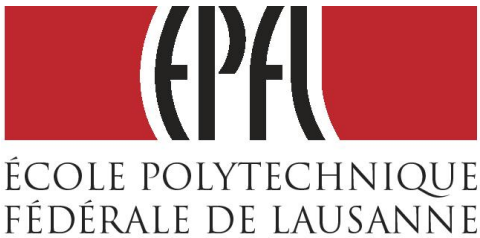


Semester Project

An adaptive Rosenbrock method with applications in electrophysiology

Author:
Giacomo Rosilho de Souza
N° Sciper:
193289

Supervisor:
Simone Rossi
Professor:
Ricardo Ruiz-Baier

Academic year : 2012/2013

Spring Semester

Master Program

Date of submission : June 6, 2013

Contents

1 Introduction

The aim of this project is to implement the Rosenbrock method ROS3P in the C++ library LifeV for the solution of systems of ordinary differential equations arising in electrophysiology.

LifeV is a C++ finite element library which has applications in several domains such as fluid mechanics, solid mechanics, heat transfer, fluid structure interaction, electrocardiology and transport in porous media. In the domain of electrophysiology LifeV implements cardiac cell models which can require the resolution of stiff reaction-diffusion equations. To solve this system the use of implicit methods is required. We focus in particular on one step metods.

Implicit methods need the solution of a non linear system at each time step. The Rosenbrock methods are linearly implicit Ruge-Kutta methods and just requires the solution of s linear systems per time step, where s is the number of stages. Most of the implicit Runge Kutta methods suffer from order reduction when applied to parabolic problems. For this reason we have chosen the third order Rosenbrock method ROS3P, which satisfies additional order conditions to keep the same convergence order even when applied to non linear parabolic problems.

In the first part of this report we derive the Rosenbrock methods from Runge-Kutta methods, explaining time adaptivity strategy and finally giving the algorithm of our implementaion in LifeV. After we present the cardiac reaction-diffusion models, deriving the monodomain equations from the bidomain equations. In the following we explain the strategy that we adopt to solve these models and how we discretize them. Finally we show some numerical experiments done with our approach and compare our results with the ones given by other methods.

2 ROS3P

In this section we will first give an introduction to Runge-Kutta methods, which are an important class of methods for the integration of differential systems. We focus on the Rosenbrock methods, in particular on the third order Rosenbrock method ROS3P. For more details we refer to [4] and [5].

We have chosen a one step method since it requires the storage of only one step and the adaptive strategy is simpler. Adaptive multistep methods may need to reconstruct some local history of the solution and consequently adaptivity is computationally costly.

Introduction to Runge-Kutta methods

Let us consider the Cauchy problem

$$\begin{cases} y'(t) = f(t, y(t)) \quad t > t_n \\ y(t_n) = y_n, \end{cases} \tag{1}$$

where $y : \mathbb{R} \to \mathbb{R}^d$ and $f : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$, for which we wish to numerically approximate the solution.

Definition 2.1. Let b_i and a_{ij} $(i, j = 1, ..., s)$ be real numbers and let $c_i = \sum_{j=1}^{s} a_{ij}$. An s stage Runge-Kutta method is given by

$$k_i = f(t_n + c_i h, y_n + h \sum_{j=1}^{s} a_{ij} k_j) \qquad i = 1, ..., s,$$

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i.$$

Notation. We will denote $A = (a_{ij})$, $b = (b_i)$, $c = (c_i)$ and following the work of Butcher we display the coefficients of a Runge-Kutta method as in Table 1, which is called *Butcher's tableau.*

c_1	a_{11}	$\cdots$	a_{1s}
$\vdots$	$\vdots$		$\vdots$
c_s	a_{s1}	$\cdots$	a_{ss}
	b_1	$\cdots$	b_s

Table 1: Butcher's tableau.

Definition 2.2. A Runge-Kutta method has order p if for all sufficiently regular problems (1) the local error satisfies

$$y_{n+1} - y(t_n + h) = \mathcal{O}(h^{p+1}) \qquad \text{as } h \to 0,$$

where $y_n = y(t_n)$.

Definition 2.3. A Runge-Kutta method is called explicit if $a_{ij} = 0$ for $j \geq i$. It is called implicit otherwise.

The explicit Runge-Kutta methods are simple to implement and computationally cheap. On the other hand if the problem is stiff explicit methods can lead to a prohibitive condition on the time step h. Implicit methods try to relax this condition but they require a solution of a non linear system at each time step. In particular implicit Runge-Kutta methods require the resolution of the non linear system

$$\begin{pmatrix} k_1 \\ \vdots \\ k_s \end{pmatrix} = \begin{pmatrix} f(t_n + c_1 h, y_n + h\sum_{j=1}^{s} a_{1j}k_j) \\ \vdots \\ f(t_n + c_s h, y_n + h\sum_{j=1}^{s} a_{sj}k_j) \end{pmatrix} \tag{2}$$

that is usually solved with the Newton method. Leading to a sequence of $ds \times ds$ linear systems to be solved at each time step.

Rosenbrock methods

Rosenbrock methods are a class of Runge-Kutta implicit methods that avoids the iterative resolution of the Newton scheme. They replace the $ds \times ds$ non linear system by a sequence of s linear systems of size $d \times d$.

Definition 2.4. An implicit Runge-Kutta method is called diagonally implicit (or semi-implicit) if $a_{ij} = 0$ for $j > i$.

Consider a diagonally implicit Runge-Kutta method

$$k_i = f(t_n + h\sum_{j=1}^{i-1} a_{ij} + ha_{ii}, y_n + h\sum_{j=1}^{i-1} a_{ij}k_j + ha_{ii}k_i) \qquad i = 1, ..., s, \tag{3}$$

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i$$

applied to the Cauchy problem (1). Linearizing equation (3) around $(t_n + h\sum_{j=1}^{i-1} a_{ij}, y_n + h\sum_{j=1}^{i-1} a_{ij}k_j)$ yelds

$$k_i = f(d_i, g_i) + ha_{ii}\frac{\partial f}{\partial t}(d_i, g_i) + ha_{ii}\frac{\partial f}{\partial y}(d_i, g_i)k_i \qquad i = 1, ..., s,$$

$$d_i = t_n + h\sum_{j=1}^{i-1} a_{ij},$$

$$g_i = y_n + h\sum_{j=1}^{i-1} a_{ij}k_j.$$

A great computational advantage is given substituting the jacobian $\frac{\partial f}{\partial y}(d_i, g_i)$ with the approximation $J = \frac{\partial f}{\partial y}(t_n, y_n)$. Thanks to that we need to evaluate

the jacobian just once per time step. To gain more freedom in the computation of coefficient k_i we consider a linear combination of the terms k_j for $1 \leq j \leq i$, obtaining the Rosenbrock methods.

Definition 2.5. An s-stage Rosenbrock method is given by

$$\begin{aligned} k_i &= f(t_n + \alpha_i h, y_n + h \sum_{j=1}^{i-1} \alpha_{ij} k_j) \\ &\quad + h\gamma_i \frac{\partial f}{\partial t}(t_n, y_n) + hJ \sum_{j=1}^{i} \gamma_{ij} k_j \qquad i = 1, ..., s, \\ y_{n+1} &= y_n + h \sum_{i=1}^{s} b_i k_i, \end{aligned} \tag{4}$$

where α_{ij}, γ_{ij} and b_i are the method's coefficients, $J = \frac{\partial f}{\partial y}(t_n, y_n)$, $\alpha_i = \sum_{j=1}^{i-1} \alpha_{ij}$ and $\gamma_i = \sum_{j=1}^{i} \gamma_{ij}$.

Observe that at each stage we must solve a linear system, with unknown k_i and matrix $I - h\gamma_{ii}J$. Another computational advantage is given by methods for which $\gamma_{11} = ... = \gamma_{ss} := \gamma$, where the linear system's matrix is the same for all the stages and we can perform only one LU decomposition per step.

Implementation

In addition to the resolution of the linear system the implementation of a Rosenbrock method as it is given in 2.5 requires a matrix vector multiplication $J \sum_{j=1}^{i-1} \gamma_{ij} k_j$. The latter can be avoided with the change of variable

$$u_i = \sum_{j=1}^{i} \gamma_{ij} k_j \qquad i = 1, ..., s.$$

Let us see how to write the Rosenbrock method with this change of variable. We denote by $K = (k_1 \cdots k_s)$ and $U = (u_1 \cdots u_s)$ the matrices having respectively the stages k_i and u_i as columns. Since $\Gamma = (\gamma_{ij})$ is a lower triangular matrix we have

$$U = K\Gamma^{\top}.$$

Calling $(\Gamma^{-1})_{ij} =: \overline{\gamma}_{ij}$ the coefficients of Γ^{-1} and setting $C = (c_{ij})$ as

$$C = -\begin{pmatrix} 0 & \overline{\gamma}_{12} & \cdots & \overline{\gamma}_{1s} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \overline{\gamma}_{s-1,s} \\ 0 & \cdots & \cdots & 0 \end{pmatrix} = \operatorname{diag}(\gamma_{11}^{-1} \cdots \gamma_{ss}^{-1}) - \Gamma^{-1}$$

we observe that

$$
\begin{aligned}
k_i &= \left(U\Gamma^{-\top}\right)_i = (u_1 \cdots u_s) \begin{pmatrix} \overline{\gamma}_{i1} \\ \vdots \\ \overline{\gamma}_{is} \end{pmatrix} = (u_1 \cdots u_i) \begin{pmatrix} \overline{\gamma}_{i1} \\ \vdots \\ \overline{\gamma}_{ii} \end{pmatrix} \\
&= \overline{\gamma}_{ii} u_i + (u_1 \cdots u_{i-1}) \begin{pmatrix} \overline{\gamma}_{i1} \\ \vdots \\ \overline{\gamma}_{i,i-1} \end{pmatrix} = \overline{\gamma}_{ii} u_i - (u_1 \cdots u_{i-1}) \begin{pmatrix} c_{i1} \\ \vdots \\ c_{i,i-1} \end{pmatrix} \\
&= \frac{1}{\gamma_{ii}} u_i - \sum_{j=1}^{i-1} c_{ij} u_j. \qquad (5)
\end{aligned}
$$

Denoting $\mathcal{A} = (\alpha_{ij})$ we can set $A = \mathcal{A}\Gamma^{-1}$ and $m = \Gamma^{-\top} b$. Implying

$$
\begin{aligned}
\sum_{j=1}^{i-1} \alpha_{ij} k_j &= \sum_{j=1}^{i-1} \alpha_{ij} \left(U\Gamma^{-\top}\right)_j = \sum_{j=1}^{i-1} \alpha_{ij} \sum_{k=1}^{s} \overline{\gamma}_{jk} u_k \\
&= \sum_{j=1}^{s} \sum_{k=1}^{s} \alpha_{ij} \overline{\gamma}_{jk} u_k = \sum_{j=1}^{s} \left(\sum_{k=1}^{s} \alpha_{ik} \overline{\gamma}_{kj} \right) u_j \\
&= \sum_{j=1}^{s} a_{ij} u_j = \sum_{j=1}^{i-1} a_{ij} u_j \qquad (6)
\end{aligned}
$$

and

$$
\sum_{i=1}^{s} b_i k_i = \sum_{i=1}^{s} b_i \sum_{k=1}^{s} \overline{\gamma}_{ik} u_k = \sum_{i=1}^{s} \left(\sum_{k=1}^{s} \overline{\gamma}_{ki} b_k \right) u_i = \sum_{i=1}^{s} m_i u_i. \qquad (7)
$$

Putting (5), (6) and (7) in (4) leads

$$
\begin{aligned}
\frac{1}{\gamma_{ii}} u_i - \sum_{j=1}^{i-1} c_{ij} u_j = f(t_n + \alpha_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} u_j) \\
+ h\gamma_i \frac{\partial f}{\partial t}(t_n, y_n) + hJu_i \qquad i = 1, ..., s,
\end{aligned}
$$

which can be written

$$
\begin{aligned}
\left(\frac{1}{\gamma_{ii}} I - hJ \right) u_i = f(t_n + \alpha_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} u_j) \\
+ h\gamma_i \frac{\partial f}{\partial t}(t_n, y_n) + \sum_{j=1}^{i-1} c_{ij} u_j \qquad i = 1, ..., s.
\end{aligned}
$$

Doing a last variable change $U_i = hu_i$ we obtain the formulation of the Rosenbrock method as it is implemented in LifeV

$$\left(\frac{1}{h\gamma_{ii}}I - J\right) U_i = f(t_n + \alpha_i h, y_n + \sum_{j=1}^{i-1} a_{ij}U_j) + h\gamma_i \frac{\partial f}{\partial t}(t_n, y_n) + \sum_{j=1}^{i-1} \frac{c_{ij}}{h} U_j \qquad i = 1, ..., s, \tag{8}$$

$$y_{n+1} = y_n + \sum_{i=1}^{s} m_i U_i.$$

Time step adaptivity

Time adaptivity of the time step is of crucial importance. Stiff equations require small time steps, but an equation can be stiff at certain regions and non stiff in others. If the method is non adaptive the time step must be small enough to guarantee stability in the stiff region, imposing an unnecessary tiny step size in other regions. In Figure 1 we can see an example with a small stiff region with respect to the domain size. In this section we will

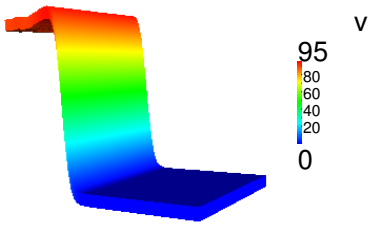


Figure 1: Potential wave modeled with the FitzHugh-Nagumo model.

explain how to control the time step.

Consider a Runge-Kutta method of order p defined by the Butcher's tableau

$$\begin{array}{c|c} c & A \\ \hline & b\top \end{array}.$$

We will denote the approximate solution given by this method with y_n. Suppose now that there exist some coefficients $\hat{b}$ such that the method given by

$$\begin{array}{c|c} c & A \\ \hline & \hat{b}\top \end{array}$$

is of order $\hat{p} < p$ (usually $\hat{p} = p - 1$). At each step we compute the stages k_i, which are the same for both methods, and we find the solutions

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i,$$

$$\hat{y}_{n+1} = y_n + \sum_{i=1}^{s} \hat{b}_i k_i.$$

The local errors, then, satisfies

$$y_{n+1} - y(t_{n+1}) = \mathcal{O}\left(h^{p+1}\right),$$

$$\hat{y}_{n+1} - y(t_{n+1}) = \mathcal{O}\left(h^{\hat{p}+1}\right),$$

when $y_n = y(t_n)$. This implies

$$y_{n+1} - \hat{y}_{n+1} = y_{n+1} - y(t_{n+1}) + y(t_{n+1}) - \hat{y}_{n+1} = \mathcal{O}\left(h^{\hat{p}+1}\right).$$

The latter equation suggests us to look at $\hat{y}_{n+1}$ as an approximation of y_{n+1} of order $\hat{p}$. Since y_{n+1} and $\hat{y}_{n+1}$ follow the same differential equation they will have the same behaviour, thus if the approximation $\hat{y}_{n+1}$ of y_{n+1} get worse then the approximation y_{n+1} of $y(t_{n+1})$ will probably get worse too. We can cheaply measure the error

$$err_{n+1} := \|y_{n+1} - \hat{y}_{n+1}\| = \|\sum_{i=1}^{s}(b_i - \hat{b}_i)k_i\|$$

and detect the deterioration of the approximation y_{n+1} of $y(t_{n+1})$.

Since the local error $y_{n+1} - \hat{y}_{n+1}$ is of order $\hat{p}$ we can write

$$err_{n+1} \approx C_{n+1} h_{n+1}^{\hat{p}+1}.$$

Fixing the error tolerance Tol and supposing $err_{n+1} \leq Tol$, we search for h_{n+2} such that at the next step we have the optimal error $err_{n+2} = Tol$. Supposing that the constants C_{n+1} have a linear behaviour with respect to time we can write $\frac{C_{n+2}}{C_{n+1}} \approx \frac{C_{n+1}}{C_n}$ and obtain

$$\begin{aligned} Tol = err_{n+2} &= C_{n+2} h_{n+2}^{\hat{p}+1} = \frac{C_{n+2}}{C_{n+1}} C_{n+1} h_{n+2}^{\hat{p}+1} = \frac{C_{n+1}}{C_n} C_{n+1} h_{n+2}^{\hat{p}+1} \\ &= \frac{err_{n+1}}{err_n} \left(\frac{h_n}{h_{n+1}}\right)^{\hat{p}+1} \frac{err_{n+1}}{h_{n+1}^{\hat{p}+1}} h_{n+2}^{\hat{p}+1}. \end{aligned}$$

The time step for the next iteration will be given by the relation

$$h_{n+2} = \left(\frac{Tol}{err_{n+1}}\right)^{\frac{1}{\hat{p}+1}} \left(\frac{err_n}{err_{n+1}}\right)^{\frac{1}{\hat{p}+1}} \frac{h_{n+1}}{h_n} h_{n+1}. \tag{9}$$

If $err_{n+1} > Tol$ the $n+1$ step is rejected and we recompute y_{n+1} redefining the time step h_{n+1} as in (9). Practically there are some other safety factors and checks in the implementation, which can be seen in the following algorithm. A coefficient D is introduced to divide the step size by a significant factor when the step is rejected two or more times consecutively. While the factor $S \in (0,1)$ reduces the theoretical optimal increment and fac_{max} is the maximum relative increment of the step size. The tolerance Tol depends on an absolute tolerance Tol_{abs} and a relative tolerance Tol_{rel}.

function ROSENBROCK($f, y_n, t_n, t_F, h_{n+1}, m, \hat{p}$)
$S = 0.95$
$D = 1.5$
$Tol_{abs} = 0.001$
$Tol_{rel} = 0.001$
$fac_{max} = 5$
$fac = 1$
$rejected = false$

for $k = n, n+1, ..., \infty$ **do**
if $t_k == t_F$ **then**
break
end if
$U =$**computeStages**(f, y_k, t_k, h_{k+1})
$err_{k+1} = \|U(m - \hat{m})\|$
if $k > n$ **then**
$Tol = Tol_{abs} + \|y_k\| Tol_{rel}$
$fac = S \left(\frac{Tol}{err_{k+1}}\right)^{\frac{1}{\hat{p}+1}} \left(\frac{err_k}{err_{k+1}}\right)^{\frac{1}{\hat{p}+1}} \frac{h_{k+1}}{h_k}$
if $err_{k+1} > Tol$ **then**
if $rejected$ **then**
$h_{k+1} = h_{k+1}/D$
else
$h_{k+1} = fac\, h_{k+1}$
$rejected = true$
end if
$k = k - 1$
continue
end if
end if
$y_{k+1} = y_k + Um$
$t_{k+1} = t_k + h_{k+1}$
$h_{k+2} =$**min**$(fac\, h_{k+1}, fac_{max}\, h_{k+1}, t_F - t_{k+1})$
$rejected = false$
end for
end function

ROS3P

We will give the coefficients of the ROS3P method in Table 2, where $\hat{m} = \Gamma^{-\top}\hat{b}$ and $\gamma = \gamma_{11} = \gamma_{22} = \gamma_{33}$.

$\gamma = 7.886751345948129e-01$	
$a_{21} = 1.267949192431123e+00$	$c_{21} = -1.607695154586736e+00$
$a_{31} = 1.267949192431123e+00$	$c_{31} = -3.464101615137755e+00$
$a_{32} = 0.000000000000000e+00$	$c_{32} = -1.732050807568877e+00$
$\alpha_1 = 0.000000000000000e+00$	$\gamma_1 = 7.886751345948129e-01$
$\alpha_2 = 1.000000000000000e+00$	$\gamma_2 = -2.113248654051871e-01$
$\alpha_3 = 1.000000000000000e+00$	$\gamma_3 = -1.077350269189626e+00$
$m_1 = 2.000000000000000e+00$	$\hat{m}_1 = 2.113248654051871e+00$
$m_2 = 5.773502691896258e-01$	$\hat{m}_2 = 1.000000000000000e+00$
$m_3 = 4.226497308103742e-01$	$\hat{m}_3 = 4.226497308103742e-01$

Table 2: Coefficients of the Rosenbrock method ROS3P.

3 Cardiac reaction-diffusion models

In this section we will present the governing equations of cardiac electrophysiology and a modification of the original two variable ionic model of Fitz-Hugh and Nagumo. For more details we refer to [2] and [1].

The model of cardiac ventricular tissue is based on the two following physiological assumptions:

- the tissue can be modeled as an arrangements of cardiac cells, which orientation can be defined along their principal axis and rotates counterclockwise from epicardium to endocardium,
- additionally cardiac cells are organized in a laminar structure, which is a set of muscle sheets running radially from epicardium to endocardium.

Let Ω be the computational domain representing a piece of cardiac tissue. Thanks to the previous considerations at every point $\mathbf{x} \in \Omega$ we can define a local frame of reference by the triplet of orthonormal axis $\mathbf{a}_l(\mathbf{x}), \mathbf{a}_t(\mathbf{x}), \mathbf{a}_n(\mathbf{x})$, with $\mathbf{a}_l(\mathbf{x})$ parallel to the fiber, $\mathbf{a}_t(\mathbf{x})$ and $\mathbf{a}_n(\mathbf{x})$ tangent and orthogonal to the radial laminae.

Anisotropic bidomain model

In the bidomain model the tissue is represented by the superposition of two anisotropic continuous media, the intra and extra cellular media. Both media are anisotropic since the electrical conductivity is different in the direction parallel to the fibers $\mathbf{a}_l(\mathbf{x})$ and in the orthogonal directions $\mathbf{a}_t(\mathbf{x})$ and $\mathbf{a}_n(\mathbf{x})$.

We assume the following decomposition for the anisotropic conductivity tensors $D_i(\mathbf{x}), D_e(\mathbf{x})$

$$D_{i,e}(\mathbf{x}) = \sigma_l^{i,e}\mathbf{a}_l(\mathbf{x})\mathbf{a}_l(\mathbf{x})^\top + \sigma_t^{i,e}\mathbf{a}_t(\mathbf{x})\mathbf{a}_t(\mathbf{x})^\top + \sigma_n^{i,e}\mathbf{a}_n(\mathbf{x})\mathbf{a}_n(\mathbf{x})^\top$$

where $\sigma_{l,t,n}^{i,e}$ are the conductivity coefficients in the intra and extracellular media measured in the corresponding directions. The intra and extra cellular electric potentials u_i, u_e are described by a system of two parabolic reaction-diffusion equations and a system of ordinary differential equations. We denote by $v = u_i - u_e$ the difference of potential across the two media and by

$$I_m = c_m \frac{\partial v}{\partial t} + I_{ion}(v, w, c)$$

the membrane current per unit volume, where I_{ion} is determined by the ionic model and $c_m = \chi C_m$ with χ the ratio of membrane area per tissue volume and C_m the surface capacitance.

The bidomain model can be written as

$$\begin{cases} c_m \dfrac{\partial v}{\partial t} - \operatorname{div}(D_i \nabla u_i) + I_{ion}(v, w, c) = 0 & \Omega \times (0,T) \text{ (10a)} \\ -c_m \dfrac{\partial v}{\partial t} - \operatorname{div}(D_e \nabla u_e) - I_{ion}(v, w, c) = -I_{app}^e & \Omega \times (0,T) \text{ (10b)} \\ \dfrac{\partial w}{\partial t} = R(v, w) & \Omega \times (0,T) \text{ (10c)} \\ \dfrac{\partial c}{\partial t} = S(v, w, c) & \Omega \times (0,T) \text{ (10d)} \\ \mathbf{n}^\top D_{i,e} \nabla u_{i,e} = 0 & \partial\Omega \times (0,T) \text{ (10e)} \\ v(\mathbf{x}, 0) = v_0(\mathbf{x}),\ w(\mathbf{x}, 0) = w_0(\mathbf{x}),\ c(\mathbf{x}, 0) = c_0(\mathbf{x}) & \Omega, \qquad \text{(10f)} \end{cases}$$

where I_{app}^e is an applied current, which is the stimulus for the heart contraction, satisfying the compatibility condition $\int_\Omega I_{app}^e = 0$ arising from the boundary conditions. By the Fick law $\mathbf{j}_{i,e} = -D_{i,e} \nabla u_{i,e}$ are the intra and extra cellular current density, the current conservation law says that

$$\operatorname{div} \mathbf{j}_i = -I_m \quad \text{and} \quad \operatorname{div} \mathbf{j}_e = I_m - I_{app}^e.$$

Anisotropic monodomain model

The bidomain model requires the inversion of a often ill conditioned matrix and the fast variations in time and space of the potential require a fine grid in space and time. For these reasons simulations with the bidomain model can be computationally expensive. The monodomain model is an approximation of the bidomain model with one parabolic equation.

We denote by $J_{tot} = \mathbf{j}_i + \mathbf{j}_e = -D_i \nabla u_i - D_e \nabla u_e$ the total current flowing across the medias and recalling that $u_i = v + u_e$ we get the relation

$$\begin{aligned} & J_{tot} = -D_i \nabla v - D \nabla u_e \\ \Leftrightarrow \qquad & \nabla u_e = -D^{-1} D_i \nabla v - D^{-1} J_{tot}, \end{aligned} \tag{11}$$

where $D = D_i + D_e$. Using the properties of the orthonormal frame it is easy to verify that

$$D_e = \left(\mu_l^e \mathbf{a}_l(\mathbf{x})\mathbf{a}_l(\mathbf{x})^\top + \mu_t^e \mathbf{a}_t(\mathbf{x})\mathbf{a}_t(\mathbf{x})^\top + \mu_n^e \mathbf{a}_n(\mathbf{x})\mathbf{a}_n(\mathbf{x})^\top\right) D$$

where $\mu_{l,t,n}^e = \frac{\sigma_{l,t,n}^e}{\sigma_{l,t,n}^e + \sigma_{l,t,n}^i}$. Summing equations (10a) and (10b) we obtain $\operatorname{div} J_{tot} = -I_{app}^e$ and assuming that the conductivity coefficients are constant in space we get

$$\begin{aligned}
\operatorname{div}\left(D_e D^{-1} J_{tot}\right) &= \mu_l^e \operatorname{div}\left(\mathbf{a}_l(\mathbf{x})\mathbf{a}_l(\mathbf{x})^\top J_{tot}\right) + \mu_t^e \operatorname{div}\left(\mathbf{a}_t(\mathbf{x})\mathbf{a}_t(\mathbf{x})^\top J_{tot}\right) \\
&\quad + \mu_n^e \operatorname{div}\left(\mathbf{a}_n(\mathbf{x})\mathbf{a}_n(\mathbf{x})^\top J_{tot}\right) \\
&= -\mu_l^e I_{app}^e + (\mu_t^e - \mu_l^e) \operatorname{div}\left(\mathbf{a}_t(\mathbf{x})\mathbf{a}_t(\mathbf{x})^\top J_{tot}\right) \\
&\quad (\mu_n^e - \mu_l^e) \operatorname{div}\left(\mathbf{a}_n(\mathbf{x})\mathbf{a}_n(\mathbf{x})^\top J_{tot}\right), \qquad (12)
\end{aligned}$$

where we have used $I = \mathbf{a}_l(\mathbf{x})\mathbf{a}_l(\mathbf{x})^\top + \mathbf{a}_t(\mathbf{x})\mathbf{a}_t(\mathbf{x})^\top + \mathbf{a}_n(\mathbf{x})\mathbf{a}_n(\mathbf{x})^\top$. Neglecting the two last terms, which will in any case vanish in a media with equal anisotropic ratio $\mu_l^e = \mu_t^e = \mu_n^e$, we get

$$\operatorname{div}\left(D_e D^{-1} J_{tot}\right) \approx -\mu_l^e I_{app}^e. \qquad (13)$$

Using (11) we can rewrite equation (10b) as

$$-c_m \frac{\partial v}{\partial t} + \operatorname{div}\left(D_e D^{-1} D_i \nabla v\right) + \operatorname{div}\left(D_e D^{-1} J_{tot}\right) - I_{ion}(v, w, c) = -I_{app}^e$$

and making the approximation (13) we obtain

$$-c_m \frac{\partial v}{\partial t} + \operatorname{div}\left(D_e D^{-1} D_i \nabla v\right) - I_{ion}(v, w, c) \approx -(1 - \mu_l^e) I_{app}^e.$$

Let us consider the boundary conditions, we can write

$$\begin{aligned}
\mathbf{n}^\top D_e D^{-1} J_{tot} &= \mu_l^e \mathbf{n}^\top \mathbf{a}_l(\mathbf{x})\mathbf{a}_l(\mathbf{x})^\top J_{tot} + \mu_t^e \mathbf{n}^\top \mathbf{a}_t(\mathbf{x})\mathbf{a}_t(\mathbf{x})^\top J_{tot} \\
&\quad + \mu_n^e \mathbf{n}^\top \mathbf{a}_n(\mathbf{x})\mathbf{a}_n(\mathbf{x})^\top J_{tot} \\
&= \mu_l^e \mathbf{n}^\top J_{tot} + (\mu_t^e - \mu_l^e)\, \mathbf{n}^\top \mathbf{a}_t(\mathbf{x})\mathbf{a}_t(\mathbf{x})^\top J_{tot} \\
&\quad + (\mu_n^e - \mu_l^e)\, \mathbf{n}^\top \mathbf{a}_n(\mathbf{x})\mathbf{a}_n(\mathbf{x})^\top J_{tot}.
\end{aligned}$$

The conditions (10e) implies $\mathbf{n}^\top J_{tot} = 0$, thus J_{tot} is parallel to $\partial\Omega$. Assuming that the fibers are also parallel to $\partial\Omega$ we obtain $\mathbf{a}_t(\mathbf{x})^\top J_{tot} = 0$ and $\mathbf{n}^\top \mathbf{a}_n(\mathbf{x}) = 0$, it follows

$$\mathbf{n}^\top D_e D^{-1} J_{tot} = 0. \qquad (14)$$

Combining (10e), (11) and (14) we get

$$-\mathbf{n}^\top D_e D^{-1} D_i \nabla v = \mathbf{n}^\top D_e D^{-1} J_{tot} + \mathbf{n}^\top D_e \nabla u_e = 0.$$

Finally the monodomain model equations are written as

$$\begin{cases} c_m \dfrac{\partial v}{\partial t} - \operatorname{div}(D_m \nabla v) + I_{ion}(v,w,c) = (1-\mu_l^e) I_{app}^e & \Omega \times (0,T) \quad (15a) \\ \dfrac{\partial w}{\partial t} = R(v,w) & \Omega \times (0,T) \quad (15b) \\ \dfrac{\partial c}{\partial t} = S(v,w,c) & \Omega \times (0,T) \quad (15c) \\ \mathbf{n}^\top D_m \nabla v = 0 & \partial\Omega \times (0,T) \quad (15d) \\ v(\mathbf{x},0) = v_0(\mathbf{x}),\ w(\mathbf{x},0) = w_0(\mathbf{x}),\ c(\mathbf{x},0) = c_0(\mathbf{x}) & \Omega, \quad (15e) \end{cases}$$

where $D_m = D_e D^{-1} D_i$.

Ionic FitzHugh-Nagumo model variant

The FitzHugh-Nagumo model is a simplified model of the ionic current I_{ion} originally developed for neurons, here we present a variant by Rogers and McCulloch [7] for cardiac cells. The model reads

$$I_{ion}(v,w) = Gv\left(1-\frac{v}{v_{th}}\right)\left(1-\frac{v}{v_p}\right) + \eta_1 v w,$$
$$\frac{\partial w}{\partial t} = \eta_2 \left(\frac{v}{v_p} - \eta_3 w\right),$$

where the coefficients are given in the Table 3.

G	v_{th}	v_p	η_1	η_2	η_3
1.5	13	100	4.4	0.012	1

Table 3: Coefficients of the modified Fitz-Hugh Nagumo model.

4 Operator splitting

If we integrate the monodomain model equations with the explicit Euler scheme the time step must be very small in order to keep stability and accuracy. On the other hand implicit methods are computationally expensive due to the non linearities of the system. Moreover, the use of a high order explicit adaptive method will be spoiled by the presence of high gradients in

the computational domain. For this reason, we opt to use a fractional step method where it will be possible to use time adaptive solvers.

Let us consider the abstract Cauchy problem

$$\begin{cases} \frac{\partial y}{\partial t}(t,\mathbf{x}) = (\mathcal{L}_1 + \mathcal{L}_2)\, y(t,\mathbf{x}) \quad t \in (0,T), \\ y(0,\mathbf{x}) = y_0(\mathbf{x}), \end{cases}$$

where $\mathcal{L}_1$ and $\mathcal{L}_2$ are given operators. We define the splitting step by $h \ll T$ and $t_n = nh$. Consider the sequence of initial valued problems:

$$\begin{cases} \frac{\partial y_1}{\partial t}(t,\mathbf{x}) = \mathcal{L}_1 y_1(t,\mathbf{x}) \qquad t_{n-1} < t \leq t_n, \\ y_1(t_{n-1},\mathbf{x}) = y_{sp}(t_{n-1},\mathbf{x}), \end{cases}$$

and

$$\begin{cases} \frac{\partial y_2}{\partial t}(t,\mathbf{x}) = \mathcal{L}_2 y_2(t,\mathbf{x}) \qquad t_{n-1} < t \leq t_n, \\ y_2(t_{n-1},\mathbf{x}) = y_1(t_n,\mathbf{x}), \end{cases}$$

where $y_2(0) = y_0$ and $y_{sp}(t_n) := y_2(t_n)$ is the splitting solution of the problem.

In the case of the monodomain equations we choose $\mathcal{L}_2$ as the operator associated with the diffusion part, thus to solve the monodomain equations with the Fitz-Hugh Nagumo model for the ionic current we follow the scheme:

$$\begin{cases} c_m \frac{\partial v_1}{\partial t} = (1-\mu_l^e) I_{app}^e - I_{ion}(v_1,w,c) & t_{n-1} < t \leq t_n, \\ \frac{\partial w}{\partial t} = R(v_1,w) & t_{n-1} < t \leq t_n, \\ \frac{\partial c}{\partial t} = S(v_1,w,c) & t_{n-1} < t \leq t_n, \\ v_1(t_{n-1},\mathbf{x}) = v_2(t_{n-1},\mathbf{x}), \end{cases} \tag{16}$$

$$\begin{cases} c_m \frac{\partial v_2}{\partial t} = \operatorname{div}(D_m \nabla v_2) \quad t_{n-1} < t \leq t_n, \\ v_2(t_{n-1},\mathbf{x}) = v_1(t_n,\mathbf{x}). \end{cases} \tag{17}$$

The advantage of this method is that (16) can be solved in every point $\mathbf{x}$ of the space indipendently and the system is much simpler. Thus for every time step we can solve system (16) from t_{n-1} to t_n applying an adaptive method.

5 Discretization

The solution of equations (16) and (17) is done independently at each time step. Equation (17) is solved by $\mathbb{P}_1$ finite elements for the spacial discretization and with the backward euler method for the time integration. Let T_h be a trianguation of the domain Ω and $K \in T_h$ an element, we set

$$V_h = \{v_h \in C^0(\Omega) \,:\, v_h|_K \in \mathbb{P}_1(K)\, \forall K \in T_h,\, \mathbf{n}^\top (D_m \nabla v_h) = 0 \text{ in } \partial\Omega\}.$$

Then the weak form of (17) reads

$$\text{Find } v_h \in V_h \text{ such that } \int_\Omega c_m \frac{\partial v_h}{\partial t} u_h = -\int_\Omega (D_m \nabla v_h) \cdot \nabla u_h \qquad \forall u_h \in V_h$$

and the corresponding linear system is

$$c_m M \frac{\partial \mathbf{v}}{\partial t} = -K\mathbf{v}$$

where $M_{ij} = \int_\Omega \phi_j \phi_i$ is the mass matrix, $K_{ij} = \int_\Omega (D_m \nabla \phi_j) \nabla \phi_i$ is the stiffness matrix, $\{\phi_i\}_{i=1}^N$ is the lagrangian basis of V_h corresponding to the nodes $\{\mathbf{x}_i\}_{i=1}^N$ and $\mathbf{v}(t) = (v_i(t))$, where $v_h(\mathbf{x}, t) = \sum_{i=1}^N v_i(t)\phi_i(\mathbf{x})$. Applying the backward Euler method we obtain

$$\left(c_m \frac{M}{h} + K\right) \mathbf{v}^{n+1} = c_m \frac{M}{h} \mathbf{v}^n.$$

where $\mathbf{v}^n = (v_i(t_n))$.

Equation (16) is solved at every node of the mesh with the time adaptive Rosenbrock method ROS3P.

6 Numerical experiments

In this section we will present some results obtained with the methods presented above. First of all we show that with our method we are able to reproduce spiral waves. Then we show the improvements made in the zero dimensional case using the ROS3P method with time step adaptivity. Finally we show the results obtained by the use of the splitting method and ROS3P when solving the monodomain model.

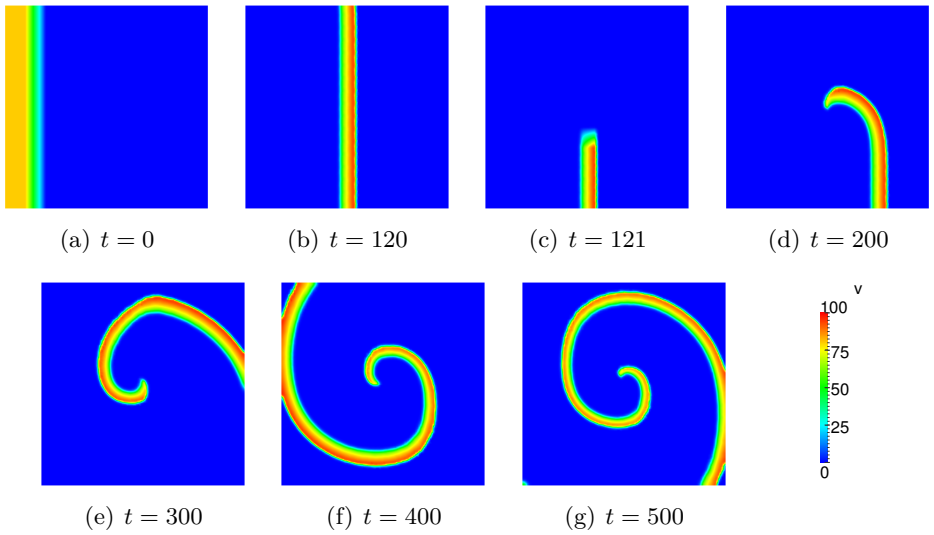
Spiral wave

Cardiac fibrillation is the main cause of sudden cardiac death in the world, this is a disturbance in the heart's rythms which renders it unable to pump blood effectively. Experimental work has shown that some arrhythmias like ventricular fibrillation begin with the presence of waves of electrical activation that rotate at higher frequencies than the heart's natural pacemaker, preventing normal function. Nowadays cardiac models are used to try to understand these phenomenons, this is why it is important that our approach is able to reproduce these waves [3].

To induce spiral waves we have solved to monodomain model in $\Omega = [0, 1] \times [0, 1]$ with the initial conditions

$$v_0(\mathbf{x}) = \begin{cases} 80 & \text{if } x_1 \leq 0.1 \\ 80\frac{0.2 - x_1}{0.1} & \text{if } 0.1 \leq x_1 \leq 0.2 \\ 0 & \text{if } x_1 \geq 0.2. \end{cases} \qquad w_0(\mathbf{x}) = 0.001$$

and at $t = 120$ we have forced $v(\mathbf{x}, t) = 0$ if $x_2 > 0.3$. This leads to spiral waves, in Figure 2 we can see a snapshot of the solution.



(a) $t = 0$ (b) $t = 120$ (c) $t = 121$ (d) $t = 200$

(e) $t = 300$ (f) $t = 400$ (g) $t = 500$

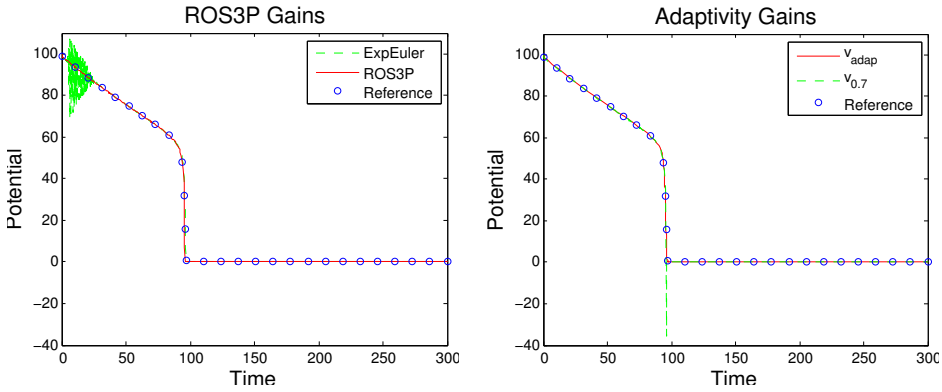
Figure 2: Spiral wave effect.

Improvements in the zero dimensional case

Here we will show the improvements made in the zero dimensional case using the Rosenbrock method ROS3P instead of the explicit Euler method and also the gains obtained using the adaptive time step strategy. For this let us consider the monodomain model in the zero dimensional case without applied current and the FitzHugh-Nagumo model for the ionic currents, which leads to the following problem

$$\begin{cases} \frac{\partial v}{\partial t} = -Gv\left(1 - \frac{v}{v_{th}}\right)\left(1 - \frac{v}{v_p}\right) - \eta_1 vw & 0 < t \leq 300 \\ \frac{\partial w}{\partial t} = \eta_2\left(\frac{v}{v_p} - \eta_3 w\right) & 0 < t \leq 300 \\ v(0) = 100 \\ w(0) = 0.025. \end{cases} \tag{18}$$

Since explicit Euler is a first order method and ROS3P is of third order to compare them we had used a time step $h = 0.3$ for explicit Euler and $h = 0.3^{1/3} \approx 0.6694$ for ROS3P. The results can be seen in Figure 3(a), where they are also compared with a reference solution computed with explicit Euler with a time step $h = 0.0001$. We remark that the explicit Euler method with $h = 0.3$ has some stability problems while ROS3P does not oscillate although with the chosen time step they should have similar precision. This can be explained by the fact that Rosenbrock methods are



(a) Comparing Explicit Euler with $h = 0.3$ and ROS3P with $h = 0.3^{1/3} \approx 0.6694$.

(b) Differences between ROS3P with and without step size adaptivity.

Figure 3: Gains using the right strategies.

derived from implicit Runge-Kutta methods, thus they are more suitable for stiff problems than the explicit ones.

We will denote by v_{adap} the solution given by the ROS3P method with time step adaptivity and a starting time step $h_1 = 1$ and by v_h the solution given also by ROS3P but without adaptivity and a fixed time step h. In Figure 3(b) we can see the reference solution and two other solutions: v_{adap} and $v_{0.7}$. It is easy to see that the adaptive version gives a more accurate solution and moreover it does only take 53 iterations and 0.002 seconds, while the non adaptive version needs 430 iterations and 0.016 seconds, that is 8 times more time. If we compute the relative errors we obtain

$$\frac{\max |v_{ref} - v_{adap}|}{\max |v_{ref}|} = 0.05,$$

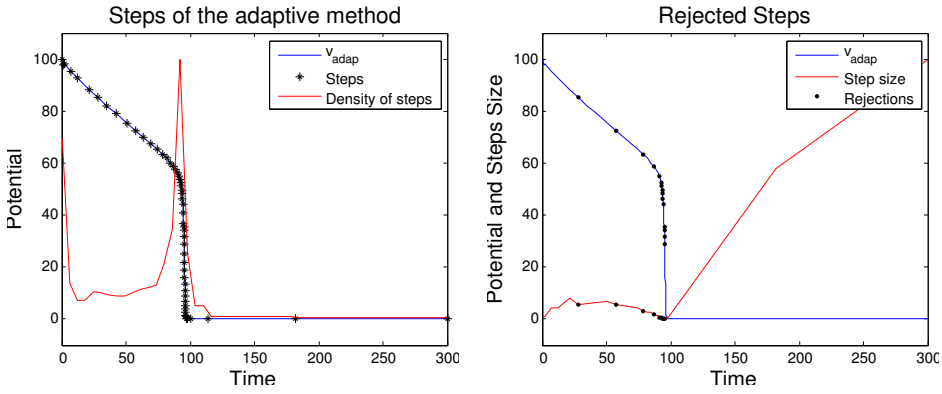
$$\frac{\max |v_{ref} - v_{0.7}|}{\max |v_{ref}|} = 0.44,$$

where v_{ref} is the reference solution. To obtain a comparable relative error we use a time step $h = 0.6$, which gives

$$\frac{\max |v_{ref} - v_{0.6}|}{\max |v_{ref}|} = 0.06,$$

which requires 501 iterations and 0.022 seconds. This is nearly 10 times more iterations and 11 times more time, than the adaptive ROS3P method.

In Figure 4(a) we see the steps done by ROS3P with time step adaptivity, observe that they are very dense where the solution has a high slope and where it is almost constant the method does just a few steps. The have e better idea of the time steps size we can look at Figure 4(b) where we



(a) Steps taken by the adaptive method.

(b) Illustration of the rejected steps and the steps size magnified.

Figure 4: Details of the steps taken by the ROS3P method with time step adaptivity.

compare the solution v_{adap} with the steps size magnified and we also show where there had been a rejecteion of the step. Observe that the most part of the rejected steps are in the region where the solution has a high slope and where the solution is constant the steps size is increasing.

Scalability test

Here we will show some performance results of the splitting method coupled with ROS3P when applied to the system (16)-(17). We have chosen the domain $\Omega = [0,1] \times [0,1]$ and two initial conditions: the gating variable is in both cases

$$w_0(\mathbf{x}) = 0.001,$$

while for the potential we have

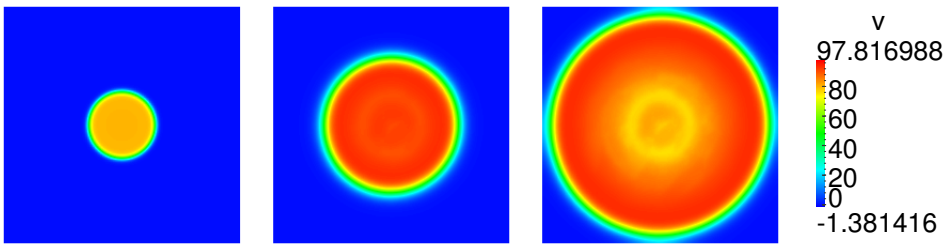
$$v_0^1(\mathbf{x}) = 80\left(\frac{1}{2} + \frac{1}{2}\tanh\left(-300((x_1 - 0.4)(x_1 - 0.6) + (x_2 - 0.4)(x_2 - 0.6))\right)\right)$$

and

$$v_0^2(\mathbf{x}) = \begin{cases} 80 & \text{if } x_1 \leq 0.1 \\ 80\frac{0.2 - x_1}{0.1} & \text{if } 0.1 \leq x_1 \leq 0.2 \\ 0 & \text{if } x_1 \geq 0.2. \end{cases}$$

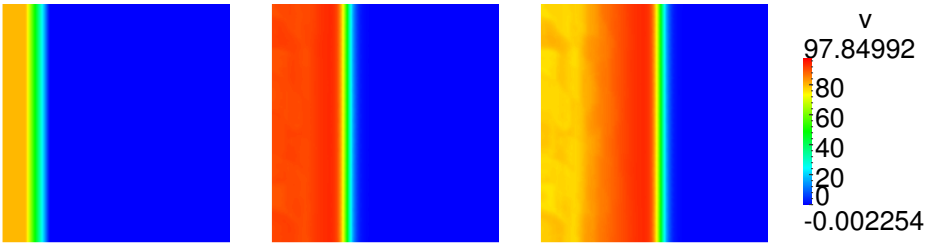
In Figures 5 and 6 we can see some snapshots of the solutions with initial conditions (v_0^1, w_0) and (v_0^2, w_0).

To check the scalability behaviour of the method we have done the several simulations on four different meshes changing the number of processors used during the computations. In the following graphs we plot the time taken by



(a) $t = 0$ (b) $t = 10$ (c) $t = 20$

Figure 5: Solution of the monodomain model with initial condition (v_0^1, w_0).



(a) $t = 0$ (b) $t = 10$ (c) $t = 20$

Figure 6: Solution of the monodomain model with initial condition (v_0^2, w_0).

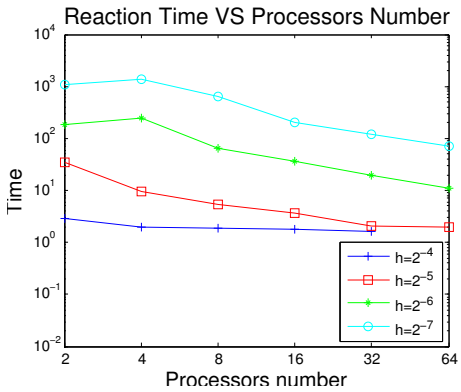
the reaction and the diffusion steps over the number of processors for the different meshes, where h here stays for the element size.

The results of the tests are exibithed in Figures 7 and 8. From Figures 7(a) and 8(a) we see that the reaction step has some scalability effect, except for the case of the coarsest mesh. This is due by the fact that the resolution of this step is done independently at every node of the mesh, thanks to the operator splitting strategy. The diffusion step has a weaker scalability effect, see Figures 7(b) and 8(b), which can be explained by the fact that me must solve a huge linear system over the whole mesh, thus the result at each node depends on its neghbours, which implies more communication between the processors.

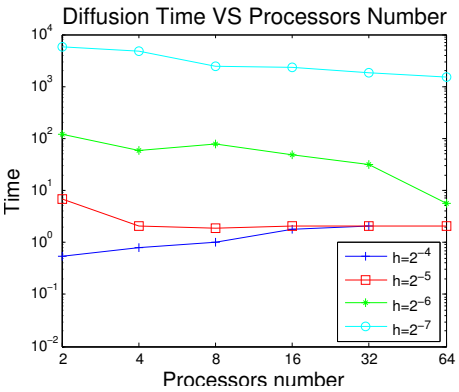
Comparison with ICI and SVI

In this section we compare the splitting method coupled with ROS3P with two other methods: ionic current interpolation (ICI) and state variable interpolation (SVI). First we give a brief description of ICI and SVI [6], then we make the comparison.

For simplicity we suppose $I_{app} = 0$, then the monodomain equations can

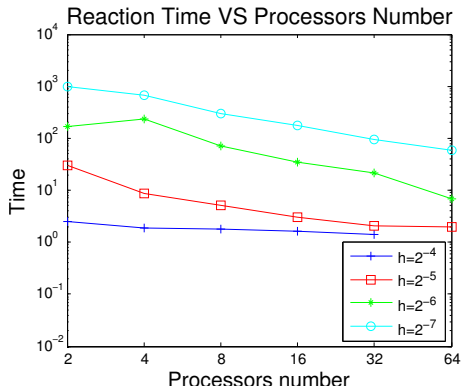


(a) Time taken by the reaction step in the splitting method.

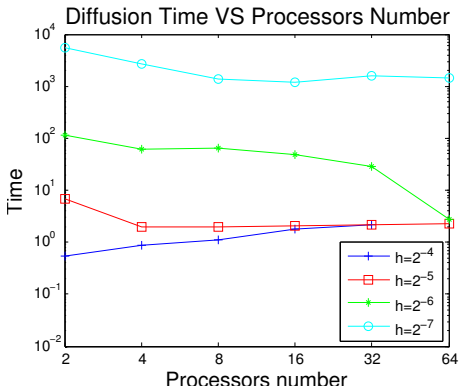


(b) Time taken by the diffusion step in the splitting method.

Figure 7: Scalability of the reaction and the diffusion step in the splitting method when applied to the monodomain problem with initial condition (v_0^1, w_0) for different meshes.



(a) Time taken by the reaction step in the splitting method.



(b) Time taken by the diffusion step in the splitting method.

Figure 8: Scalability of the reaction and the diffusion step in the splitting method when applied to the monodomain problem with initial condition (v_0^2, w_0) for different meshes.

be rewritten

$$c_m \frac{\partial v}{\partial t} - \operatorname{div}(D_m \nabla v) + I_{ion}(v,u) = 0$$
$$\frac{\partial u}{\partial t} = f(v,u),$$

where u stays for the gating variables and the ionic current concentrations. Employing a semi-implicit time discretization we obtain the equations

$$c_m \frac{v^{n+1} - v^n}{h} - \operatorname{div}(D_m \nabla v^{n+1}) + I_{ion}(v^n, u^{n+1}) = 0 \tag{19}$$
$$\frac{u^{n+1} - u^n}{h} = f(v^n, u^n).$$

Equation (19) leads to a linear system

$$Av^{n+1} = b^{n+1}$$

where

$$b_i^{n+1} = \int_\Omega \left(\frac{c_m}{h} v^n - I_{ion}(v^n, u^{n+1}) \right) \phi_i. \tag{20}$$

At the beginning of each time step u^{n+1} is computed with forward Euler and then we compute b^{n+1}, which requires the evaluation of $I_{ion}(v,u)$ at the interior of the elements.

Since we know the values of v^n and u^{n+1} at each node we can compute I_{ion} at the nodes and then make an interpolation to evaluate I_{ion} in the interior. With the ICI approach I_{ion} is given by

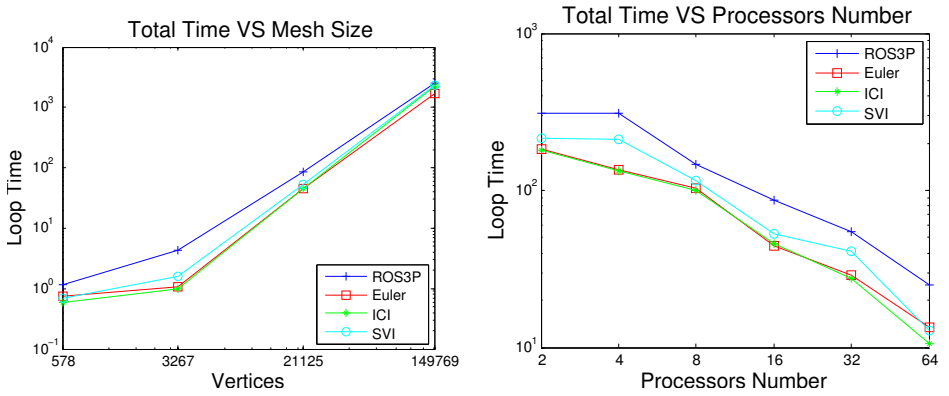
$$I_{ion}^{ICI}(\mathbf{x}) = \sum_{i=1}^{N} I_{ion}(v_i^n, u_i^{n+1}) \phi_i(\mathbf{x}).$$

Instead of interpolate I_{ion}, in the SVI approach we interpolate the electric potential and the state variables at the interior of the elements, which gives

$$I_{ion}^{SVI}(\mathbf{x}) = I_{ion}\left(\sum_{i=1}^{N} v_i^n \phi_i(\mathbf{x}), \sum_{i=1}^{N} u_i^{n+1} \phi_i(\mathbf{x}) \right).$$

From Figure 9(a) we see that ROS3P is the most costly method but on the other hand it gives order three convergence, while the others have only order one in time. Moreover it seems that it has the same asymptotical behaviour of the other methods. In fact, as can be seen in Figures 7 and 8, if we reduce the mesh size h the time required to solve the linear system seems to increase quicker than the time required to solve the ROS3P part.

Figure 9(b) shows that for a fixed mesh the scalability factors of the methods are proportional.



(a) Time taken by the different methods versus the mesh size on 16 processors.

(b) Time taken by the different methods versus the number of processors on the mesh with $h = 2^{-6}$.

Figure 9: Comparison of the time taken by the different methods.

7 Conclusions

In this report we have presented numerical methods that were implemented and tested in LifeV for the simulation of cardiac electrophysiology. In particular we have derived the Rosenbrock methods from the diagonally implicit Runge-Kutta methods. We also use a splitting scheme that allows us to use adaptivity of the ROS3P method.

In the zero dimensional case we saw that using more sophisticated techniques can apport a great gain in terms of computational time and accuracy. On the other hand in the multidimensional case we saw that, although our approach gives good accuracy in time, it is the most computationally expensive, while the other methods also give stable solutions.

To better exploit the stability properties of the ROS3P method we should consider stiffer systems, than the one found with the FitzHugh-Nagumo model. Further developments should incorporate more physiologically detailed models of electrophysiology.

References

[1] R.H. Clayton et al. Models of cardiac tissue electrophysiology: Progress, challenges and open questions. *Elsevier*, Progress in Biophysics and Molecular Biology, 2010.

[2] Piero Colli Franzone, Peter Deuflhard, Bodo Erdmann, Jens Lang, and Luca F. Pavarino. Adaptivity in space and time for reaction-diffusion systems in electrocardiology. *SIAM J. Sci. Comput.*, 28(3):942–962 (electronic), 2006.

[3] Fenton F. H., Cherry E. M., Hastings H. M., and Evans S. J. Multiple mechanisms of spiral wave breakup in a model of cardiac electrical activity. *Chaos*, 12(3):852–892 (electronic), 2002.

[4] E. Hairer and G. Wanner. *Solving ordinary differential equations. II*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2010. Stiff and differential-algebraic problems, Second revised edition, paperback.

[5] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2010. Structure-preserving algorithms for ordinary differential equations, Reprint of the second (2006) edition.

[6] Pathmanathan P., Mirams G. R., Southern J., and Whiteley J. P. The significant effect of the choice of ionic current integration method in cardiac electro-physiological simulations. *Int. J. Numer. Meth. Biomed. Engng*, 27:1751–1770 (electronic), 2011.

[7] Rogers and McCulloch. A collocation-galerkin finite element model of cardiac action potential propagation. *IEEE Trans. Biomed. Engrg.*, 41:743–757, 1994.