second layer of the grid, allocated to the \(v\) field, and \([i]\) expands to 2, which is the third layer of the grid, for the value of the diffusion term, \(\nabla D \nabla u = \chi^{-1} \nabla \sigma_{\text{eff}} \nabla u\). So, the function of \texttt{diff} is computation of 

\[(I_u)_n = \nabla D \nabla u_n\]

where \(u_n\) stands for the \(u\) field at the current time step \(n\) (regarding the approximation of the second derivatives, see below, Section 4.3); the result of this computation, denoted here as \((I_u)_n\), is placed into layer 2 of the grid. The \texttt{euler} device, with the parameter \texttt{ode=fhncub}, performs a forward Euler step for the cubic FitzHugh-Nagumo ODE system,

\[
\begin{align*}
\frac{\partial u}{\partial t} &= \frac{1}{\epsilon} (u - u^2/3 - v) + I_u, \\
\frac{\partial v}{\partial t} &= \epsilon (u + \beta - \gamma v) + I_v, 
\end{align*}
\]

in which parameters \(\epsilon\) and \(\gamma\) are given by the values of the global variables \texttt{eps} and \texttt{gam} defined previously in the script (in the included parameter file \texttt{fhn.par}), the value of parameter \(\beta\) is taken from layer \([b]\) which expands to \(3\) (parameter \(\beta\) is spatially dependent in this simulation, and layer 3 was pre-filled with values by the same \texttt{k_func} device that computed the initial conditions), the value of parameter \(I_u\) is taken from layer \([i]\) which contains the values of the anisotropic diffusion term \((I_u)_n\), computed for this time

Figure 2: “Ring of devices” set up by \texttt{sample.bbs} script (see fig. 11 in the Appendix).
step by the preceding **diff** device, and the value of parameter $I_v = 0$ by default. Overall, with $u_\alpha(x, y)$ and $v_n(x, y)$ designating the fields $u$ and $v$ at the $n$-th time step, the pair of devices computes

$$
u_{n+1} = u_n + k \left[ \frac{1}{\epsilon} \left( u_n - \frac{1}{3} u_n^2 - v_n \right) + \nabla D \nabla u_n \right],$$

$$v_{n+1} = v_n + k \epsilon (u_n + \beta(x, y) - \gamma v_n),$$

where $k$ is the time step, represented by the global variable $ht$ in the **BeatBox** script.

### 4.1.2 Operator splitting

Operator splitting is another popular “divide and conquer” strategy. In this approach, the right-hand sides still are split into simpler parts, but now an evolution sub-step is done for each part in turn, as if this part was the whole right-hand side. For example, computation of kinetics and diffusion in the right-hand side of (3) can be split into the kinetics part and diffusion part, and then one device performs the diffusion substep, and another device performs the kinetics substep. So, the **BeatBox** script fragment from Section 4.1.1 can be modified as

```c
diffstep v0=[u] v1=[i] Dpar=D Trans=D/4 hx=dx ht=dt; euler v0=[u] v1=[v] ht=dt ode=fhncub
```

where the device **diffstep** computes the diffusion term, and does a forward Euler step with it, as if this was the only term in the equation. Combined with the fact that now in the **euler** device the parameter $1u$ is not specified so it defaults to zero, the given fragment of the script implements the following computation scheme:

$$u_{n+1/2} = u_n + \frac{k}{\epsilon} \nabla D \nabla u_n,$$

$$u_{n+1} = u_{n+1/2} + k \left[ \frac{1}{\epsilon} \left( u_{n+1/2} - \frac{1}{3} u_{n+1/2}^2 - v_n \right) \right],$$

$$v_{n+1} = v_n + k \epsilon (u_n + \beta(x, y) - \gamma v_n).$$

Once again, this is just a simple example illustrating how the **BeatBox** paradigm naturally fits the idea of operator splitting. This of course applies first of all to the simplest (Lie) splitting; more sophisticated, higher-order operator splitting schemes could be implemented at the **BeatBox** script level, or on the device, i.e. a C-source code level.

### 4.2 Kinetics solvers

#### 4.2.1 Explicit solvers

Both the monomodal “reaction-diffusion” models of the form (2) or the more complicated bidomain models (1) have equations with time derivatives. Solving those equations in **BeatBox** is done as if they were ordinary differential equations, $\frac{dV}{dt} = -\frac{1}{C_m} (I_{ion}(V, g) - I_{eff}(\vec{r}, t))$ and $\frac{dg}{dt} = f(g, V, \vec{r}),$ (4) (depending on $\vec{r}$ as a parameter) either with the value of the diffusion term, computed by the corresponding diffusion device, appearing in the voltage equation, or within the operator-splitting paradigm, i.e. performing time sub-steps as if the model was restricted to the ODEs representing the reaction terms, leaving the space-dependent part of the model to be computed at alternative sub-steps.

The simplest and arguably most popular in practice solver for ODEs is the first-order explicit (time-forward) scheme known as the **forward Euler** scheme, which for a system of ODEs (4) means:

$$V_{n+1} = V_n - \frac{k}{C_m} (I_{ion}(V_n, g_n) - I_{eff}(\vec{r}, t_n)),$$

$$g_{n+1} = g_n + k f(g_n, V_n, \vec{r}),$$

where $t_n$ is the $n$-th value in the time grid, $k = t_{n+1} - t_n$ is the time step, and $V_n = V(\vec{r}, t_n), \ g_n = g(\vec{r}, t_n)$. This scheme is implemented in **BeatBox** in the **euler** device.

The Euler scheme’s well known disadvantages is its low accuracy due to only first-order approximation of the ODE, and, as any explicit scheme, only conditional stability (see e.g. [36]). The first disadvantage does not usually play a crucial role in cardiac dynamics studies as the proven accuracy of cardiac dynamics models themselves is not particularly high; there is, however, **rk4** device implementing **Runge-Kutta fourth-order scheme** for cases when accuracy is essential, and other standard explicit solvers may be easily implemented in a similar way. The stability consideration is more significant as for stiff models it severely limits the maximal allowable time step $k$, hence making simulations costly.

#### 4.2.2 Exponential solvers

The standard solution to the stability problem is, of course, using implicit or semi-implicit schemes. The latter possibility is much more popular as fully implicit approaches for nonlinear equations are numerically challenging. Among the semi-implicit approaches available in cardiac dynamics, the exponential scheme for ionic gates, known as the **Rush-Larsen technique** [37], is very popular. The idea is based on the observation that in the models of ionic excitability, since the seminal work by Hodgkin and Huxley [20], an important role is played by equations of the form:

$$\frac{dy}{dt} = \alpha(V)(1 - y) - \beta(V)y,$$

(6)

where the dynamic variable $y$, called the **gating variable**, possibly in conjunction with other gating variables, determines the permittivity of certain ionic currents. A convenient (even if not biophysically precise) interpretation is that a channel is open if all of the gates controlling that channel are open, and the variable $y$ is the probability for that gate to be open. Hence $\alpha$ and $\beta$ are transition probabilities per unit of time, of a closed gate to open, or for an open gate to close, respectively. In equation (6) the transition probabilities depend on the current value of the transmembrane voltage $V$, as in the Hodgkin-Huxley model; in more modern models gating variables of some channels may depend on other dynamical variables, say the concentration of calcium ions. The importance of the gating variables is that equations of the type (6) are often the stiffest in the whole cardiac excitation model. The Rush-Larsen scheme in its simplest form can be obtained by assuming that $V$ does not change much during a time step, $t \in [t_n, t_{n+1}]$, and replacing $V(t)$ with the constant value $V_n = V(t_n)$ turns (6) into a linear differential equation with constant coefficients, the solution of
which can be written in a closed form, which gives:

$$y_{n+1} = A(V_n) + B(V_n)y_n$$

(7)

where

$$A(V) = \exp \left( -(\alpha(V) + \beta(V))k \right),$$

$$B(V) = \frac{\alpha(V)}{\alpha(V) + \beta(V)} \left[ 1 - A(V) \right].$$

(8)

As far as equation (6) is concerned, this scheme is unconditionally stable, and gives an exact answer if \(V(t) = \text{const}\), i.e. its first-order accuracy depends exclusively on the speed of change of the transmembrane voltage \(V\). This scheme is implemented in the BeatBox device rushlarsen. Naturally, this device requires a more detailed description of the excitable model than euler: the gating variables \(y\) and their transition rates \(A, B\) need to be explicitly identified for rushlarsen whereas euler only requires a definition of the functions computing the right-hand sides of the dynamic equations, i.e. \(I_{ion}\) and \(g\).

Some modern cardiac excitation models use a Markov chain description of the ionic channels. This description is based on the assumption that an ionic channel can be in a finite number of discrete states, and transitions between the states can happen with certain probability per unit of time, which may depend on control variables, such as transmembrane voltage \(V\) or calcium ion concentration \(c\). The time evolution of the vector \(u\) of the probabilities of the channel to be in each particular state is described by the system of linear ODE, known in particular as Kolmogorov (forward) equations, or the master equation

$$\frac{du}{dt} = M(V, c)u,$$

(9)

where \(M\) is the matrix of transition rates. The extension of the Rush-Larsen idea to this system was done in [40]. Assuming again that the control variables do not change much within a time step and replacing them with a constant, \(V(t) = V_n\) and \(c(t) = c_n\) for \(t \in [t_n, t_{n+1}]\), the system (9) is a system of homogeneous linear equations with constant coefficients and its exact solutions can be explicitly written. Assuming that \(M\) is diagonalizable, the resulting computational scheme can be written as:

$$u_{n+1} = T(V_n, c_n)u_n,$$

(10)

where

$$T(V, c) = S(V, c) \exp (A(V, c)k) S(V, c)^{-1}$$

(11)

and \(S\) and \(A\) are respectively the matrix of eigenvectors and the diagonal matrix of eigenvalues of \(M\). This matrix Rush-Larsen scheme is also implemented in the device rushlarsen mentioned earlier.

Finding eigenvalues and eigenvectors for the diagonalisation and computing exponentials are relatively time consuming operations. For that reason the rushlarsen device does a tabulation. That is, for the case when the coefficients \(A, B\) depend on \(V\) and matrices \(T\) depend only on one control variable, e.g. \(V\) (are “univariate”), their values are precomputed for a sufficiently fine grid of the control variable at the start time.

If matrix \(M\) depends on multiple control variables, e.g. both \(V\) and \(c\) (are “multivariate”), it can sometimes be presented as a sum of univariate matrices. Then rushlarsen uses Lie operator splitting and integrates each of the subsystems associated with each of the univariate matrices using the tabulated “matrix Rush-Larsen” separately. For some kinetics models, \(M\) can be presented as a sum of one or more univariate matrices and a remainder, which is multivariate but uniformly small. In that case the subsystem associated with the small remainder is done using the forward Euler method. Finally, if any such decomposition is not possible, “matrix Rush-Larsen” step still can be done, just without tabulation, but by doing the diagonalization “on the fly”, i.e. at run time rather than start time. Although such computation is relatively costly, the benefit of larger time step may still outweigh the expense. The possibility of tabulating multivariate function theoretically exists but is not considered in BeatBox due to resource implications.

The diagonalization is done using appropriate routines from GSL [19]; the relevant subset of GSL is included in BeatBox distribution for portability and the users convenience.

4.3 Monodomain diffusion and boundary conditions

We focus here on the device diff which is the main device implementing the diffusion term in the monodomain diffusion, which mathematically can be described as:

$$\mathcal{L}u = \sum_{j,k=1}^3 \frac{\partial}{\partial x_j} \left( D_{jk} (\vec{r}) \frac{\partial u}{\partial x_j} \right)$$

(12)

with the naturally associated non-flux boundary conditions,

$$\sum_{j,k=1}^3 n_k D_{jk}(\vec{r}) \frac{\partial u}{\partial x_j} = 0$$

(13)

where \(\vec{u} = (u_p)\) is the normal to the boundary \(\Gamma\) of the domain \(\mathcal{D}\), i.e. excitable tissue. Currently this is implemented for the transversely isotrop case, i.e. when the diffusion tensor \(D = (C_{\alpha\beta})^{-1} \delta_{\alpha\beta} = (D_{jk})\) has only two different eigenvalues: the bigger, simple eigenvalue \(D_{11}\) corresponding to the direction along the tissue fibers, and the smaller, double eigenvalue \(D_{12}\), corresponding to the directions across the fibres, as this is the most popular case in modelling anisotropic cardiac tissue (the modification for the general orthotropic case is straightforward though). In this case,

$$D_{jk} = D_{11} \delta_{jk} + (D_{12} - D_{11}) \delta_{jk} f_j f_k,$$

(14)

where \(\vec{f} = (f_k)\) is the unit vector of the fiber direction. The simple finite-difference approximation of (12,13) in diff device is along the lines described, e.g. in [14]. In detail, we have

$$(\mathcal{L}u)_p = \sum_{q \in \{0, \pm 1\}^3} W^p_q u_{p+q},$$

(15)

where \(\vec{r}p\) is the 3D index of a grid node with position \(\vec{r}_p\), \(u_p = u(\vec{r}_p)\) is the value of the field \(u\) at the grid node \(p\), \((\mathcal{L}u)_p\) is the value of the diffusion operator approximation at that point, \(q \in \{0, \pm 1\}^3\) is its increment, and the weights \(W^p_q\) are defined by the following expressions:

$$W^p_q = \overline{W^p_q} - \overline{W^p_q},$$

(16)
The timestepping is by using a forward Euler scheme with error, however these require extra information about the problem (20)–(24). Slope lines are with slopes characterized by two norms,

\[ \|\varepsilon\|_{L^\infty} = \max_{t \in [0,T]} \|\varepsilon(\vec{r},t)\|, \]
\[ \|\varepsilon\|_{L^2} = \left( \frac{1}{T \mu(D)} \int_0^T \int_D \|\varepsilon(\vec{r},t)\|^2 \, d^2\vec{r} \, dt \right)^{1/2}. \]

where \( \mu(D) \) stands for the area of \( D \), and all the integrals are calculated by the trapezoidal rule. Each \( h \) has four corresponding points on each graph, corresponding to four simulations, with different position of the centre of the circle \((x_0,y_0)\) with respect to the grid \((\vec{r}_h)\). We can see that the convergence is worse than \( h^2 \), but better than \( h^1 \). The \( L^2 \) norm of the error converges faster than \( L^\infty \) norm, which is an indication that the main source of error is localized — this is, of course, to be expected, as the boundary conditions, in a sense, approximate the curvilinear boundary \( \Gamma \) with one consisting of pieces of straight lines parallel to the \( x \) and \( y \) axes, thus typically making an error \( O(h) \). We stress that in cases where the realistic tissue geometry is available as a set of points with the same resolution as the computational grid, the knowledge of any curvilinear boundary is in any case unavailable, so any loss of accuracy associated with it, or, equivalently, any notional gain of accuracy that would be associated with using a curvilinear boundary instead, would be purely theoretical.

### 4.4 Bidomain diffusion

Computations using the bidomain tissue description (1) differ primarily by the presence of the equation:

\[ \nabla \cdot (\sigma_1 + \sigma_0) \nabla \Phi_i = \nabla \cdot \sigma_0 \nabla V - I_{ext}, \]

which is elliptic with respect to both \( \Phi_i \) and \( V \). We have implemented a solver for elliptic equations in the elliptic device. This uses Full Multigrid iterations with vertex-centered restriction/prolongation operators with bi/trilinear interpolation, and a (multicoloured) Gauss-Seidel or a Jacobi smoother.

The linear system to which the solver applies naturally occurs through discretization of the diffusion operator in the same way as described in the previous subsection. The solution to the problem (28) with non-flux boundary conditions

\[ W^P_{ij} = \psi_{i+q,j} - \psi_{i+q,j-1} \]

\[ \widetilde{W}^P_{ij} = \frac{\psi_{i+q,j} - \psi_{i+q,j-1}}{4h^2} \left( 3 \sum_{j=1}^{j=3} \left( D_{jk}^{P+q} - D_{jk}^{P-q} \right), \right. \]

\[ W^P(0,0,0) = - \sum_{\mathbf{q} \neq (0,0,0)} W^P_{ij}, \]

where \( j, k \in \{1, 2, 3\}, \psi_p \) is the grid indicator function of the domain \( D \), that is, \( \psi_p = 1 \) if \( \vec{r}_p \in D \) and \( \psi_p = 0 \) otherwise, \( D_{jk}^p = D_{jk}(\vec{r}_p), \mathbf{q}_1 = (1, 0, 0), \mathbf{q}_2 = (0, 1, 0), \mathbf{q}_3 = (0, 0, 1). \)
is not unique, so the solver will require an extra "pinning" condition, to select a unique solution out of infinitely many: the solution at a given point should have a given value. For solving the bidomain model (1) using operator splitting, the elliptic device can be used to solve the elliptic equation with respect to $\Phi$, leaving the parabolic diffusion equation for time-stepping $V$ using the diff device and time-stepping $V$ and $g$ according to the reaction kinetics via an ODE solver, such as euler device.

We illustrate this computation scheme on an example of a bidomain problem with an exact solution. We consider a bidomain system (1) with a one-component “cell model”, $\dim g = 0$, corresponding to the Zeldovich-Frank-Kamenetsky [42] also known as Nagumo equation [28] and Schrödinger model [38]:

$$\frac{\partial V}{\partial t} = V(V - \alpha)(1 - V) + \left( D_I \frac{\partial^2}{\partial x_1^2} + D_I \frac{\partial^2}{\partial x_2^2} \right) \Phi_i,$$

$$\left[D_D + D_I \right] \frac{\partial^2}{\partial x_1^2} \Phi_i = \left[D_D \frac{\partial^2}{\partial x_1^2} + D_I \frac{\partial^2}{\partial x_2^2} \right] V.$$

(29)

If posed on the whole plane, $(x, y) \in \mathbb{R}^2$, this system has a family of exact solutions in the form of plane waves,

$$V^* = \left\{ 1 + \exp \left[(x \cos \theta + y \sin \theta - s - ct) / \sqrt{2D_e} \right] \right\}^{-1},$$

$$\Phi_i^* = KV^*,$$

(31)

(32)

where the angle of the wave propagation, $\theta$, and its initial phase, $s$, are arbitrary constants, and the other parameters of the solution are defined by $c = \sqrt{2D_e} (\frac{1}{2} - \alpha)$, $D_D = D_D D_I / (D_I + D_D)$, $K = D_I / (D_I + D_D)$, $D_D = D_D \cos^2 \theta + D_I \sin^2 \theta$, $D_E = D_D \sin^2 \theta + D_I \cos^2 \theta$.

For testing BeatBox as a bidomain solver, we consider the problem for the system (29,30) in a square domain of size $L$, $D = [0, L]^2$, for a time interval $t \in [0, T]$, with the initial and (non-homogeneous) Dirichlet boundary conditions set in terms of (31,32) as

$$V(x, y, 0) = V^*, \quad \Phi_i(x, y, 0) = \Phi_i^*, \quad (x, y) \in \mathcal{D};$$

$$V(x, y, t) = V^*, \quad \Phi_i(x, y, t) = \Phi_i^*, \quad (x, y) \in \mathcal{\Gamma}, t \in (0, T),$$

$$\mathcal{\Gamma} = \partial \mathcal{D} = \{0, L\} \times \{0, L\} \cup \{0, L\} \times \{0, L\}.$$

To implement solution of this problem in a BeatBox script, we split the equations into fragments, determining the substeps in the calculations:

$$\text{diff}(1): \quad S_1 = \nabla D^e \nabla V,$$

$$\text{elliptic}: \quad \nabla \left(D_I \nabla \Phi_i \right) = S_1,$$

$$\text{diff}(2): \quad S_2 = \nabla D^e \nabla \Phi_i,$$

$$\text{euler}: \quad V_t = V(V - \alpha)(1 - V) + S_2.$$

The resulting fragment of BeatBox looks like this:

```plaintext
// source term in the elliptic equation
// solving the elliptic equation
def str domain x0=xil x1=xir y0=yil y1=yir;
diff [domain] v0=[u] v1=[s]
  Dpar=Dix Dtrans=Dy hx=hx;
  // source term in the parabolic equation
euler [domain] v0=[u] v1=[u]
  ode=zfkt ht=ht par={alpha=alpha 1u=0[s]};
```

In this fragment, the first diff device computes the right-hand side of the elliptic equation, $S_1 = \nabla \cdot D^e \nabla V$ and deposits the result $S_1$ into the layer [s]; then elliptic solves the elliptic equation for $\Phi$, using the provided fine-tuning algorithm parameters and puts the result into the layer [p].

The second diff device computes $\nabla \cdot D^e \nabla \Phi_i$ and puts the result $S_2$ into the layer [s] (which is therefore “recycled”), and the euler device does the time step of the cell model. The interior of the domain $D$ is mapped to the subgrid [domain] with the grid $x$-coordinate from xil to xir and $y$-coordinate from yil to yir. The non-homogeneous, non-stationary Dirichlet boundary conditions (34) were implemented by a k_func device, computing the boundary values for $V$ and $\Phi$ for the grid nodes surrounding this subgrid [domain], i.e., those with grid coordinates xil-1,xir+1,yil-1,yir+1 (this part of the script is not shown).

The accuracy of the computational scheme is illustrated in fig. 4. We take $L = 10$, $T = 40$, $\alpha = 0.13$, $D_e = 2$, $D_I = 0.2$, $D_D = 8$, $D_I = 2$ and $s = -5$. The time step $k = 3h^2 / (16D_e)$ is varied together with the space step with the coefficient $3/ (16D_e)$ chosen from the considerations of numerical stability [14], resulting in quadratic convergence of the algorithm, as should be expected.

5. COMPLEX GEOMETRIES AND DOMAIN PARTITIONING

Sharing work between processes in the MPI version of BeatBox is done by splitting the computational grid into subdomains, such that computation in each subdomain is done by a process. The method of domain decomposition is illustrated in fig. 5, for a 2D domain. Each of the x, y and z dimensions of the grid is separated by a certain number of equal subintervals (approximately equal, when the grid size is not divisible by the number of subintervals). The number of subintervals in different dimensions do not have to be the same. In the example shown in fig. 5, the x and y dimensions are split to have 3 subintervals each; the z dimension is not split. The grid nodes, in which computations are done, are represented in fig. 5 by solid circles (“bullets”).

The continuity of computations across subdomains necessary for devices involving the diffusion operator is achieved by using message passing with exchange buffers. The depth of the exchange buffer in each direction is one grid point. This imposes a limitation on the stencils that can be used by diffusion-like devices, such as a 9-point stencil for 2D and up to a 27-point stencil for 3D. In fig. 5, the hollow circles represent the fictitious grid nodes which are images of corresponding nodes from neighbouring subdomains, and the dashed lines designate the whole buffers, including the nodes to be sent and nodes to be received. The buffer exchange should be effected twice (forwards and backwards) for each buffer size, i.e. four exchanges in 2D simulations and six exchanges in 3D simulations. If the buffer exchanges are done in the correct order, then this will ensure correct
In bidomain computations, the buffer exchanges can be done either before each iteration or more seldom; in the latter case the result is different from the sequential run, but inasmuch as both MPI and sequential results are close to the actual solution, the difference between the two should be negligible. Similar consideration applies to the use of the Gauss-Seidel smoother, which of course will also give negligible results if applied in each subdomain separately.

When working with complex (non-cuboidal) domains, the BeatBox approach is to inscribe the domain into the smallest cuboid, and then proceed as before, with the difference that computations are only done at those grid nodes that belong to that domain, and the one outside domain remain idle. This is also illustrated schematically in fig. 5, where the boundary of the domain is drawn by a closed bold solid black line. This creates a challenge to the performance: with high-degree parallelization and complicated geometry of the domain, the load imbalance between processes can become significant; in particular, a large number of partitions will contain no points of the domain (in fig. 5, there is one such partition, in the bottom left corner). This problem is well known and there are efficient tools for solving them for structured as well as unstructured grids, see e.g. [24, 21]. In the current version of BeatBox, however, only the crudest optimization method is used: the partitions that are completely idle are not allocated to processes, which considerably limits the expected slow-down because of the uneven load (roughly speaking, at worst twice on average).

6. INPUT/OUTPUT AND VISUALIZATION

Finally, we briefly mention some input/output options currently available in BeatBox. Each of these options is implemented in the corresponding device. This includes:

- Full precision binary input and output of a specified subset grid (devices load and dump respectively);
- Discretized “fixed-point” (1 byte per value) output of three selected layers of a subset of the grid (device ppmout);
- Plain text outputs of a defined subset of the grid or list of expressions involving global variables (devices record and k_print).

Some computational devices also have i/o options. For instance, device k_func, which performs computations by formulas specified in the BeatBox script, can read data from of a plain text file; such data are interpreted as a tabulated univariate vector-function and is often used to create initial conditions by phase-distribution method. Another example is device singz, which finds phase singularities in z-cross-sections of the grid. In addition to assigning the coordinates of the singularity points to global variables, it can also output those to a plain text file and/or visualize. Many devices have an optional debug parameter for printing plain text messages about details of their work.

Regarding run-time visualization, the sequential version of BeatBox has a number of devices for 1D and 2D visualization via X11 protocol if available (devices k_draw, k_plot, k_paint). 3D output typically requires much more tuning to be effective. Theoretically, one possibility is to do the tuning in the interactive mode while the computations are stopped, as it is done e.g. in PZSCROLL, see [2]; this, however, would go against one of the principles of BeatBox, that all details of the run are specified in the input script, so any simulation is reproducible. Instead, currently the visualization is done by post-processing of the output data; most often the data produced by ppmout.

7. PARALLEL SCALING PERFORMANCE

Figure 6 illustrates the computation time taken by the MPI version of BeatBox on ARCHER (UK national supercomputing facility, http://www.archer.ac.uk/) as a function of the number of processes, for three series of test jobs, presenting different challenges from the parallelization viewpoint. In all series, the jobs simulated the monodomain model (2) with Courtemanche et al. 1998 model of human atrial cells [15], with dim \( g = 23 \), and anisotropic diffusion (19-point stencil), but with different geometries. Figure 6(a) is for a cubic grid of \( 300 \times 300 \times 300 \) points. Figure 6(b) is for the rabbit ventricles geometry, described in [17], which is inscribed in a cuboid grid \( 119 \times 109 \times 131 \), containing 470324 tissue points, which is about 27.7% out of the total number of points in the grid. Figure 6(c) is for the human atrial geometry [39], inscribed into a cuboid grid \( 237 \times 271 \times 300 \), containing 1602787 tissue points, about 8.3% of the total. The human atrium geometry tests were used in the crude optimization of partitioning, i.e. the subdomains that do not contain tissue points are not allocated to processes; the rabbit ventricle jobs are done without such optimization. In all job series, the simulation was with an initial condition of a scroll wave with a filament along the y-axis, using the...
Figure 6: Log-log plots: the wall clock time per one time step in the simulation job, vs the number of cores. (a) Full box; (b) Rabbit ventricle geometry, (c) Human atrium geometry.

Figure 7: Generation of a scroll wave out of microscopic re-entries in excitable medium with random, space- and time-dependent distribution of parameters, modelling movement of ischaemic border zone during reperfusion; Beeler-Reuter [3] kinetics [6].

BeatBox has been used to produce simulation results presented in dozens of publications. In this section, we mention a handful of recent and representative studies, illustrating the key features of this software.

We observe that the effect of feed-back control and plain text outputs on the parallel performance is relatively small, and the main slow-down at high parallelization happens due to uneven load of the processes. The parallel scaling is, as expected, best for the full box: without ppmout it is close to ideal for up to 1536 processes, while the curves for the complicated geometries deviate from ideal noticeably. Human atrium geometry is “much thinner” than the rabbit ventricle geometry, and the deviation from ideal is more pronounced in fig. 6(c) than in fig. 6(b). However, the detrimental effect of the uneven load is limited: notice that the “without ppm” curve in fig. 6(c) is almost parallel to the ideal in the interval of 192–1536 cores, and the slow down is only slightly more than by a factor of two. Another significant factor is the bulk output. In the human atrium geometry, such outputs were 10 times more frequent, and their effects is more pronounced overall and starts increasing at smaller parallelizations. Notice that the relative effect of the bulk output is much less in the full box: an obvious explanation is that the ppmout format always outputs the full enclosing grid while computations are only done in the tissue nodes, hence the output/computation ratio is about 12 times bigger for human atrium than in the full box.

We conclude that the parallel performance is in accordance with expectations and satisfactory. The biggest challenge is to simulations with complex and “thin” geometries, and a significant improvement may be achieved by optimizing bulk outputs.

8. EXAMPLES OF USE IN RECENT AND CURRENT RESEARCH

procedure described e.g. in [22]. The output was via ppmout device, which outputs up to three selected layers of the grid, discretized to the 0..255 scale (3D extension of the ppm format of Netpbm, [35]); the corresponding curves in the graphs are marked as “without ppm” and “with ppm” respectively. Such outputs were done once in every 200 timesteps for the full box and human atrium geometries, and once in every 1000 timesteps for the rabbit atrium geometry. The full box and human atrium jobs also did plain text output of the activity at a single point once in every 20 time steps. To test the expenses associated with sophisticated control, the full box and human atrium jobs implemented feedback-controlled stimulation, similar to that implemented in the sample script described in the Appendix. To exclude the effects of the time taken by the start-up operations, we computed the time per step by running jobs identical in all respect except the number of time steps, and then considering the difference. ARCHER has 24 cores per node, and the numbers of processes in the test jobs are power-of-two multiples of 24. The “ideal” lines are drawn based on the result of the “without-ppm” job on 24 processors.

We observe that the effect of feed-back control and plain text outputs on the parallel performance is relatively small, and the main slow-down at high parallelization happens due to uneven load of the processes. The parallel scaling is, as expected, best for the full box: without ppmout it is close to ideal for up to 1536 processes, while the curves for the complicated geometries deviate from ideal noticeably. Human atrium geometry is “much thinner” than the rabbit ventricle geometry, and the deviation from ideal is more pronounced in fig. 6(c) than in fig. 6(b). However, the detrimental effect of the uneven load is limited: notice that the “without ppm” curve in fig. 6(c) is almost parallel to the ideal in the interval of 192–1536 cores, and the slow down is only slightly more than by a factor of two. Another significant factor is the bulk output. In the human atrium geometry, such outputs were 10 times more frequent, and their effects is more pronounced overall and starts increasing at smaller parallelizations. Notice that the relative effect of the bulk output is much less in the full box: an obvious explanation is that the ppmout format always outputs the full enclosing grid while computations are only done in the tissue nodes, hence the output/computation ratio is about 12 times bigger for human atrium than in the full box.

We conclude that the parallel performance is in accordance with expectations and satisfactory. The biggest challenge is to simulations with complex and “thin” geometries, and a significant improvement may be achieved by optimizing bulk outputs.

8. EXAMPLES OF USE IN RECENT AND CURRENT RESEARCH
which modelled arrhythmogenic mechanisms of the boundary layer between ischaemic and normal cardiac tissue, moving due to reperfusion. The model assumed that a certain “excitability” parameter was varying in space and time due to two factors: firstly, space-only random distribution due to properties of individual cells; secondly, deterministic smooth transition between low excitability of the ischaemic tissue and high excitability of the recovered tissue, changing in time due to reperfusion. On top of that, the isotropic diffusion coefficient was also varied along a similar transition between low diffusivity of the ischaemic tissue and high diffusivity of the recovered tissue, of a profile different from, moving synchronously with of the excitability parameter.

Figure 8: Drift of scroll wave along a thickness step, FitzHugh-Nagumo kinetics[11].

Figures 8, 9, 10 illustrate simulations in non-cuboid domains. Figure 8 shows a surface view to a simulation in an artificially defined domain, used to quantitatively test predictions of an asymptotic theory about the drift of a scroll wave in a thin layer due to sharp variations of thickness. This simulation uses two-component FitzHugh-Nagumo kinetics. Shown is the surface view at a selected moment of time, colour coding represents states of the activator variable (red colour component) and inhibitor variable (green colour component) on dark-blue background; the white line shows the trajectory of the tip of the spiral wave seen at the upper surface of the domain for a period of time preceding the moment the presented surface view.

Figure 9: Drift of scroll wave in a realistic human atrium geometry, Courtemanche et al. [15] kinetics, (a) spontaneous, caused purely by the anatomy features [22]; (b) resonant, caused by feedback-controlled electrical stimulation [23].

Figure 9(a) illustrates the corresponding geometry-induced drift in anatomically realistic model of human atrium. Shown are a number of trajectories of tips of spiral waves appearing on the surface of the atrium nearest to the viewer; the yellow background indicates prominent anatomical features (the pectinate muscles and the terminal crest). To make the visualization clearer, the trajectories are represented by lines connecting tip positions separated by exactly one period of rotation (“stroboscopic view”); shown are several trajectories starting at different initial positions and made within equal time intervals.

Figure 10: A scroll wave in human foetal heart anatomy, FitzHugh-Nagumo kinetics [34, 12].

Figure 10 shows a volume view of a scroll wave in a human foetal heart geometry [34, 12]. Shown is the surface $u = \text{const}$, where $u$ is the activator variable (FitzHugh-Nagumo model was used in this case for illustrative purposes), pink, the boundary of the computational domain, i.e. surface of the ventricular geometry, is shown in blue, both surfaces are semi-transparent. The yellow line traversing the ventricular wall is the filament of the scroll, defined as the intersection of the same $u = \text{const}$ surface shown in the picture with a $v = \text{const}$ surface.

Finally, fig. 9(b) illustrates further the capability of BeatBox for complex simulation protocols. This panel displays results of simulations in the same model as those shown in fig. 9(a), but now the initial position for the scroll wave is chosen far from any sharp features so the anatomy-induced drift is not pronounced, and instead, the scroll wave is subject to low-voltage pulses of external electric field, $I_{\text{eff}} = I_{\text{eff}}(t)$. The delivery of the pulses is controlled by a protocol similar to that illustrated by the sample script fig. 11, namely:

- There is a “registration electrode”, at a point at the surface of the atrium that is the most distant from the viewer, position of which is indicated by the purple cone.
- The signal from the registration electrode is monitored for the moment of arrival of an excitation wave, defined as the moment when the transmembrane voltage crosses a certain threshold value upwards.
- From the moment of the front arrival to the registration electrode, a certain waiting interval (delay) is observed.
9. CONCLUSIONS

The leading idea underlying BeatBox development is robustness, portability, flexibility and user-friendliness in the first place, connected with efficiency as an important but secondary consideration. In the present form, BeatBox can be exploited in sequential and parallel (MPI) modes, with run-time and/or post-processing visualization, on any unix-like platform from laptops to supercomputing facilities. The modular structure of BeatBox effectively decouples the user interface, which at present is a scripting language used to construct the ring of devices, from the implementation of the computationally intensive stages in individual devices. The current computational capabilities have emerged from the needs of the BeatBox developers in their research so far, and can be expanded in accordance with the needs of wider usership without changing the backbone ideology.

As far as MPI features are concerned, the straightforward approach to parallelization via domain decomposition, yields acceptable results for small to medium scale, becoming inefficient for number of threads beyond about 1000 for the examples tried. The limiting factors seem to be the uneven load of the parallel threads for “thin” complex geometries of the computational domains, and output, which determines possible direction of further development. The uneven load can be addressed by a more careful fine-tuning of domain decomposition to specifics of particular domain geometry, which to some extent may be achieved without violating the main principles of the domain decomposition, by allowing uneven partitioning along the coordinate axes. The slow down in cases of extensive output is a problem which is not specific for BeatBox; however, some improvement in some cases may be achieved by making any input-output operations exclusive to one or more designated threads specializing on this and relieved from computational load as such.

As the current BeatBox solvers use finite difference, regular grid ideology, incorporation of DT-MRI regular cartesian grid anatomy models into BeatBox simulations is straightforward, fig. 9 and 10, without a meshalizer step required for finite element/finite volume solvers. However, architecturally there is no principal problems in extending BeatBox functionality to the finite element approximation as long as regular mesh of finite elements is used that can be mapped to a rectangular array. Extension to irregular meshes would require more substantive changes, however the main idea of the ring of devices may be useful there as well.

10. ACKNOWLEDGMENTS

The early stages of development of BeatBox and its predecessors benefited from contributions from A.V. Karpov and R. McFarlane. The recent development of BeatBox was supported by EPSRC grant EP/I029664 (UK). VNB also gratefully acknowledges the current financial support of the EPSRC via grant EP/N014391/1 (UK).

11. REFERENCES

[1] U. Ayachit. The ParaView Guide: A Parallel Visualization Application Paperback. Kitware, USA, 2015.
[2] D. Barkley and M. Dowle. EZ-SCROLL: A code for simulating scroll waves. http://www.warwick.ac.uk/˜masax/Software/ez_software.html, 2007.
[3] G. W. Beeler and H. Reuter. Reconstruction of action potential of ventricular myocardial fibers. J. Physiol. — London, 268(1):177–210, 1977.
[4] V. N. Biktashev. A three-dimensional autowave turbulence. Int. J. of Bifurcation and Chaos, 8(4):677–684, 1998.
[5] V. N. Biktashev. Dissipation of the excitation wavefronts. Phys. Rev. Lett., 89(16):168102, 2002.
[6] V. N. Biktashev, I. V. Biktasheva, and N. A. Sarvazyan. Evolution of spiral and scroll waves of excitation in a mathematical model of ischaemic border zone. PLoS ONE, 6(9):e24388, 2011.
[7] V. N. Biktashev and A. V. Holden. Design principles of a low-voltage cardiac defibrillator based on the effect of feed-back resonant drift. J. Theor. Biol., 169(2):101–113, 1994.
[8] V. N. Biktashev, A. V. Holden, and H. Zhang. Tension of organizing filaments of scroll waves. Phil. Trans. Roy. Soc. Lond. ser. A, 347:611–630, 1994.
[9] I. V. Biktasheva and V. N. Biktashev. Wave-particle dualism of spiral waves dynamics. Phys. Rev. E, 67:026221, 2003.
[10] I. V. Biktasheva, V. N. Biktashev, W. Dawes, A. Holden, R. Saumarez, and A.M.Savill. Dissipation of the excitation front as a mechanism of self-terminating arrhythmias. Int. J. of Bifurcation and Chaos, 13(12):3645–3655, 2003.
[11] I. V. Biktashev, H. Dierckx, and V. N. Biktashev. Drift of scroll waves in thin layers caused by thickness features: asymptotic theory and numerical simulations. Phys. Rev. Lett., 114(6):068302, 2015.
[12] F. Cai, E. Pervolaraki, A. V. Holden, and I. V. Biktasheva. paper in preparation, 2016.
[13] C. E. Clancy and Y. Rudy. Linking a genetic defect to its cellular phenotype in a cardiac arrhythmia. Nature, 400:566–569, 1999.
[14] R. H. Clayton and A. V. Panfilov. A guide to modelling cardiac electrical activity in anatomically detailed ventricles. Prog. Biophys. Mol. Biol., 96:19–43, 2008.
[15] M. Courtmemanche, R. J. Ramirez, and S. Nattel. Ionic mechanisms underlying human atrial action potential properties: insights from a mathematical model. Am. J. Physiol., 275:H301–H321, 1998.
[16] M. Dowle, R. M. Mantel, and D. Barkley. Fast simulations of waves in three-dimensional excitable media, International Journal of Bifurcation and Chaos, 7(11):2529–2545, 1997.

• On expiration of the delay interval, a pulse of \( I_{eff}(t) \) of a certain duration and certain amplitude is applied.

In fig. 9(b) we see three different trajectories starting at the same point: they differ in the value of the delay interval between registration of the front arrival and delivery of the electric pulse.

The visualization in all cases was done by post-processing of the simulation data. For fig. 7, we used Iris Explorer [33]. Both panels of fig. 9 were generated with ParaView [1]. Figures 8 and 10 were produced by an in-house visualizer, based on the graphical part of Barkley and Dowle’s EZSCROLL [16, 2], which is in turn based on the Marching Cubes algorithm [25, 29].
APPENDIX

A sample BeatBox script

The complete listing of a simple but non-trivial example of a BeatBox script, which is illustrated by fig. 2, is provided in fig. 11. The main features of the syntax may be seen from the script itself which is intended to be self-explanatory, but nevertheless:

- Comments in the script can be in C style, within /*...*/ or in C++ style, between // and the end of line.
- Expression <fhn.par> means inclusion of another text file with name fhn.par, as part of the script, similar to #include <fhn.par> in C.
- The script is a sequence of sentences, each concluding with a semicolon, ;.
- Sentences starting with keyword def declare global variables, with optional initial values.
- Global variables of type str are string macros, expansion of a string macro declared as def str foo bar; is done using syntax [foo] which will produce bar in place of expansion.
- The script finishes with a sentence end;
<fhn.par> // model pars are read in from file fhn.par
def str b 3; // spatially dependent parameter in 4th layer
def real grad [1]; // its gradient is 1st command-line parameter

// Integer and real stimulation parameters
def int xr 100; def int yr 100; def int zr 100; // reg electrode position, in space steps
def int dr 5; // reg electrode size, in space steps
def real Amp 3.0; // pulse amplitude
def real Dur 0.1; // pulse duration
def real Del 6.0; // pulse delay
def real Tstart 100.0; // when to switch on the feedback loop

state geometry=ffr.bbg anisotropy=1 // the file contains tissue geometry and fibres
vmax=4; // 2 dyn vars + diffusive current + parameter

def real T; def real begin; def real out; def real end; // real vars control works of some devices
k_func name=timing nowhere=1 pgm={ // this function operates only global variables
  T = t*ht; // t is integer time counter; T is real time
  begin = eq(t,0); // 1.0 at the very beginning, otherwise 0.0
  out = eq(mod(t,100),0); // 1.0 every 100 timesteps, otherwise 0.0
  end = ge(T,100.0)); // 1.0 after 100 ms, otherwise 0.0

// This function operates at every point but only in the very beginning
k_func when=begin pgm={$
  u[u]=ifle0(x-25,1.7,-1.7); u[v]=ifle0(y-25,0.7,-0.7) // Cross-field initial conditions
  u[b]=bet+grad*(z-0.5*zmax)}; // vertical gradient of parameter

// The feedback loop
def real signal; def real front; def real Tfront;
reduce operation=max result=signal v0=[u] v1=[u] // signal=max of voltage field within given volume
  x0=xr xr1=xr+dr-1 y0=yr yr1=yr+dr-1 z0=zr zr1=zr+dr-1; // the values are arithmetic expressions
  k_poincare nowhere=1 sign=1 // remember T when signal crossed value umid upward
  pgm={
    front=signal-umid; Tfront=T
  }

k_func when=force pgm={$
  signal=ht*Amp*ge(T,Tstart)*ge(T,Tfront+Del)*le(T,Tfront+Del+Dur)}; // force lasts Dur ms starting Del ms after crossing

// The computation
diff v0=[u] v1=[v] Dpar=D Dtrans=D/4 hx=hx; // anisotropic diffusion
k_func when=force pgm={$[u]=u[i]+force} // this applies everywhere, only when force is nonzero
euler v0=[u] v1=[v] ht=ht ode=fhncub // cubic FitzHugh-Nagumo kinetics
  par={eps=eps bet=@[b] gam=gam Iu=@[i]}; // varied beta and current as calculated before

// Output
ppmout when=out file=0/04d.ppm mode=w // every 100 timesteps:
  r=[u] r0=umin r1=umax // value-discretized
  g=[v] g0=min g1=vmax // output for subsequent
  b=[i] b0=0 b1=255; // visualization
k_print when=always file=stdout list={T; force; signal}; // to monitor work of the feedback loop
record when=end file=0.rec when=end v0=0 v1=1; // ascii dump of all field values in the end of run

stop when=end;
end;

Figure 11: A sample BeatBox script.
The particular devices used in the script have the following function:

- **k_func** computes, depending on the current value of the loop counter \( t \), the “flag” global variables that control which of the other devices will or will not work at the current loop of the ring.

- **k_func**, another instance of the same device as before, but now it computes not the global variables, but the values of the field variables at every point of the grid, according to a given formula. This device works only once, at the very first loop, and its function is to produce initial conditions for the computations.

- **reduce** is a device that computes the value for a global variable based on current state of one or more of the fields represented in the computational grid. In the current example, this device emulates the work of a registration electrode, which measures the maximal value of the “transmembrane voltage” in a particular small volume in the grid. This measurement will be used as a signal in a feedback loop that is used to control the electrical excitation using a putative low-voltage defibrillation protocol.

- **k_poincare** is a device that implements the idea of a Poincare cross-section from the dynamical systems theory. In the present case, this device checks whether at the current loop, the signal, measured by the previous **reduce** device, has crossed a given threshold value in the required direction. If that has happened, then a certain flag (a global variable) is “raised” and the time (the value of the counter \( t \)) when this happened is remembered in another global variable.

- **k_func** is yet another instance of this versatile device. This instance again works with global variables: it computes, using a given formula, the value of the variable that defines the defibrillating electric field, depending on the time that has passed since the event registered by the **k_poincare** device.

- **diff** is the first of the devices which does “the actual computations” in the sense that it changes the fields represented in the computational grid. As could be guessed from its name, it computes the diffusion term, i.e. a value of the Laplacian of the field represented by one layer of the computational grid, and records the result in the other layer of the grid.

- **k_func** — this instance of the device works is “local”, i.e. it works on the computational grid: computes the action of the defibrillating electrical field (computed by the previous instance of this device) onto the excitable cells.

- **euler** is a computational device that performs the time step for the dynamic fields represented in the computational grid, with account of the given kinetic model, the values of the diffusion term, and the action of the defibrillating pulse.

- **ppmout** is one of the output devices. It only works once-in-so-many-steps (as explicitly defined in the script) and produces an output to a disk file in ppm format, where each byte represents a value of one element of the grid, from up to three selected layers of the grid, discretized to the 0..255 scale. This format is in fact an image format, could be converted to other popular and less space-consuming formats either by postprocessing or on-the-fly (not done in the current simple script).

- **k_print** is a more straightforward output device: each time it is called (in the current example, at the every step), it adds to a disk file a plain-text record of the values of the global variables involved in the feedback loop controlling the defibrillation stimuli.

- **record** is the last output device: it also outputs into a plain text, but not an explicitly listed set of global variable, but a defined subset of the values in the computational grid. In this example, the devices works only on the last step of the computations, and the output file can be used for initial conditions if continuation of the present simulations is required.

- **stop** is the device whose function is to interrupt the computations and terminate the program. Naturally this device must be present in the ring unless it is intended that the program run is to be interrupted by the operator. In the presented example, the device works simply when a certain number of time steps has been done; but it can be controlled by any global variable, which may be computed by a **k_func** device depending on how the simulations progresses, according to arbitrary criteria as defined by the user.