ESAIM: M2AN 58 (2024) 2225–2254
https://doi.org/10.1051/m2an/2024030

ESAIM: Mathematical Modelling and Numerical Analysis
www.esaim-m2an.org

EXPLICIT STABILIZED MULTIRATE METHODS FOR THE MONODOMAIN MODEL IN CARDIAC ELECTROPHYSIOLOGY ⋆

Giacomo Rosilho de Souza[1,**], Marcus J. Grote[2], Simone Pezzuto[1,3] and Rolf Krause[1,4]

Abstract. Fully explicit stabilized multirate (mRKC) methods are well-suited for the numerical solution of large multiscale systems of stiff ordinary differential equations thanks to their improved stability properties. To demonstrate their efficiency for the numerical solution of stiff, multiscale, nonlinear parabolic PDE's, we apply mRKC methods to the monodomain equation from cardiac electrophysiology. In doing so, we propose an improved version, specifically tailored to the monodomain model, which leads to the explicit exponential multirate stabilized (emRKC) method. Several numerical experiments are conducted to evaluate the efficiency of both mRKC and emRKC, while taking into account different finite element meshes (structured and unstructured) and realistic ionic models. The new emRKC method typically outperforms a standard implicit-explicit baseline method for cardiac electrophysiology. Code profiling and strong scalability results further demonstrate that emRKC is faster and inherently parallel without sacrificing accuracy.

Mathematics Subject Classification. 65L04, 65L06, 65L10, 65L20.

Received December 4, 2023. Accepted April 22, 2024.

1. Introduction

Spatial discretizations of parabolic partial differential equations (PDEs), possibly nonlinear, typically lead to large systems of stiff ordinary differential equations (ODEs). The degree of stiffness, be it mild or severe, depends in particular on the rate of change of individual components or the mesh size. For time discretization, there is generally a trade-off between explicit and implicit time-marching schemes. Explicit schemes are easier to implement and also cheaper per time-step, but require more steps due to stability constraints. In contrast, implicit schemes require sophisticated (parallel) solvers for the needed linear or nonlinear systems and thus are more expensive per time-step, but they also require fewer time-steps. Choosing between explicit and implicit time integration is a delicate process which depends on several factors, such as the stiffness of the system

Keywords and phrases. Multirate explicit stabilized methods, Rush–Larsen, electrophysiology, monodomain model, ionic model, local time-stepping.

⋆ *Dedicated to the memory of Assyr Abdulle.*

[1] Euler Institute, Universitá della Svizzera italiana, Via G. Buffi 13, 6900 Lugano, Switzerland.

[2] Department of Mathematics, University of Basel, Rheinsprung 21, 4051 Basel, Switzerland.

[3] Laboratory for Mathematics in Biology and Medicine, Department of Mathematics, Universitá di Trento, Via Sommarive 14, 38123 Trento, Italy.

[4] Faculty of Mathematics and Informatics, FernUni Schweiz, Schinerstrasse 18, 3900 Brig, Switzerland.

**Corresponding author: giacomo.rosilhodesouza@usi.ch

or the efficiency of the linear or nonlinear solvers available. Indeed, an unfortunate choice can increase the computational cost by orders of magnitude due to abrupt changes in any methods' performance.

Explicit stabilized methods, such as Runge–Kutta–Chebyshev (RKC) methods [59, 63], strike a balance between explicit and implicit schemes while their computational cost smoothly depends on the degree of stiffness. On the one hand, they are fully explicit and thus simple to implement in parallel, even in the presence of nonlinearity; on the other hand, they adapt to the current degree of stiffness by modifying the number of stages accordingly. Since their stability domain grows quadratically with the number of stages, they are significantly more efficient than standard explicit methods. Compared to implicit methods, they can also be more efficient in particular for large-scale three-dimensional linear, or nonlinear, problems [1, 3, 4, 18]. In summary, explicit stabilized schemes combine the advantages of both explicit and implicit methods while avoiding any abrupt performance deterioration when transitioning from a non-stiff to stiff regime, or from linear to nonlinear problems.

When stiffness is induced only by a few components, as in the presence of spatial local mesh refinement or coupled chemical reactions, the efficiency of RKC methods can deteriorate. To overcome that crippling effect due to a few severely stiff components, multirate Runge–Kutta–Chebyshev (mRKC) methods were proposed in [5]; they remain fully explicit, and thus easy to parallelize, though with a stability condition independent of those few severely stiff components. Explicit multirate RKC methods thus permit to circumvent any overly stringent stability constraint due to local mesh refinement; hence, they also extend local time-stepping (LTS) methods [27–30] to parabolic PDE's.

Here we shall consider the monodomain model for cardiac electrophysiology, which consists of a parabolic PDE for the diffusion of the electric potential and a system of ODE's for the ionic model of the intracellular and membrane dynamics of the cardiac cell [13]. After spatial discretization, the monodomain model can be written as

$$y' = f_F(y) + f_S(y) + f_E(y), \qquad y(0) = y_0, \tag{1}$$

where f_F is cheap but stiff ("F" for "fast"), f_S is non-stiff but expensive ("S" for "slow"), and f_E is in general a severely stiff term of the form

$$f_E(y) = \Lambda(y)(y - y_\infty(y)), \tag{2}$$

with $\Lambda(y)$ a diagonal matrix and $y_\infty(y)$ a nonlinear function. Typically, f_F represents the discrete diffusion operator while f_S, f_E depend on the ionic model; here, f_E corresponds to the gating variables and f_S to the remaining ones.

A common and efficient approach for the time integration of the non-diffusive component of (1) is based on the Rush–Larsen scheme [14, 53]. The Rush–Larsen scheme is a first-order splitting method employing explicit Euler for f_S and exponential Euler for f_E (hence the "E" notation for the "exponential") – see [39] for a recent review paper on the possible variants of this approach. The diffusive term f_F is treated either explicitly or implicitly, depending on the degree of stiffness. For the monodomain model, f_F is mildly stiff and, as a consequence, the choice between an explicit or an implicit approach somewhat challenging. Clearly, explicit stabilized schemes, such as RKC methods, can efficiently treat the f_S and f_F terms, but often perform poorly for the full monodomain model, due to its multiple components and inherent multiscale nature.

In contrast, multirate explicit stabilized methods such as mRKC [5, 16] are more appropriate for the monodomain model because they can adapt the number of stages locally to individual components depending on their respective degree of stiffness. Here following [14, 53], we shall further take full advantage of the special structure (2) of f_E by integrating it with an exponential method, while integrating f_F and f_S explicitly with the mRKC scheme. We denote this combined exponential–mRKC time integrator by *exponential multirate RKC* (emRKC). The emRKC method is cheap because it is explicit while utilizing an exponential integrator for the most severely stiff yet diagonal terms. Albeit only first-order, it is more accurate than other commonly used first-order schemes, as its special structure, inherited from mRKC, guarantees a robust coupling between the different model components. Although the emRKC method was developed specifically for the monodomain

equation (for both ventricular or atrial geometries, or even the entire heart), it can be applied to any physical model that can be written as (1) and (2).

In Section 2, we first present the monodomain model from cardiac electrophysiology. Next, in Section 3, we recall the RKC [59, 63] and mRKC methods from [5, 16], summarize their properties, and describe them from an algorithmic point of view. In Section 4, we first adapt mRKC to the particular structure (1) of the monodomain model, which yields the new emRKC method; then we analyze its accuracy and stability properties. In Section 5, we compare both mRKC and emRKC to a standard baseline method on a sequence of numerical experiments. Both two- and three-dimensional problems are considered, either with structured or unstructured meshes, including realistic ionic models. Concluding remarks are presented in Section 6.

2. Monodomain model

The monodomain model is commonly used to simulate electric potential propagation in cardiac tissue. Though a simplification of the more complex bidomain model, it remains important in practice. In this section, we provide a brief introduction to the model; for a more complete description, we refer to [10]. The monodomain model is given by the following reaction-diffusion system of equations:

$$C_m \frac{\partial V_m}{\partial t} = \chi^{-1} \nabla \cdot \big(\mathbf{D} \nabla V_m\big) + I_{\text{stim}}(t, \boldsymbol{x}) - I_{\text{ion}}(V_m, \boldsymbol{z}_E, \boldsymbol{z}_S), \qquad \text{in } \Omega \times (0, T], \tag{3a}$$

$$\frac{\partial \boldsymbol{z}_E}{\partial t} = \boldsymbol{g}_E(V_m, \boldsymbol{z}_E) = \boldsymbol{\alpha}(V_m)\big(1 - \boldsymbol{z}_E\big) - \boldsymbol{\beta}(V_m)\boldsymbol{z}_E, \qquad \text{in } \Omega \times (0, T], \tag{3b}$$

$$\frac{\partial \boldsymbol{z}_S}{\partial t} = \boldsymbol{g}_S(V_m, \boldsymbol{z}_E, \boldsymbol{z}_S), \qquad \text{in } \Omega \times (0, T], \tag{3c}$$

$$-\mathbf{D} \nabla V_m \cdot \boldsymbol{n} = 0, \qquad \text{on } \partial\Omega \times (0, T], \tag{3d}$$

$$V_m(0, \boldsymbol{x}) = V_{m,0}, \quad \boldsymbol{z}_E(0, \boldsymbol{x}) = \boldsymbol{z}_{E,0}, \quad \boldsymbol{z}_S(0, \boldsymbol{x}) = \boldsymbol{z}_{S,0}, \qquad \text{in } \Omega, \tag{3e}$$

where parameters and variables are described in Table 1. Electric conduction in the myocardium is anisotropic, with a conductivity tensor of the form

$$\mathbf{D}(\boldsymbol{x}) = \sigma_\ell \boldsymbol{a}_\ell(\boldsymbol{x}) \otimes \boldsymbol{a}_\ell(\boldsymbol{x}) + \sigma_t \boldsymbol{a}_t(\boldsymbol{x}) \otimes \boldsymbol{a}_t(\boldsymbol{x}) + \sigma_n(\mathbf{I} - \boldsymbol{a}_\ell(\boldsymbol{x}) \otimes \boldsymbol{a}_\ell(\boldsymbol{x}) - \boldsymbol{a}_t(\boldsymbol{x}) \otimes \boldsymbol{a}_t(\boldsymbol{x})), \tag{4}$$

where $\sigma_\star = \sigma_\star^i \sigma_\star^e / (\sigma_\star^i + \sigma_\star^e)$ and $\sigma_\star^i$, $\sigma_\star^e$ are also given in Table 1. The vector fields $\boldsymbol{a}_\ell(\boldsymbol{x})$ and $\boldsymbol{a}_t(\boldsymbol{x})$ are orthonormal and are aligned with the local fiber and sheet direction, respectively.

The ionic current I_{ion} models the total transmembrane currents through a set of gating and auxiliary variables, respectively denoted by $\boldsymbol{z}_E$ and $\boldsymbol{z}_S$. The gating variables $\boldsymbol{z}_E$ are probabilities that control the opening and closing of ion channels in the cell membrane, whereas the remaining variables $\boldsymbol{z}_S$ represent other quantities, such as ion concentration, or intracellular calcium dynamic. Many different ionic models exist in the literature, depending on the type of cell and the pathophysiological state, but in most cases the overall structure is fixed as in (3) and follows the so-called Hodgkin–Huxley formalism. Specifically, the total current I_{ion} is the sum of several ionic currents, each depending on its own set of gating and auxiliary variables. An archetypal instance is the celebrated Hodgkin–Huxley model for electric propagation in axons:

$$I_{\text{ion}}(V_m, \boldsymbol{z}_E) = g_{\text{Na}} m^3 h (V_m - V_{\text{Na}}) + g_{\text{K}} n^4 (V_m - V_{\text{K}}) + g_{\text{leak}} (V_m - V_{\text{leak}}),$$

where $\boldsymbol{z}_E = (m, h, n)$ and there is no auxiliary variable $\boldsymbol{z}_S$. State-of-the-art cardiac ionic models are more advanced in the sense that they include several more currents and the intracellular calcium dynamic, thus significantly increasing the computational complexity and temporal stiffness. In this respect, it is common to take advantage of the Hodgkin–Huxley structure of the system to avoid implicit numerical schemes or excessively small time steps. In fact, equation (3b) can be rewritten as:

$$\frac{\partial \boldsymbol{z}_E}{\partial t} = \boldsymbol{\Lambda}_E(V_m)\big(\boldsymbol{z}_E - \boldsymbol{z}_\infty(V_m)\big), \tag{5}$$

TABLE 1. Description of variables and parameters for the monodomain model. The values of the coefficients are from [42].

Variable	Description	Value	Units
$V_m(t, \boldsymbol{x})$	Transmembrane potential	–	mV
$\boldsymbol{z}_E(t, \boldsymbol{x})$	Vector of gating variables	–	Probability
$\boldsymbol{z}_S(t, \boldsymbol{x})$	Vector of auxiliary ionic variables	–	Depends
$I_{\text{ion}}(V_m, \boldsymbol{z}_E, \boldsymbol{z}_S)$	Transmembrane current density	–	$\mu\text{A mm}^{-2}$
$I_{\text{stim}}(t, \boldsymbol{x})$	Stimulus current density	–	$\mu\text{A mm}^{-2}$
χ	Surface area-to-volume ratio of cardiac cells	140	mm^{-1}
C_m	Cell membrane electrical capacitance	0.01	$\mu\text{F mm}^{-2}$
σ_ℓ^i, σ_t^i, σ_n^i	Intracellular electric conductivities	$[0.17, 0.019, 0.019]$	mS mm^{-1}
σ_ℓ^e, σ_t^e, σ_n^e	Extracellular electric conductivities	$[0.62, 0.24, 0.24]$	mS mm^{-1}
$\boldsymbol{\alpha}(V_m)$, $\boldsymbol{\beta}(V_m)$	Transition rates (diagonal matrices)	–	ms^{-1}

where $\boldsymbol{\Lambda}_E(V_m) = -(\boldsymbol{\alpha}(V_m) + \boldsymbol{\beta}(V_m))$ is a diagonal matrix and $\boldsymbol{z}_{E,\infty}(V_m) = -\boldsymbol{\alpha}(V_m)(\boldsymbol{\alpha}(V_m) + \boldsymbol{\beta}(V_m))^{-1}$ is the steady state value of $\boldsymbol{z}_E$. In general, $\boldsymbol{\Lambda}_E(V_m)$ is very stiff for a physiological range of V_m, but it can be efficiently integrated with exponential methods (hence the "E" notation) since $\boldsymbol{\Lambda}_E(V_m)$ is diagonal. In contrast, $\boldsymbol{g}_S(V_m, \boldsymbol{z}_E, \boldsymbol{z}_S)$ is not stiff ("S" for "slow") and rather expensive to evaluate; thus, it is usually integrated with the explicit Euler scheme. This split approach of integrating $\boldsymbol{z}_E$ and $\boldsymbol{z}_S$ with an exponential and standard explicit method, respectively, was first proposed in the seminal paper of Rush and Larsen [53] and remains very popular even to this day.

2.1. Space discretization

The spatial discretization of the monodomain equations (3) using conforming piecewise linear finite elements leads to the semi-discrete system

$$C_m \mathbf{M} \frac{\mathrm{d}\boldsymbol{V}_m}{\mathrm{d}t} = \chi^{-1} \mathbf{A} \boldsymbol{V}_m + \mathbf{M} \boldsymbol{I}_{\text{stim}}(t) - \mathbf{M} I_{\text{ion}}(t, \boldsymbol{V}_m, \boldsymbol{z}_E, \boldsymbol{z}_S), \tag{6a}$$

$$\frac{\mathrm{d}\boldsymbol{z}_E}{\mathrm{d}t} = g_E(\boldsymbol{V}_m, \boldsymbol{z}_E), \tag{6b}$$

$$\frac{\mathrm{d}\boldsymbol{z}_S}{\mathrm{d}t} = g_S(\boldsymbol{V}_m, \boldsymbol{z}_E, \boldsymbol{z}_S), \tag{6c}$$

with appropriate initial conditions and where $\mathbf{A}$ and $\mathbf{M}$ are respectively the stiffness and mass matrices. (For the sake of simplicity, we keep using the vector notation for $\boldsymbol{z}_S$ and $\boldsymbol{z}_E$.) Equations (6b) and (6c) are evaluated pointwise, due to the absence of differential operators in space in the equations. The nonlinear term I_{ion} is evaluated component-wise and then interpolated, whereas $\boldsymbol{I}_{\text{stim}}(t)$ is obtained by linear interpolation of $I_{\text{stim}}(t, \boldsymbol{x})$. This is commonly done in the cardiac modeling community and called Ionic Current Interpolation (ICI), see [44,45]. Alternatively, one could first evaluate $\boldsymbol{V}_m$, $\boldsymbol{z}_E$, $\boldsymbol{z}_S$ at the quadrature nodes and then evaluate I_{ion} there, an approach known as State Variable Interpolation (SVI). Here, we opt for ICI due to its lower computational cost, as SVI requires interpolating every state variable at the quadrature nodes, which is expensive for ionic models with a large number N of state variables. Finally, the mass matrix $\mathbf{M}$ is always lumped to take advantage of explicit time integrators.

The semi-discrete equation (6) can be written as (1) for $y = (\boldsymbol{V}_m, \boldsymbol{z}_E, \boldsymbol{z}_S)^\top$ by defining

$$f_F(t,y) = \begin{pmatrix} (\chi C_m \mathbf{M})^{-1} \mathbf{A} \boldsymbol{V}_m \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \qquad f_S(t,y) = \begin{pmatrix} C_m^{-1}(\boldsymbol{I}_{stim}(t) - I_{ion}(t, \boldsymbol{V}_m, \boldsymbol{z}_E, \boldsymbol{z}_S)) \\ \mathbf{0} \\ g_S(\boldsymbol{V}_m, \boldsymbol{z}_E, \boldsymbol{z}_S) \end{pmatrix}, \tag{7a}$$

$$f_E(t,y) = \begin{pmatrix} \mathbf{0} \\ g_E(\boldsymbol{V}_m, \boldsymbol{z}_E) \\ \mathbf{0} \end{pmatrix} = \Lambda(y)(y - y_\infty(y)), \tag{7b}$$

with

$$\Lambda(y) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \Lambda_E(\boldsymbol{V}_m) & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad y_\infty(y) = \begin{pmatrix} \mathbf{0} \\ \boldsymbol{z}_{E,\infty}(\boldsymbol{V}_m) \\ \mathbf{0} \end{pmatrix}. \tag{7c}$$

Note that f_F is stiff but very cheap to evaluate, since it only involves a matrix-vector multiplication thanks to mass-lumping. The f_S term, which contains the most complicated terms of the ionic model, is expensive yet non-stiff since the stiff terms of the ionic model are contained in g_E. Clearly, f_E is very stiff but straightforward to integrate exponentially, since $\Lambda(y)$ is diagonal.

The time stepping methods presented in the sequel are not limited to the present spatial discretization. Still, the resulting mass matrix must be diagonal, or very cheap to invert, regardless of the spatial discretization used, to preserve the efficiency of any explicit method. For standard conforming triangular or tetrahedral finite elements, mass-lumping is available up to a certain polynomial order [12,24]. Alternative spatial discretizations that lead to diagonal or block diagonal mass matrices at any order are, for instance, higher order finite elements with orthogonal basis functions, spectral finite element methods [6], finite differences, and discontinuous Galerkin methods [43].

3. Multirate explicit stabilized methods

In this section, we recall the multirate explicit stabilized mRKC method from [5] for

$$y' = f(y) = f_F(y) + f_S(y), \qquad y(0) = y_0, \tag{8}$$

where f_F is fast (stiff) but cheap, while f_S slow (non-stiff or mildly stiff) yet expensive to evaluate.

To simplify the notation, we restrict the discussion to autonomous problems. Still, the methods proposed here also apply to non-autonomous problems while all listed pseudo-codes apply to the general non-autonomous case. For the sake of brevity, we concentrate here on algorithmic aspects of the mRKC method from a practical point of view. First, in Section 3.1, we recall standard explicit stabilized RKC methods without any multirate time-stepping strategy. Then, in Section 3.2, we summarize the mathematical derivation and main properties of the mRKC method. The full mRKC Algorithm is then listed in Section 3.3. Finally, in Section 3.4 we apply the mRKC method to the monodomain model (3).

3.1. Explicit stabilized methods

The coefficients of standard explicit Runge–Kutta (RK) methods for

$$y' = f(y), \qquad y(0) = y_0, \tag{9}$$

are generally optimized for high accuracy. Those high-order methods, however, suffer from severe stability constraints on the time-step. In contrast, explicit stabilized (ES) methods fix the order p (usually $p = 1, 2$) while utilizing an increasing number of stages $s \geq p$ and the corresponding free coefficients to optimize for stability. The resulting stability polynomial, $R_s(z)$, of ES methods then satisfies $|R_s(z)| \leq 1$ for $z \in [-\beta s^2, 0]$, where s is the number of stages and $\beta > 0$ depends on the method. Therefore, the stability domain grows

quadratically with s, and hence with the number of function evaluations, so that ES methods remain efficient without sacrificing explicitness. Moreover, the number of stages s can easily be adapted every time-step to accommodate the problem's instantaneous stiffness, measured in terms of the spectral radius ρ of the right-hand side's Jacobian. Since ES methods are defined *via* a recursive relation, they can be implemented using only three vectors, regardless of s, and are therefore very efficient memory-wise.

Various explicit stabilized methods, such as RKC [59], RKL [41], RKU [50] and ROCK [1,3], are available. Both first- and second-order versions exist for RKC, RKL, and RKU methods, while ROCK methods are second- or fourth-order accurate. Here, we shall only consider the classical first-order RKC method from Algorithm 1, see [2,59,63–66] for further details.

Algorithm 1. One step of the RKC method.

1: **function** RKC_STEP(t_n, y_n, Δt, f, ρ)
2: $\varepsilon = 0.05$
3: $s, \ell_s =$ GET_STAGES(Δt, ρ, ε)
4: $y_{n+1} =$ RKC_ITERATION(t_n, y_n, Δt, f, s, ε)
5: **return** y_{n+1}
6: **function** GET_STAGES(Δt, ρ, ε)
7: $\beta = 2 - 4\varepsilon/3$
8: $s = \lceil \sqrt{\Delta t\, \rho/\beta} \rceil$, $\ell_s = \beta s^2$
9: **return** s, ℓ_s
10: **function** RKC_ITERATION(t, y, Δt, f, s, ε)
11: $\mu_j, \nu_j, \kappa_j, c_j =$ GET_COEFFS(s, ε)
12: $g_0 = y$
13: $g_1 = g_0 + \mu_1 \Delta t f(t, g_0)$
14: **for** $j = 2, \ldots, s$ **do**
15: $g_j = \nu_j g_{j-1} + \kappa_j g_{j-2} + \mu_j \Delta t f(t + c_{j-1}\Delta t, g_{j-1})$
16: **return** g_s
17: **function** GET_COEFFS(s, ε)
18: $\omega_0 = 1 + \varepsilon/s^2$, $\omega_1 = T_s(\omega_0)/T_s'(\omega_0)$
19: $b_j = 1/T_j(\omega_0)$, $j = 0, \ldots, s$
20: $\mu_1 = \omega_1/\omega_0$
21: $c_0 = 0$, $c_1 = \mu_1$
22: **for** $j = 2, \ldots, s$ **do**
23: $\mu_j = 2\omega_1 b_j/b_{j-1}$, $\nu_j = 2\omega_0 b_j/b_{j-1}$, $\kappa_j = -b_j/b_{j-2}$
24: $c_j = \nu_j c_{j-1} + \kappa_j c_{j-2} + \mu_j$
25: **return** $\mu_j, \nu_j, \kappa_j, c_j$

At each time step, the function RKC_STEP of Algorithm 1 computes the approximate solution y_{n+1} at the new time t_{n+1}. Its input arguments are the the solution y_n at the current time t_n, the step size Δt, the right-hand side f, and an approximation ρ to the spectral radius of $\partial f/\partial y(t_n, y_n)$. The spectral radius ρ is either known or estimated using any simple method, such as Gershgorin's theorem ([32], p. 89) or a nonlinear power iteration, as described in Algorithm 5 in Appendix A (see also [37,38,57,58,64]).

In Algorithm 1, we note that the main iteration in Line 15 can be performed using only three distinct vectors for g_{j-2}, g_{j-1}, and g_j. Moreover, at Line 8 the number of stages is proportional to the square root of ρ, instead of ρ, as in standard explicit (RK) methods. The damping parameter $\varepsilon > 0$ guarantees that the stability polynomial $R_s(z)$ satisfies $|R_s(z)| \leq 1 - \varepsilon$ along the negative real axis for $z \in [-\beta s^2, -\delta]$ and small $\delta > 0$. It also slightly extends the stability domain into the imaginary direction thereby ensuring a stable narrow strip along the negative real axis; typically, we set $\varepsilon = 0.05$. The method's coefficients μ, ν, κ, c depend on Chebyshev

polynomials of the first kind, $T_j(x)$. Indeed, the stability polynomial $R_s(z)$ is a shifted and scaled Chebyshev polynomial.

3.2. Modified equation and averaged force

Explicit stabilized methods, such as RKC, certainly alleviate the stringent stability constraint on the time-step due to the stiff term f_F in (8). Nonetheless, the number s of evaluations of the expensive term f_S still depends on the stiffness of f_F, as ρ, and hence s, are still dictated by the stiffness of f_F. To ensure that the number of evaluations of f_S only depends on its own stiffness, and no longer on that of f_F, the evaluation of f_F and f_S must be decoupled.

Following [5], we therefore introduce the modified equation

$$y'_\eta = f_\eta(y_\eta), \qquad y_\eta(0) = y_0, \tag{10}$$

with a free parameter $\eta > 0$. Here, the averaged force $f_\eta : \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$f_\eta(y) = \frac{1}{\eta}(u(\eta) - y), \tag{11}$$

where the auxiliary solution $u(s)$ is defined by the auxiliary equation

$$u' = f_F(u) + f_S(y) \quad s \in [0, \eta], \qquad u(0) = y. \tag{12}$$

We now briefly summarize the main properties of the above modified equation and averaged force; for further details, we refer to [5].

Accuracy. The modified equation (10) is a first-order approximation to (8) with error $\mathcal{O}(\eta)$, since it follows from (11) and (12) that

$$f_\eta(y) = f_S(y) + \frac{1}{\eta}\int_0^\eta f_F(u(s))\,\mathrm{d}s = f_S(y) + f_F(y) + \mathcal{O}(\eta). \tag{13}$$

In fact, equation (13) also implies that f_η is a particular average of $f_F + f_S$ along the auxiliary solution u.

Stability. If the multirate problem (8) is contractive ([31], Chap. IV.12), then, under some assumptions, equation (10) is also contractive and

$$\|y(t) - y_\eta(t)\| \le C(\eta)\int_0^t e^{\mu(t-s)}\|f_F(y(s)) + f_S(y(s))\|\,\mathrm{d}s \tag{14}$$

holds, where $y(t)$ is the exact solution to (8), $\mu < 0$, and $C(\eta) \le 1$ with $C(\eta) = \mathcal{O}(\eta)$ for $\eta \to 0$. Hence, the solutions of (8) and (10) remain close even during long-time integration.

Stiffness. The spectral radius ρ_η of the Jacobian of f_η decreases with increasing η. Hence, for η sufficiently large, it holds

$$\rho_\eta \le \rho_S, \tag{15}$$

where ρ_S if the spectral radius of the Jacobian of f_S. As the stiffness of (10) then depends only on the slow term f_S, the numerical solution of (10) with any explicit method will require fewer right-hand side evaluations than that of (8).

Clearly, there is a trade-off between choosing η sufficiently small to preserve accuracy in (13) and (14) yet sufficiently large to satisfy the stiffness condition (15). Fortunately, both conditions are typically satisfied simultaneously, as (15) is already satisfied for quite small $\eta \le \Delta t$ (often $\eta \ll \Delta t$), where Δt is the step size used to solve the modified equation (10). Then, equation (10) not only is a good approximation to (8) but also much less stiff thanks to (15). Discretization of the modified equation (10) finally leads to the mRKC method, first proposed in [5].

3.3. Multirate explicit stabilized method algorithm

Instead of solving (8) directly, the multirate explicit stabilized (mRKC) method [5] solves the modified problem (10) using an RKC method. In doing so, every evaluation of f_η, $u(\eta)$ is approximated by solving (12) with yet another RKC method. Special care is taken when choosing the number of stages s and m needed for (10) and (12), respectively, and the value of η. For an accuracy and stability analysis of the method, we refer to [5,49].

In Algorithm 2, we list the mRKC method for the numerical solution of (8). The input parameters of MRKC_STEP are t_n, y_n and Δt (as for RKC_STEP), the two right-hand sides f_F, f_S and the spectral radii ρ_F, ρ_S of their respective Jacobians. A single step of Algorithm 2 mainly consists of an outer RKC_ITERATION (Line 6), which solves (10) with the approximate averaged force $\overline{f}_\eta$, and an inner RKC_ITERATION (Line 10), which solves (12) whenever $\overline{f}_\eta$ is needed.

Algorithm 2. One step of the mRKC method.

1: **function** MRKC_STEP(t_n, y_n, Δt, f_F, f_S, ρ_F, ρ_S)
2: s, ℓ_s = GET_STAGES(Δt, ρ_S, 0.05)
3: $\eta = 2\Delta t/\ell_s$ ▷ *For RKC,* $\ell_s = \beta s^2 \approx 2s^2$
4: m, ℓ_m = GET_STAGES(η, ρ_F, 0.05)
5: $\overline{f}_\eta(t, y)$ = AVERAGED_FORCE(t, y, η, f_F, f_S, m) ▷ $\overline{f}_\eta$ *is the approximated averaged force* (11)
6: y_{n+1} = RKC_ITERATION(t_n, y_n, Δt, $\overline{f}_\eta$, s, 0.05) ▷ *This call approximates* (10)
7: **return** y_{n+1}
8: **function** AVERAGED_FORCE(t, y, η, f_F, f_S, m)
9: $f_u(r, u) = f_F(r, u) + f_S(t, y)$ ▷ f_u *is the right-hand side of* (12)*,* f_S *is frozen*
10: u_η = RKC_ITERATION(t, y, η, f_u, m, 0.05) ▷ *This call approximates* (12)
11: **return** $\frac{1}{\eta}(u_\eta - y)$

Since the mRKC method is composed of two embedded RKC iterations, the stabilization process is "multiplicative", in the sense that the number of outer RKC iterations or stages, s, suffices to stabilize the f_S term, whereas the product of outer and inner RKC iterations, $s \times m$, suffices to stabilize the f_F term. This property is reflected in the computational cost.

Computational cost. For each evaluation of $\overline{f}_\eta$, there is a call to the inner loop, which involves one evaluation of f_S and m evaluations of f_F. Since there are s evaluations of $\overline{f}_\eta$ during each time step, there are s evaluations of f_S and $s \times m$ evaluations of f_F. According to Algorithm 2, we compute

$$\#f_S \text{ eval.} = s \approx \sqrt{\frac{\Delta t \rho_S}{\beta}}, \tag{16}$$

$$\#f_F \text{ eval.} = s \times m \approx \sqrt{\frac{\Delta t \rho_S}{\beta}}\sqrt{\frac{\eta \rho_F}{\beta}} = \sqrt{\frac{\Delta t \rho_S}{\beta}}\sqrt{\frac{2\Delta t \rho_F}{\beta^2 s^2}} \approx \sqrt{\frac{2\Delta t \rho_F}{\beta^2}} \approx \sqrt{\frac{\Delta t \rho_F}{\beta}}. \tag{17}$$

Hence, the number of f_S and f_F evaluations depends only on their own inherent stiffness.

The total number of evaluations of f_F is approximately the same as with a standard RKC method, for which the number of stages is given by $\sqrt{\Delta t \rho/\beta}$ with $\rho \approx \rho_F$. In contrast, the number of evaluations of f_S are drastically reduced. Finally, the "multiplicative" property of the mRKC method is clearly apparent in (17), as $s \times m$ satisfies precisely the stability condition $\Delta t \rho_F \approx \beta(sm)^2$ for f_F.

3.4. The mRKC method for the monodomain model

To apply mRKC to the semi-discrete monodomain model (6) from Section 2.1, we must first rewrite it as in (8). The monodomain model (6), however, fits the general form (1), which involves the additional f_E term from the ionic model. In this section, we briefly describe a straightforward but somewhat simplistic and at times inefficient approach to deal with f_E and thus apply mRKC directly to (6). Then, in Section 4, we shall adapt mRKC to differential equations as in (1) by judiciously exploiting the particular structure of f_E, which will yield a more efficient solver for (6). To directly apply mRKC to (6), we first need to absorb the f_E term from the ionic model into f_F and f_S. With y, f_F, f_S, f_E as in (7a), we define

$$\tilde{y}' = \tilde{f}_F(t,\tilde{y}) + \tilde{f}_S(t,\tilde{y}), \tag{18a}$$

with

$$\tilde{f}_F(t,y) = f_F(t,y) + f_E^1(t,y), \qquad \tilde{f}_S(t,y) = f_S(t,y) + f_E^2(t,y). \tag{18b}$$

In doing so, we have split the ionic model as $f_E^1(t,y) + f_E^2(t,y) = f_E(t,y)$, where f_E^1 contains only the stiffest components while f_E^2 contains all the remaining ones. Now, mRKC can be applied to the monodomain model (6) by solving (18).

Depending on the ionic model and the mesh size, the term $\tilde{f}_F$ in (18) is usually stiffer than f_F, but still cheap to evaluate since f_E^1 involves only a few components. In contrast, as f_E^2 is significantly less stiff than f_E^1, the term $\tilde{f}_S$ is only mildly stiff but more expensive to evaluate. Nevertheless, this approach not only remains unsatisfactory, as it does not exploit the special structure of f_E, but also cumbersome to implement because the stiffest gating variables in f_E must be identified. In the next section, we shall introduce a different more efficient mRKC approach, which exploits the particular structure of f_E without the need for splitting the gating variables.

Remark 3.1. For the cardiac models considered here, it turns out that the stiffest components f_E^1 from the ionic model always reduced to a single variable for the entire simulation. To identify that stiffest gating variable, we temporarily set $\tilde{f}_F(t,y) = f_E^1(t,y)$ thereby ignoring diffusion. Next, selecting one gating variable at a time in f_E^1, we run the simulation while evaluating the stiffness of $\tilde{f}_F$ and $\tilde{f}_S$ at each time step with the nonlinear power iteration – see Algorithm 5 in the appendix. The stiffest gating variable is that with the highest spectral radius for $\tilde{f}_F$ over the entire simulation.

4. Exponential multirate explicit stabilized methods

As shown in Section 3.4, the mRKC method can be directly applied to (1) by splitting f_E and absorbing the two terms into f_F and f_S. Still, whenever f_E is stiffer than f_F, it will induce a higher number of stages m in Algorithm 2. To improve on the efficiency, we shall now exploit the special structure (2) of f_E, as in the Rush–Larsen method [53], by using an exponential Euler step to integrate f_E. The combined mRKC/exponential Euler method then yields the emRKC method for (1), and hence for the monodomain equation (6).

4.1. Exponential Euler method

A very efficient method for the numerical solution of

$$y' = f_E(t,y), \qquad y(0) = y_0, \tag{19}$$

with $f_E(y) = \Lambda(y)(y - y_\infty(y))$ as in (2) and $\Lambda(y)$ a diagonal matrix, is the exponential Euler method [33]:

$$y_{n+1} = y_n + \Delta t\, \varphi(\Delta t\, \Lambda(y_n)) f_E(y_n), \tag{20}$$

where

$$\varphi(z) = \frac{e^z - 1}{z} \tag{21}$$

is an exponential-like function. In contrast to the explicit Euler method, $y_{n+1} = y_n + \Delta t\, f_E(y_n)$, the exponential Euler method employs the matrix exponential $\varphi(\Delta t\, \Lambda(y_n))$, which dampens all large eigenvalues (in magnitude) and stabilizes the discrete dynamical system. Hence, the exponential Euler method is not only unconditionally stable but here also relatively cheap because $\Lambda(y)$ is diagonal. Higher-order variants of this method exist, we refer to [33] for a review article.

4.2. Exponential Euler step for the auxiliary equation

Instead of splitting f_E, as in Section 3.4, we could also entirely include it into the "fast" term f_F. Hence, the auxiliary equation (12) would read

$$u' = f_F(u) + f_S(y) + f_E(u) \quad t \in [0, \eta], \qquad u(0) = y. \tag{22}$$

As in Section 3.4, however, the cost would again increase whenever f_E is stiffer than f_F.

To design an explicit multirate method for (1), which avoids any further stability restriction due to f_E while taking advantage of its particular diagonal structure, we now modify the auxiliary problem (22) by taking inspiration from splitting methods as follows: first, advance the f_E term with the exponential Euler method and then integrate the remaining terms. This yields the split auxiliary equation:

$$y_E = y + \eta\, \varphi(\eta \Lambda(y)) f_E(y), \tag{23a}$$

$$u' = f_F(u) + f_S(y_E) \quad t \in [0, \eta], \qquad u(0) = y_E. \tag{23b}$$

In contrast to (22), the stiffness of (23b) is independent from the f_E term; hence, approximating u with any explicit method is cheaper.

The emRKC method is derived similarly to mRKC though by considering the split auxiliary equation (23) instead of (22). Hence, emRKC also solves the modified equation (10) instead of the original problem (1) and the definition (11) of the averaged force f_η remains identical, except that $u(\eta)$ in (11) is now computed from (23).

Remark 4.1. Alternatively, we may also split the auxiliary equation with y_E as in (23a) yet with (23b) replaced by $u' = f_F(u) + f_S(y_E) + \varphi(\eta\Lambda(y)) f_E(y)$ with $u(0) = y$; hence, the contribution of $\varphi(\eta\Lambda(y)) f_E(y)$ is added progressively during the integration. In our numerical experiments, both splittings led to a similar performance.

4.3. Pseudo-code for the exponential multirate explicit stabilized method

In Algorithm 3, we list a pseudo-code for the emRKC method. Its main differences with respect to Algorithm 2 are: first, the additional input argument, f_E, for emRKC_Step and Averaged_Force; second, the redefinition of the f_u term at Lines 9 and 10; third, the different input argument in Line 11. To avoid round-off errors due to small diagonal values in $\Lambda(y)$ or small η, the computation of the exponential Euler step is done employing the identity indicated in Line 14.

4.4. Stability analysis

Here we analyse the stability of the emRKC method starting from previous results for mRKC. In particular, we show that there are no constraints on the step size Δt for sufficiently many internal stages s, m. Hence, we consider the standard linear scalar test equation,

$$y' = \lambda_F y + \lambda_S y + \lambda_E y, \qquad y(0) = y_0, \tag{24}$$

where $\lambda_F, \lambda_S, \lambda_E \leq 0$ correspond to f_F, f_S, f_E, respectively. Before tackling the analysis for the emRKC method, we briefly recall the stability analysis for the mRKC method [5, 49].

Algorithm 3. One step of the emRKC method.

function emRKC_Step(t_n, y_n, Δt, f_F, f_S, f_E, ρ_F, ρ_S)
  $s,\ \ell_s = \text{GET\_STAGES}(\Delta t,\ \rho_S,\ 0.05)$
  $\eta = 2\Delta t/\ell_s$
  $m, \ell_m = \text{GET\_STAGES}(\eta,\ \rho_F,\ 0.05)$
  $\overline{f}_\eta(t,y) = \text{AVERAGED\_FORCE}(t,\ y,\ \eta,\ f_F,\ f_S,\ f_E,\ m)$
  $y_{n+1} = \text{RKC\_ITERATION}(t_n,\ y_n,\ \Delta t,\ \overline{f}_\eta,\ s,\ 0.05)$
  return y_{n+1}
function AVERAGED_FORCE(t, y, η, f_F, f_S, f_E, m)
  $y_E = \text{EXPONENTIAL\_EULER}(y,\ \eta,\ \Lambda(y),\ y_\infty(y))$
  $f_u(r,u) = f_F(r,u) + f_S(t,y_E)$
  $u_\eta = \text{RKC\_ITERATION}(t,\ y_E,\ \eta,\ f_u,\ m,\ 0.05)$
  return $\frac{1}{\eta}(u_\eta - y)$
function EXPONENTIAL_EULER(y, η, Λ, y_∞)
  return $y + (e^{\eta\Lambda} - I)(y - y_\infty)$ ▷ $\eta\varphi(\eta\Lambda)\Lambda(y - y_\infty) = (e^{\eta\Lambda} - I)(y - y_\infty)$

4.4.1. Stability analysis for the mRKC method

The stability analysis of the mRKC scheme [5,49] also relies on (24) yet with $\lambda_E = 0$; hence, we consider the test equation

$$y' = \lambda_F y + \lambda_S y, \qquad y(0) = y_0. \tag{25}$$

First, we recall a key result from [5], crucial for proving the stability of the mRKC method when applied to the test equation (25).

Lemma 4.1. *Let $P_m(z)$ be the stability polynomial of the inner RKC iteration of mRKC (Line 10 of Algorithm 2). Let $\Phi_m(z) = (P_m(z) - 1)/z$. If*

$$-\ell_s \le \Delta t\, \Phi_m(\eta\lambda_F)(\lambda_F + \lambda_S) \le 0 \tag{26}$$

holds, the mRKC method applied to the test equation (25) is stable.

Sketch of proof. Since mRKC is equivalent to an s-stage RKC method applied to the averaged force $\overline{f}_\eta(y)$ (Line 6 of Algorithm 2), we conclude that mRKC is stable if

$$-\ell_s \le \Delta t \frac{\partial \overline{f}_\eta}{\partial y}(y) \le 0. \tag{27}$$

To complete the proof, one shows that $\overline{f}_\eta(y) = \Phi_m(\eta\lambda_F)(\lambda_F + \lambda_S)y$ when mRKC is applied to the test equation (25); thus, condition (26) is equivalent to (27). □

In [49], the conditions on s, m, η to satisfy (26) were studied in two different scenarios. In the first case, λ_F, λ_S can take any non-positive values and general stability conditions on s, m, η are derived. In the second case, it is assumed that $\lambda_F \ll \lambda_S \le 0$, thus yielding milder stability conditions. We recall these conditions in Lemma 4.2 below.

Lemma 4.2. *Let $\rho_F = |\lambda_F|$ and $\rho_S = |\lambda_S|$, $\ell_s = \beta s^2$ and $\ell_m = \beta m^2$. Define the hypotheses:*

H1 : $\lambda_F, \lambda_S \le 0$ and s, m, η such that

$$\Delta t\, \rho_S \le \ell_s, \qquad \eta\, \rho_F \le \ell_m, \qquad \eta = \frac{6\Delta t}{\ell_s}\frac{m^2}{m^2 - 1}. \tag{28}$$

H2 : $\lambda_F \ll \lambda_S \leq 0$ *and* s, m, η *such that*

$$\Delta t\, \rho_S \leq \ell_s, \qquad \eta\, \rho_F \leq \ell_m, \qquad \eta = \frac{2\Delta t}{\ell_s}. \tag{29}$$

If either H1 or H2 are true, then (26) is satisfied.

For the proof of this lemma and more details on the condition $\lambda_F \ll \lambda_S \leq 0$[1] we refer to [49]. By combining Lemmas 4.1 and 4.2, we eventually prove the stability of the mRKC method for the test equation.

Theorem 4.3. *If either hypothesis H1 or H2 in Lemma 4.2 is satisfied, the mRKC method applied to the test equation (25) is stable.*

Proof. Follows from Lemmas 4.1 and 4.2. □

Note that H2 places stronger restrictions on λ_F, λ_S, but also yields a smaller η than H1, and hence a smaller number of stages m which results in fewer f_F function evaluations and therefore a more efficient algorithm. For the context of this work, it makes sense to assume that H2 is satisfied, hence $\lambda_F \ll \lambda_S \leq 0$, which stands for f_F significantly stiffer than f_S. Indeed, it was found in [5], that in general for systems of ODEs stemming from the spatial discretization of parabolic PDEs, the stability conditions (29) yield stable solutions. Note that (29) are the constraints encoded in Algorithms 2 and 3.

We also remark that there is no constraint on the step size Δt. Indeed, given ρ_F, ρ_S, one can always choose Δt according to the desired accuracy. Stability is then achieved by setting s sufficiently large to ensure $\Delta t \rho_S \leq \ell_s$ and finally computing η and m.

4.4.2. Stability analysis for the emRKC method

The stability analysis of the emRKC method follows from that for the mRKC method. In particular, we find that the same stability conditions hold and therefore there are no constraints on the step size Δt.

Theorem 4.4. *If eithe hypothesis H1 or H2 in Lemma 4.2 is satisfied, the emRKC method applied to the test equation (24) is stable.*

Proof. Following [5], we deduce for emRKC applied to (24) that

$$u_\eta = (P_m(\eta\lambda_F) + \Phi_m(\eta\lambda_F)\eta\lambda_S)y_E = (P_m(\eta\lambda_F) + \Phi_m(\eta\lambda_F)\eta\lambda_S)e^{\eta\lambda_E}y. \tag{30}$$

The averaged force $\overline{f}_\eta(y)$ is therefore given by

$$\begin{aligned} \overline{f}_\eta(y) &= \frac{1}{\eta}(u_\eta - y) = \frac{1}{\eta}\big((P_m(\eta\lambda_F) + \Phi_m(\eta\lambda_F)\eta\lambda_S)e^{\eta\lambda_E} - 1\big)y \\ &= \frac{1}{\eta}\big((P_m(\eta\lambda_F) - 1 + \Phi_m(\eta\lambda_F)\eta\lambda_S)e^{\eta\lambda_E} + e^{\eta\lambda_E} - 1\big)y \\ &= \big(\Phi_m(\eta\lambda_F)(\lambda_F + \lambda_S)e^{\eta\lambda_E} + \varphi(\eta\lambda_E)\lambda_E\big)y. \end{aligned} \tag{31}$$

Since emRKC is an s-stage RKC method applied to $\overline{f}_\eta(y)$ (Line 6 of Algorithm 3), it is stable if

$$-\ell_s \leq \Delta t \frac{\partial \overline{f}_\eta}{\partial y}(y) \leq 0. \tag{32}$$

[1] Essentially we need $\lambda_F \leq 4\lambda_S$.

By using $\varphi(z) \geq 0$, $\lambda_E \leq 0$ and (26), we obtain

$$\begin{aligned} 0 \geq \Delta t \frac{\partial \overline{f}_\eta}{\partial y}(y) &= \Delta t\big(\Phi_m(\eta\lambda_F)(\lambda_F+\lambda_S)e^{\eta\lambda_E} + \varphi(\eta\lambda_E)\lambda_E\big) \\ &\geq -\ell_s e^{\eta\lambda_E} + \frac{\Delta t}{\eta}(e^{\eta\lambda_E}-1) = -\ell_s + (e^{\eta\lambda_E}-1)\left(\frac{\Delta t}{\eta} - \ell_s\right) \geq -\ell_s. \end{aligned} \tag{33}$$

The last inequality follows from $e^{\eta\lambda_E} - 1 \leq 0$ and the fact that $\frac{\Delta t}{\eta} - \ell_s \leq 0$ holds for both choices of η in (28) or (29). □

Note that λ_E does not impact the choice of s,m, or η due to its exponential integration.

4.5. Accuracy analysis

To prove that the emRKC method is first-order accurate, we assume that f_F, f_S, f_E are sufficiently smooth. Furthermore, for simplicity of presentation, we restrict ourselves to the autonomous case when f_F, f_S, f_E are time independent.

Theorem 4.5. *The emRKC method is first-order accurate.*

Proof. We start by computing a Taylor expansion of the exact auxiliary solution $u(\eta)$ defined by the split auxiliary equation (23). We have $y_E = y + \eta f_E(y) + \mathcal{O}(\eta^2)$ and $u(s) = u(0) + \mathcal{O}(s)$. Since $u(0) = y_E$, we thus obtain

$$\begin{aligned} u(\eta) &= u(0) + \int_0^\eta u'(s)\,\mathrm{d}s = y_E + \int_0^\eta f_F(u(s)) + f_S(y_E)\,\mathrm{d}s = y_E + \int_0^\eta f_F(y_E) + f_S(y_E) + \mathcal{O}(s)\,\mathrm{d}s \\ &= y_E + \eta f_F(y_E) + \eta f_S(y_E) + \mathcal{O}(\eta^2) = y + \eta(f_F(y) + f_S(y) + f_E(y)) + \mathcal{O}(\eta^2). \end{aligned} \tag{34}$$

Since the inner RKC method (Line 11 of Algorithm 3) used to approximate the integral in (34) is first-order accurate, it introduces a local error of $\mathcal{O}(\eta^2)$; hence, the numerical auxiliary solution u_η satisfies

$$u_\eta = u(\eta) + \mathcal{O}(\eta^2) = y + \eta(f_F(y) + f_S(y) + f_E(y)) + \mathcal{O}(\eta^2). \tag{35}$$

Therefore, the numerical averaged force $\overline{f}_\eta(y)$ computed in Algorithm 3 satisfies

$$\overline{f}_\eta(y) = \frac{1}{\eta}(u_\eta - y) = f_F(y) + f_S(y) + f_E(y) + \mathcal{O}(\eta). \tag{36}$$

Hence it is a first-order approximation in η of the exact right-hand side. At any time step given y_n, the outer RKC method (Line 6 of Algorithm 3) computes y_{n+1} as

$$\begin{aligned} g_0 &= y_n, \quad g_1 = g_0 + \mu_1 \Delta t \overline{f}_\eta(g_0), \\ g_j &= \nu_j g_{j-1} + \kappa_j g_{j-2} + \mu_j \Delta t \overline{f}_\eta(g_{j-1}), \quad j = 2, \ldots, s, \\ y_{n+1} &= g_s. \end{aligned} \tag{37}$$

By setting $d_j = g_j - y_n$ and using $\nu_j + \kappa_j = 1$ we can rewrite (37) as

$$\begin{aligned} d_0 &= 0, \quad d_1 = \mu_1 \Delta t \overline{f}_\eta(t_n, y_n) \\ d_j &= \nu_j d_{j-1} + \kappa_j d_{j-2} + \mu_j \Delta t \overline{f}_\eta(y_n + d_{j-1}), \quad j = 2, \ldots, s, \\ y_{n+1} &= y_n + d_s. \end{aligned} \tag{38}$$

From Lemma 3.1 of [16] with $A = 0$ and $r_j = \mu_j \Delta t \overline{f}_\eta(y_n + d_{j-1})$ we obtain

$$y_{n+1} = y_n + \sum_{j=1}^{s} \frac{b_s}{b_j} U_{s-j}(\omega_0) r_j = y_n + \Delta t \sum_{j=1}^{s} \frac{b_s}{b_j} U_{s-j}(\omega_0) \mu_j \overline{f}_\eta(y_n + d_{j-1}), \tag{39}$$

where the b_j are as in Algorithm 1 and $U_j(x)$ denotes the Chebyshev polynomial of the second kind of degree j. Since $d_j = \mathcal{O}(\Delta t)$ (shown recursively) it follows

$$y_{n+1} = y_n + \Delta t \sum_{j=1}^{s} \frac{b_s}{b_j} U_{s-j}(\omega_0) \mu_j \big(\overline{f}_\eta(y_n) + \mathcal{O}(\Delta t)\big) = y_n + \Delta t \overline{f}_\eta(y_n) + \mathcal{O}\big(\Delta t^2\big), \tag{40}$$

where the last equality follows from the identity $\sum_{j=1}^{s} \frac{b_s}{b_j} U_{s-j}(\omega_0)\mu_j = 1$ (see [16]). Finally, we use (40) in (36) to obtain

$$y_{n+1} = y_n + \Delta t(f_F(y_n) + f_S(y_n) + f_E(y_n)) + \mathcal{O}(\Delta t^2 + \Delta t \eta). \tag{41}$$

Since $\eta = \mathcal{O}(\Delta t)$, the local error is $\mathcal{O}(\Delta t^2)$ and the method is first-order accurate. □

5. Numerical experiments

In this section, we apply the mRKC and emRKC methods to the monodomain model (3) and compare them with the popular implicit-explicit Rush–Larsen (IMEX-RL) method described below. Their performance depends on a number of factors, such as the mesh size and regularity, the nonlinearity, the stiffness, and the number of ionic state variables; moreover, the performance of the IMEX-RL method also depends on the type of solver used during each time step. To take into account all these factors, we perform numerical experiments either with structured or unstructured meshes for different mesh sizes, while considering ionic models of varying size and stiffness, both in two and three space dimensions thereby including both direct and iterative linear solvers into the comparison.

More specifically, we first assess in Section 5.1 the effective stability properties of the emRKC method when compared with the theoretical results of Section 4.4. Then, in Section 5.2 we perform a two-dimensional experiment on a structured mesh, where we compare the various methods for different ionic models and mesh sizes. In Section 5.3, we repeat the experiment yet for a three-dimensional structured mesh; hence, an iterative solver is used for the linear systems needed in the IMEX-RL method. Next in Section 5.4, we consider a more realistic three-dimensional geometry of the left atrium with an unstructured mesh. Again we repeat the numerical experiment for the same geometry but in the presence of unhealthy fibrotic tissue when the diffusion tensor is discontinuous. We conclude in Section 5.6 with a scalability experiment.

The implicit-explicit-Rush–Larsen method. The well-known IMEX-RL method for the monodomain model will be used as a benchmark to assess the performance of mRKC and emRKC. It is based on an IMEX scheme for (6a) and the Rush–Larsen approach [53] for the ionic model (6b) and (6c); hence, $\boldsymbol{z}_S$ is integrated explicitly whereas the gating variables $\boldsymbol{z}_E$ *via* an exponential integrator.

The IMEX-RL method proceeds as follows. First, it solves the ODE subsystem (6b) and (6c) with the Rush–Larsen method [53]. Hence, at the beginning of each time step, the method freezes the electric potential $\boldsymbol{V}_m$ and solves (6b) with the exponential Euler method. Then it solves (6c) with the explicit Euler method. Finally, it solves (6a) by treating the linear term implicitly and the remaining terms explicitly. A pseudo-code for the resulting IMEX-RL method is provided in Algorithm 4.

Since the stiffest terms $g_E(\boldsymbol{V}_m, \boldsymbol{z}_E)$ and $\mathbf{A}\boldsymbol{V}_m$ are treated exponentially and implicitly, respectively, the scheme only has a mild stability constraint. It is also quite efficient because $\Lambda_E(\boldsymbol{V}_m)$ is diagonal and the implicit step is linear in the unknown.

Algorithm 4. One step of the IMEX-RL method.

1: **function** IMEX-RL_STEP(t_n, $\boldsymbol{V}_{m,n}$, $\boldsymbol{z}_{E,n}$, $\boldsymbol{z}_{S,n}$, Δt)
2: $\quad \boldsymbol{z}_{E,n+1} = \boldsymbol{z}_{E,n} + (e^{\Delta t \Lambda_E(\boldsymbol{V}_{m,n})} - I)(\boldsymbol{z}_{E,n} - \boldsymbol{z}_{E,\infty}(\boldsymbol{V}_{m,n}))$
3: $\quad \boldsymbol{z}_{S,n+1} = \boldsymbol{z}_{S,n} + \Delta t g_S(\boldsymbol{V}_{m,n}, \boldsymbol{z}_{E,n+1}, \boldsymbol{z}_{S,n})$
4: $\quad$ Solve for $\boldsymbol{V}_{m,n+1}$: $C_m \mathbf{M} \frac{\boldsymbol{V}_{m,n+1} - \boldsymbol{V}_{m,n}}{\Delta t} = \chi^{-}1 \mathbf{A} \boldsymbol{V}_{m,n+1} + \mathbf{M} \boldsymbol{I}_{stim}(t_n) - \mathbf{M} I_{ion}(t_n, \boldsymbol{V}_{m,n}, \boldsymbol{z}_{E,n+1}, \boldsymbol{z}_{S,n+1})$.
5: $\quad$ **return** $\boldsymbol{V}_{m,n+1}$, $\boldsymbol{z}_{E,n+1}$, $\boldsymbol{z}_{S,n+1}$

TABLE 2. Ionic models' size and stiffness, with N the number of state variables and $\rho_{\max}$ the maximal spectral radius of the right-hand side's Jacobian in (6).

	HH	CRN	TTP
N	3	20	18
$\rho_{\max}$	40	130	1000

Computational setup. The monodomain model (3) is implemented in Python employing the FEniCSx [8,55, 56] library, where, under the hood, all expensive computations are performed in C/C++. In particular, all linear algebra operations are performed through the PETSc [9,17] library. As ionic models should not be implemented by hand, due to their high complexity, their definitions were obtained from the CellML [40] model repository; in doing so, we adapted the C code generated by the Myokit [11] library and created wrappers to call it from Python. Numerical experiments of Section 5.2 are performed on our workstation, while those of Sections 5.3–5.6 on the Piz Daint supercomputer at the Swiss National Supercomputing Centre (CSCS).

The mRKC method always incorporates the splitting of the gating variables described in (18b) – see also Remark 3.1. In contrast, the emRKC method employs the same splitting as the IMEX-RL method, which is the usual one used in the cardiac community.

Unless stated otherwise, the initial values for $\boldsymbol{V}_m$, $\boldsymbol{z}_E$, $\boldsymbol{z}_S$ are provided by the ionic model's definition and are uniformly set throughout the computational domain. Here we shall consider the Hodgkin–Huxley (HH) [34], the Courtemanche–Ramirez–Nattel (CRN) [15] and the ten Tusscher–Panfilov (TTP) [62] model. Thus our numerical experiments span a wide range of situations from the small nonstiff HH model, to the larger and mildly stiff CRN model, and even to the large and very stiff TTP model, see Table 2 and [60,61]. In all numerical experiments, we use the parameter values specified in Table 1. Finally, the vector fields $\boldsymbol{a}_l(\boldsymbol{x}), \boldsymbol{a}_t(\boldsymbol{x}), \boldsymbol{a}_n(\boldsymbol{x})$ (see (4)), which define the fibers' orientation in Section 5.4, are generated with lifex-fiber [7,47]. The applied stimulus $\boldsymbol{I}_{\text{stim}}(t)$ is specified individually for each experiment.

The linear systems needed in the IMEX-RL method are solved differently in two or three space dimensions: in 2-D we use the Cholesky factorization, and in 3-D the conjugate gradient method preconditioned using the algebraic multigrid library hypre [19]. For mRKC and emRKC, the spectral radii ρ_F, ρ_S are computed only once at the beginning of the simulation employing Algorithm 5 in Appendix A.

During the numerical experiments, we always monitor the relative error in the potential $\boldsymbol{V}$ at the final time with respect to the $L^2(\Omega)$-norm:

$$\text{rel. } L^2\text{-err.} = \frac{\|\boldsymbol{V}^\star - \boldsymbol{V}_N\|_{L^2(\Omega)}}{\|\boldsymbol{V}^\star\|_{L^2(\Omega)}}, \tag{42}$$

where $\boldsymbol{V}_N$ is the numerical solution obtained with either one of the IMEX-RL, mRKC, emRKC methods and $\boldsymbol{V}^\star$ is a reference solution to (6) computed with a much smaller time-step $\Delta t = 10^{-4}$ ms using a method similar to IMEX-RL but fully explicit, *i.e.*, Line 4 in Algorithm 4 is performed explicitly; we denote this method EXEX-RL. The final time is chosen such that a propagating wave is located inside the domain and the solution is still far from equilibrium.

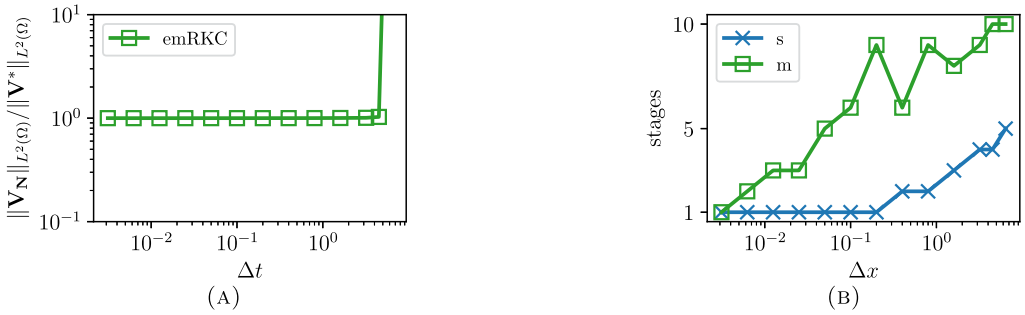


FIGURE 1. Effective stability. Relative norm of the numerical solution $\mathbf{V}_N$ and number of stages for increasing step size Δt. Recall that s, m are the number of stages used by the outer and inner RKC iterations, respectively. (a) Relative norm of $\mathbf{V}_N$ as Δt increases. (b) Number of stages s, m as Δt increases.

5.1. Practical stability limit

In Section 4.4 we proved for a scalar linear test equation that the emRKC method is stable for all step sizes $\Delta t > 0$ if the numbers of stages s, m are sufficiently large and satisfy the stability condition (29). In this first numerical experiment, we study the effective stability properties of emRKC in a more realistic setting.

To assess the practical stability properties of the emRKC method, we solve the monodomain equation (3) with increasing step size Δt and verify that the norm of the solution remains bounded, ignoring accuracy here. To push the method to its stability limit, we choose a setting where the stiffness of f_F and f_E is severe, *i.e.*, we choose a relatively small mesh size $\Delta x = 0.025\,\text{mm}$ in a one dimensional domain $\Omega = [0, 20]\,\text{mm}$ and employ the TTP ionic model. Although the stiffness of f_S depends on the ionic model, too, it usually does only mildly so. We integrate the monodomain equation up to the final time $T = 100\,\text{ms}$ with an applied stimulus $\chi I_{\text{stim}}(t, \boldsymbol{x}) = 50\,\mu\text{A}\,\text{mm}^{-3}$ for $\boldsymbol{x} \in [0, 1.5]\,\text{mm}$ and $t \leq 2\,\text{ms}$, and else $\chi I_{\text{stim}}(t, \boldsymbol{x}) = 0$.

In Figure 1a we display the relative norm of the numerical solution $\mathbf{V}_N$ with respect to a reference solution $\mathbf{V}^*$ as a function of the step size Δt. We see that the method remains stable up to $\Delta t \leq 4.5\,\text{ms}$. In contrast, a standard explicit scheme, such as the EXEX-RL method described at the beginning of Section 5 (*i.e.*, as in Algorithm 4 but performing Line 4 explicitly), remains stable only up to $\Delta t \leq 0.003\,\text{ms}$. In Figure 1b we show the number of stages s, m taken by emRKC. We observe that $m > 1$ already for small Δt, indicating that the diffusion f_F has to be stabilized very soon by the inner iterations of emRKC. The outer stages are needed to stabilize the non stiff term f_S, as indicated by $s > 1$ only for relatively large Δt. The "V" shaped behavior of m is typical and further discussed in the next Section 5.2.

In Section 4.4, we proved that for the test equation (24) the step size Δt can be taken arbitrarily large, as the emRKC method remains stable as long as the number of stages s, m are sufficiently large and satisfy the stability condition (29). In contrast, as shown in the current experiment, due to nonlinearity the maximal step size is bounded in practice. Nevertheless, this practical bound is remarkably large, $\Delta t \leq 4.5\,\text{ms}$ when compared to the stability bound for a standard explicit method, where $\Delta t \leq 0.003\,\text{ms}$, thus yielding a maximal step size increase by a factor of 1500! In that sense, we claim that in practice emRKC has no stability restriction on the step size, since we rarely use $\Delta t > 0.1\,\text{ms}$ because of accuracy requirements.

5.2. Two-dimensional rectangle

First, we consider the two-dimensional version of the benchmark problem proposed in [42], hence (3) with $\Omega = [0, 20] \times [0, 7]\,\text{mm}^2$ and the final time $T = 25\,\text{ms}$. To initiate the propagation of the action potential, the applied stimulus is defined as $\chi I_{\text{stim}}(t, \boldsymbol{x}) = 50\,\mu\text{A}\,\text{mm}^{-3}$ if $t \leq 2\,\text{ms}$ and $\|\boldsymbol{x}\|_\infty \leq 1.5\,\text{mm}$, else $\chi I_{\text{stim}}(t, \boldsymbol{x}) = 0$.

To begin we assess the accuracy of the various time integration methods, in particular we verify that the first-order of convergence predicted by Theorem 4.5 also holds in practice. More precisely, for each mesh size $\Delta x = 0.2\,\text{mm}$, $0.1\,\text{mm}$, $0.05\,\text{mm}$ and every ionic model HH, CRN, TTP, we successively reduce the step size as $\Delta t_i = 2^{-i}\,\text{ms}$ for $i = 0, \ldots, 9$, *i.e.*, from $1\,\text{ms}$ down to approximately $0.002\,\text{ms}$. As shown in Figure 2, all methods achieve the expected first-order convergence. Moreover, in all but one case, the emRKC is equally or even more accurate than the standard IMEX-RL method.

Nonetheless, in Figures 2a, 2d and 2g we also notice that some data points for large Δt are missing for the mRKC and emRKC methods, indicating instability, yet only for the HH model and not for the more realistic CRN and TTP models. We believe that instabilities in the HH model are due to the overly large imaginary part of the eigenvalues of the right-hand side's Jacobian, compared to the more realistic CRN and TTP models whose eigenvalues have relatively small imaginary parts [60, 61]. In fact, those instabilities become even more pronounced as the mesh size decreases, probably because the stability of mRKC and emRKC for non-real eigenvalues deteriorates as f_F becomes increasingly stiffer; this effect, however, is not well understood yet, since a stability analysis of mRKC and emRKC with complex eigenvalues is still lacking.

In Figure 3, we display the number of stages s, m used by the mRKC and emRKC methods. For the CRN and TTP models, a closer look reveals that both tend to lose some accuracy when the number of inner stages m increases. As expected, the number of outer stages s typically remains the same for both methods, since s depends on f_S for emRKC but on $f_S + f_E^2$ for mRKC, yet with f_E^2 much less stiff than f_S. The number of inner stages m for emRKC depends only on the diffusion term f_F, whereas for mRKC it depends on $f_F + f_E^1$. In Figure 3, we also observe that the number of stages m for the two methods often differ, thus indicating that f_E^1 is indeed stiffer than the diffusion f_F. In those instances, the efficiency of mRKC deteriorates against emRKC. We also observe for mRKC that m depends not only on the ionic model but also on the mesh size, while for emRKC it depends only on the mesh size. Finally, the "V" shape behavior in the graph of m apparent at larger step sizes stems from the "multiplicative" stabilization procedure of the methods, as discussed in Section 3. Indeed, for decreasing Δt, a constant number of outer stages s also implies a decreasing number of inner stages m. In contrast, for decreasing s, the number of inner stages m can slightly increase to compensate for the weaker stabilization of the outer loop. Still, $s \times m$ always decreases with Δt.

Finally, in Figure 4 we display work *vs.* accuracy by monitoring the L^2-norm relative errors against the total CPU time. The emRKC method tends to be the most efficient scheme except for one instance, while the mRKC method also performs reasonably well. The high performance of emRKC compared to the other methods stems from a number of factors. First, in comparison to IMEX-RL, the explicit emRKC method is cheaper and tends to be more accurate because the f_S, f_E terms take part in the stabilization procedure for the diffusion term f_F (Line 11 of Algorithm 3); thus, a strong coupling between the different model's components is guaranteed (not so in the splitting strategy of IMEX-RL). Second, in comparison to mRKC, emRKC is cheaper due to the smaller number of inner stages m and has higher accuracy because it employs the more accurate exponential integrator for the gating variables.

5.3. Three-dimensional cuboid

Next, we perform a series of numerical experiments very similar to those in Section 5.2 but in three dimensions. Hence, we set $\Omega = [0, 20] \times [0, 7] \times [0, 3]\,\text{mm}^3$ while the applied stimulus and final simulation time remain identical. Again all methods achieved first-order convergence in time for all three ionic models – those plots are omitted here for the sake of brevity. We also omit the graphs of the number of stages taken by mRKC and emRKC, which are identical to the two-dimensional case.

The work *vs.* accuracy diagrams, shown in Figure 5, again confirm the high efficiency of emRKC. Still, it tends to be slower than in 2-D probably because the inner iterations are more expensive when the diffusion matrix stems from a three-dimensional problem. Again we note that the mRKC and emRKC schemes are stable for the CRN and TTP models, even at the largest step size $\Delta t = 1\,\text{ms}$, but not for the HH model.

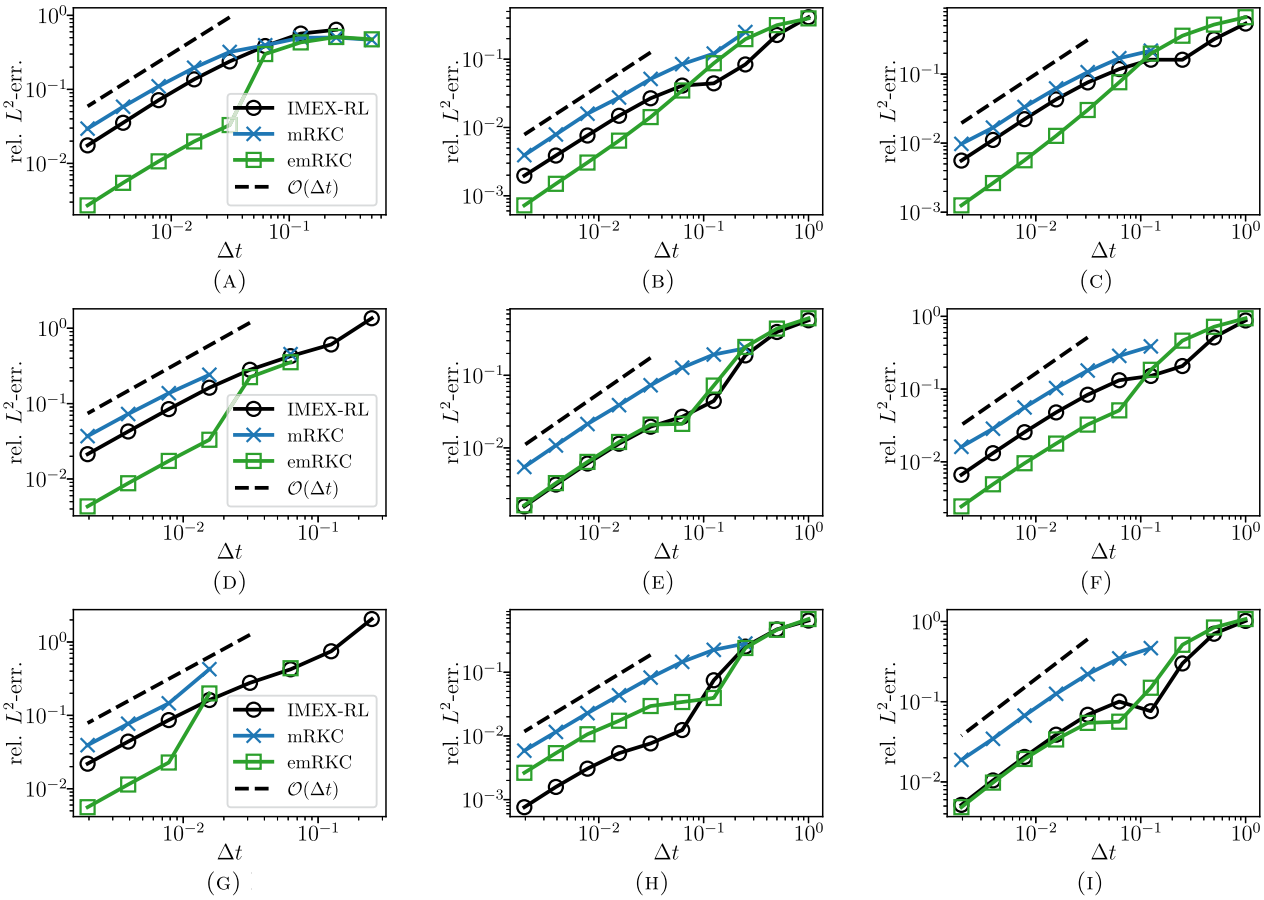


FIGURE 2. Two dimensional rectangle. Convergence experiments with different mesh sizes Δx and ionic models. (a) HH model, $\Delta x = 0.2\,\text{mm}$. (b) CRN model, $\Delta x = 0.2\,\text{mm}$. (c) TTP model, $\Delta x = 0.2\,\text{mm}$. (d) HH model, $\Delta x = 0.1\,\text{mm}$. (e) CRN model, $\Delta x = 0.1\,\text{mm}$. (f) TTP model, $\Delta x = 0.1\,\text{mm}$. (g) HH model, $\Delta x = 0.05\,\text{mm}$. (h) CRN model, $\Delta x = 0.05\,\text{mm}$. (i) TTP model, $\Delta x = 0.05\,\text{mm}$.

5.4. Left atrium geometry

Here we compare the performance of the mRKC and emRKC methods against the IMEX-RL benchmark but now consider a realistic left atrium geometry. The computational mesh was obtained by refining twice the realistic mesh used in [20, 21], which led to an average mesh size of 0.26 mm, maximal mesh size of 0.69 mm, minimal mesh of size 0.026 mm, and about 1.7 million degrees of freedom for the FE-solution. The vector fields $\boldsymbol{a}_l(\boldsymbol{x})$ and $\boldsymbol{a}_t(\boldsymbol{x})$ which define the fiber and sheet orientation in (4), have been generated with the lifex-fiber library [7, 47]. The initial value, shown in Figure 6a, is no longer constant throughout Ω but instead generated by solving the monodomain equation for $t \in [0, 800]$ ms and applying the same stimulus-pacing protocol as in [23, 54] in a fixed region of the domain.

As in Sections 5.2 and 5.3, we perform convergence and efficiency experiments for the three ionic models HH, CRN, and TTP. The final time is set to $T = 50\,\text{ms}$, see Figure 6 for an illustration of the solution at $t = T$. Here, the applied stimulus is identically zero, as the initial condition already contains a propagating wave. Again, all three methods achieve first-order convergence in time for all three models, as shown in Figure 7. Unlike the

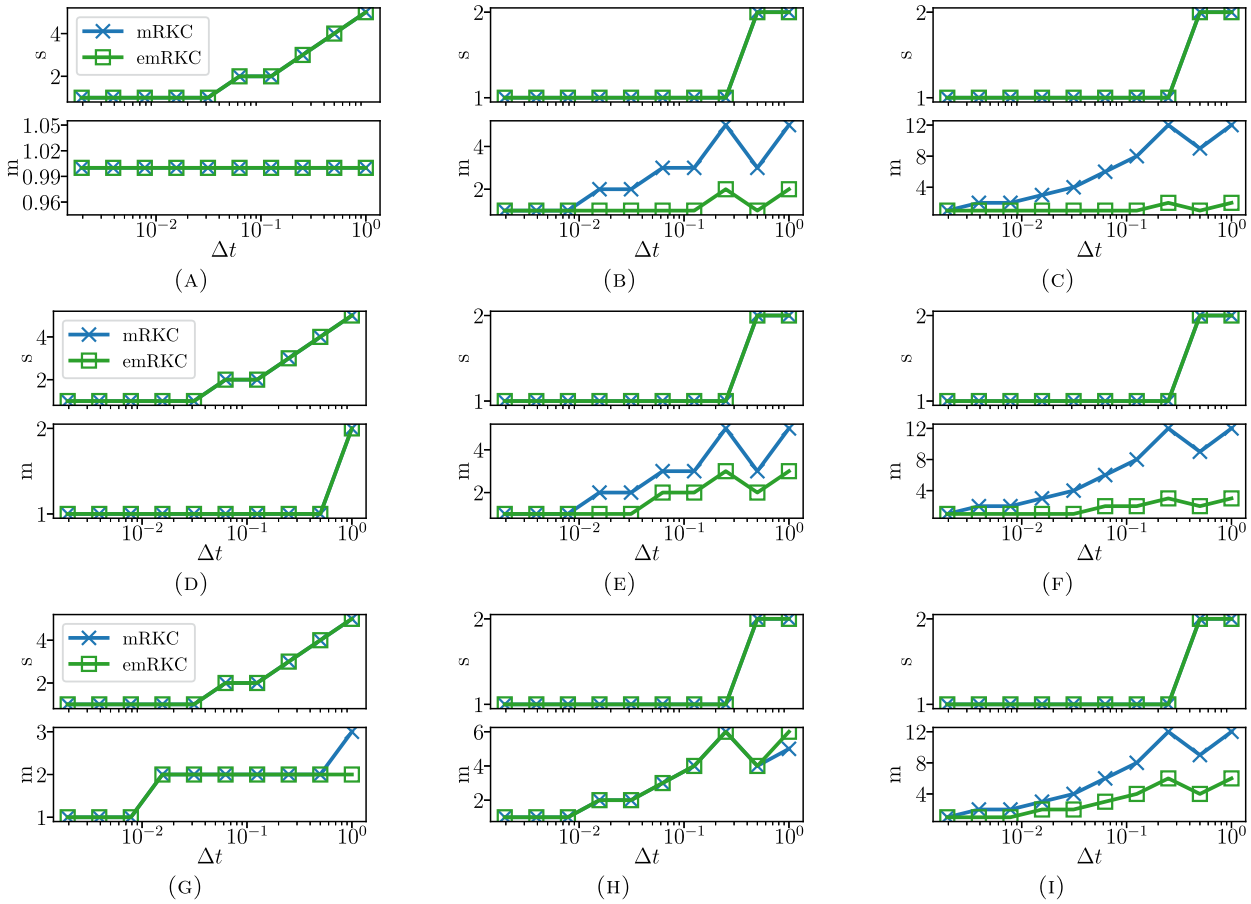


FIGURE 3. Two dimensional rectangle. Number of stages taken by the mRKC and emRKC methods, with different mesh sizes Δx and ionic models. Recall that s, m are the number of stages taken by the outer and inner RKC iterations, respectively. (a) HH model, $\Delta x = 0.2\,\text{mm}$. (b) CRN model, $\Delta x = 0.2\,\text{mm}$. (c) TTP model, $\Delta x = 0.2\,\text{mm}$. (d) HH model, $\Delta x = 0.1\,\text{mm}$. (e) CRN model, $\Delta x = 0.1\,\text{mm}$. (f) TTP model, $\Delta x = 0.1\,\text{mm}$. (g) HH model, $\Delta x = 0.05\,\text{mm}$. (h) CRN model, $\Delta x = 0.05\,\text{mm}$. (i) TTP model, $\Delta x = 0.05\,\text{mm}$.

previous cases, however, the mRKC method here is unstable for the TTP model, whereas the emRKC method always remains stable. The relative error for emRKC is always comparable if not better (HH case) to the error for IMEX-RL, which is the standard method in the cardiac community. Note that relative error of 1%–5% is comparable to the spatial error, which is of the same order for linear FE and mesh size of $\approx 0.25\,\text{mm}$, see [45].

In Figure 8, we observe that the number of stages used by the mRKC and emRKC methods is higher because of the smaller minimal mesh size. Therefore, the mRKC method evaluates the stiff part f_E^1 of the ionic model many times (18b), which may be the source of internal instabilities. As the internal stability analysis of mRKC is not available, this issue is not fully understood yet. Finally, we note that both explicit multirate methods always take a number of stages significantly larger than one, indicating that any standard explicit method would be highly unstable for such large step sizes. In Figure 9, the results from the efficiency experiment again suggest that emRKC is slightly faster than the IMEX-RL method, despite the high number of stages (Fig. 8) due to the mesh induced increased stiffness.

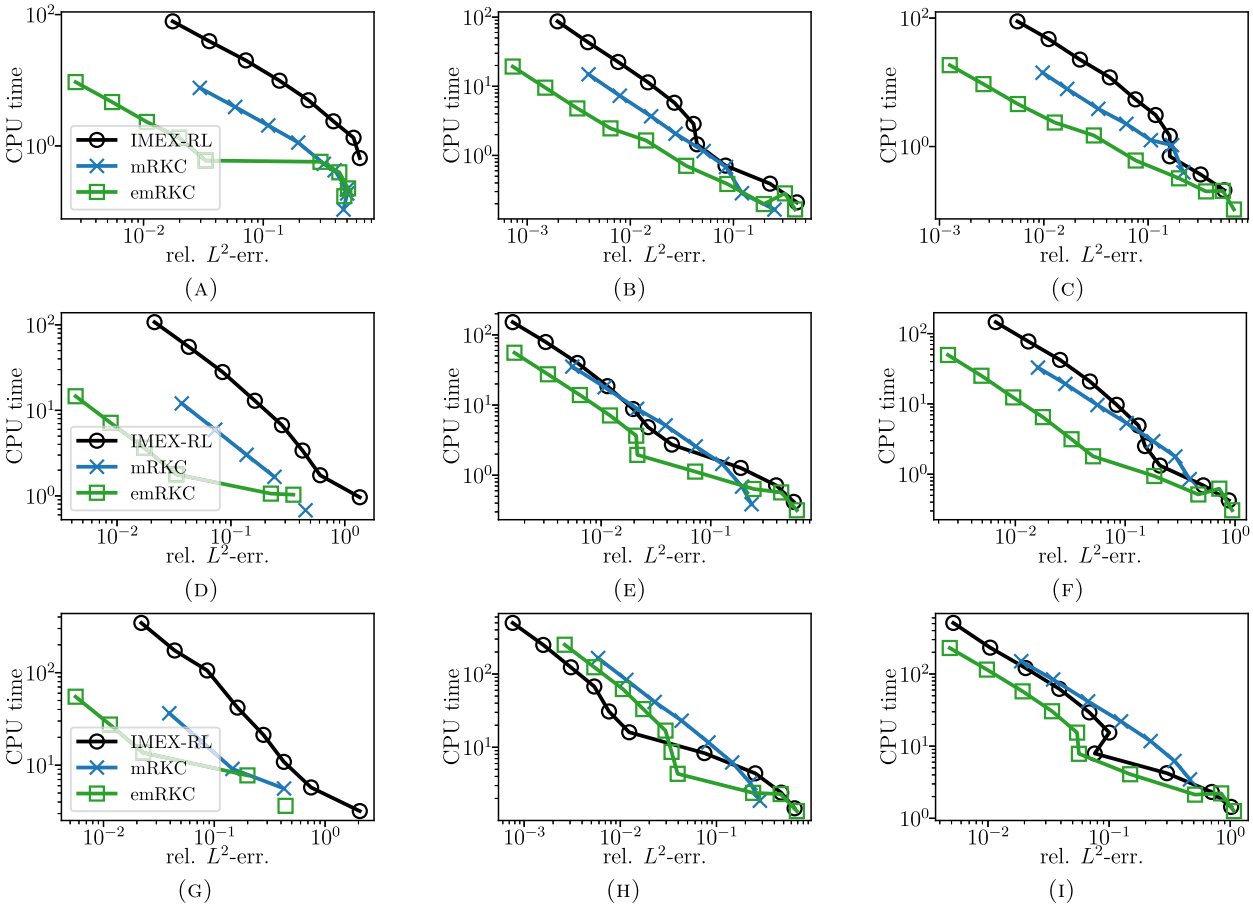


FIGURE 4. Two dimensional rectangle. Efficiency experiments with different mesh sizes Δx and ionic models. (a) HH model, $\Delta x = 0.2\,\text{mm}$. (b) CRN model, $\Delta x = 0.2\,\text{mm}$. (c) TTP model, $\Delta x = 0.2\,\text{mm}$. (d) HH model, $\Delta x = 0.1\,\text{mm}$. (e) CRN model, $\Delta x = 0.1\,\text{mm}$. (f) TTP model, $\Delta x = 0.1\,\text{mm}$. (g) HH model, $\Delta x = 0.05\,\text{mm}$. (h) CRN model, $\Delta x = 0.05\,\text{mm}$. (i) TTP model, $\Delta x = 0.05\,\text{mm}$.

Finally, we provide some code profiling results for the emRKC and IMEX-RL methods. First, we measure how much time the emRKC method spends in evaluating f_F, f_S, and in performing the exponential Euler step for f_E (see Algorithm 3). To do so, we apply emRKC with $\Delta t = 2^{-6}\,\text{ms} = 0.015625\,\text{ms}$ and profile the EMRKC_STEP function of Algorithm 3. We choose this Δt since it is the largest one yielding a relative error of about 1% (Fig. 7). Moreover, in view of the efficiency results depicted in Figure 9, we deduce that the outcome of the profiling is only weakly affected by the size of Δt. The results for the three ionic models are displayed in Figure 10, where for each graph we show the average over five runs. The four wedges in the pie chart are as follows: F represents the time spent in evaluating f_F, S the time spent evaluating f_S, E the time for computing the exponential step for f_E, and "other" the time required for all remaining operations, such as communications and vector additions. The size of each wedge is proportional to the time spent in the corresponding task, with the total time (spent in the EMRKC_STEP routine) at the center of the pie chart. At first sight, the time spent in F seems to depend on the ionic model, even though the number of matrix-vector multiplications only depends on the mesh. Still, a closer look at those timings reveals that this variability is due to large fluctuations in the

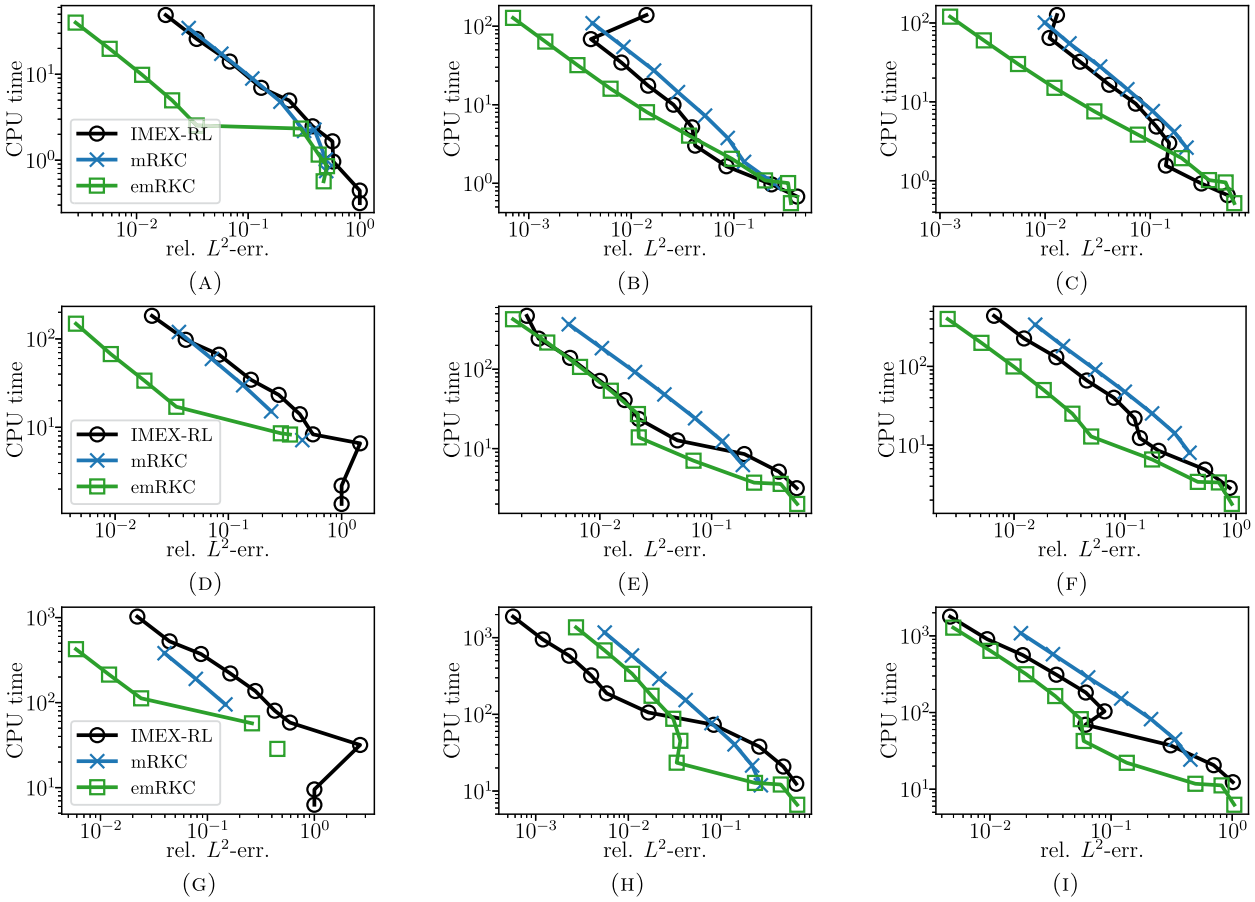


FIGURE 5. Three dimensional cuboid. Efficiency experiments with different mesh sizes Δx and ionic models. (a) HH model, $\Delta x = 0.2\,\text{mm}$. (b) CRN model, $\Delta x = 0.2\,\text{mm}$. (c) TTP model, $\Delta x = 0.2\,\text{mm}$. (d) HH model, $\Delta x = 0.1\,\text{mm}$. (e) CRN model, $\Delta x = 0.1\,\text{mm}$. (f) TTP model, $\Delta x = 0.1\,\text{mm}$. (g) HH model, $\Delta x = 0.05\,\text{mm}$. (h) CRN model, $\Delta x = 0.05\,\text{mm}$. (i) TTP model, $\Delta x = 0.05\,\text{mm}$.

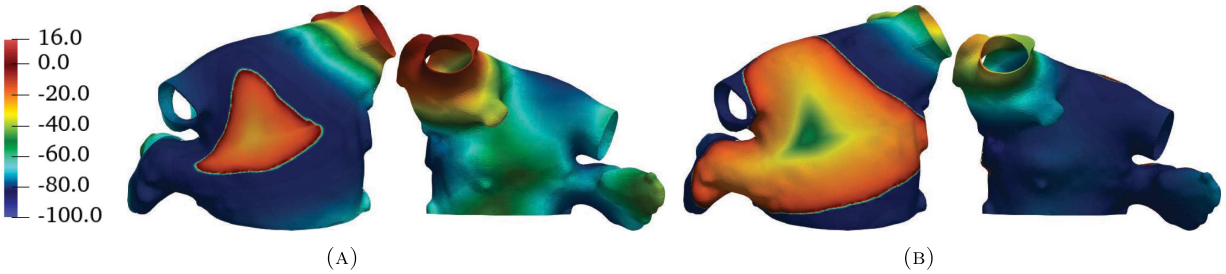


FIGURE 6. Left atrium geometry. Initial value and solution at $t = 50\,\text{ms}$ for the CRN ionic model. (a) Initial value. (b) Solution at $t = 50\,\text{ms}$.

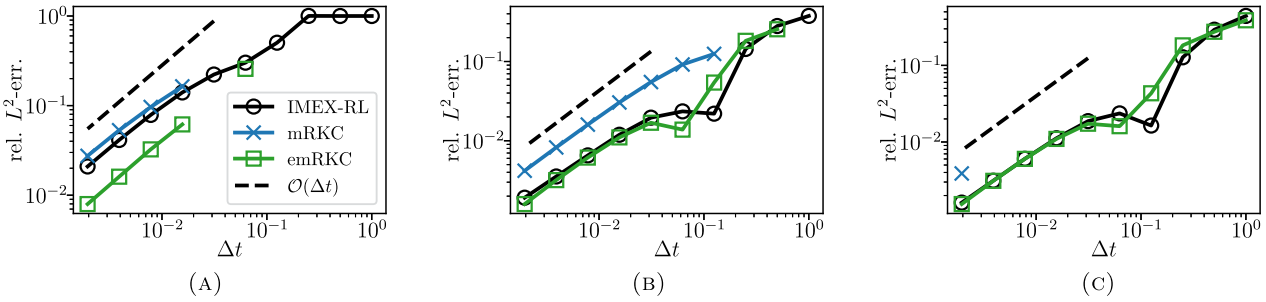


FIGURE 7. Left atrium geometry. Convergence experiments with different ionic models. (a) HH model. (b) CRN model. (c) TTP model.

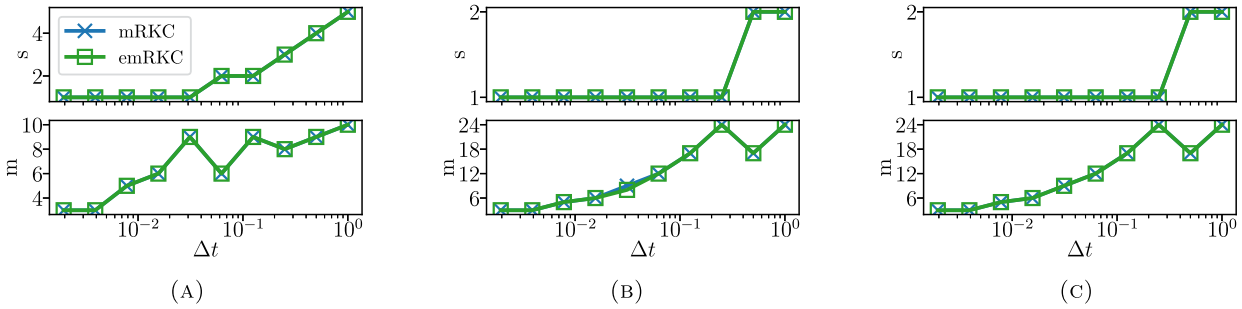


FIGURE 8. Left atrium geometry. Number of stages used by the mRKC and emRKC methods with different ionic models. Recall that s, m are the number of stages taken by the outer and inner RKC iterations, respectively. (a) HH model. (b) CRN model. (c) TTP model.

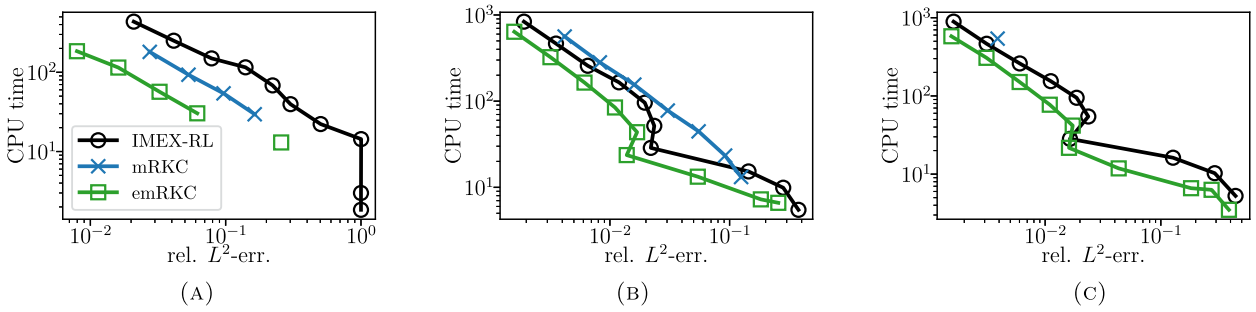


FIGURE 9. Left atrium geometry. Efficiency experiments with different ionic models. (a) HH model. (b) CRN model. (c) TTP model.

time spent computing matrix-vector multiplications. In contrast, the execution timings for S and E are truly ionic model dependent.

Next, we perform the same profiling experiments for the IMEX-RL code while measuring the time spent in IMEX-RL_STEP of Algorithm 4. The corresponding pie charts are displayed in Figure 11, where the total execution time is again distributed between the four wedges F, S, E and "other". Here, F represents the time

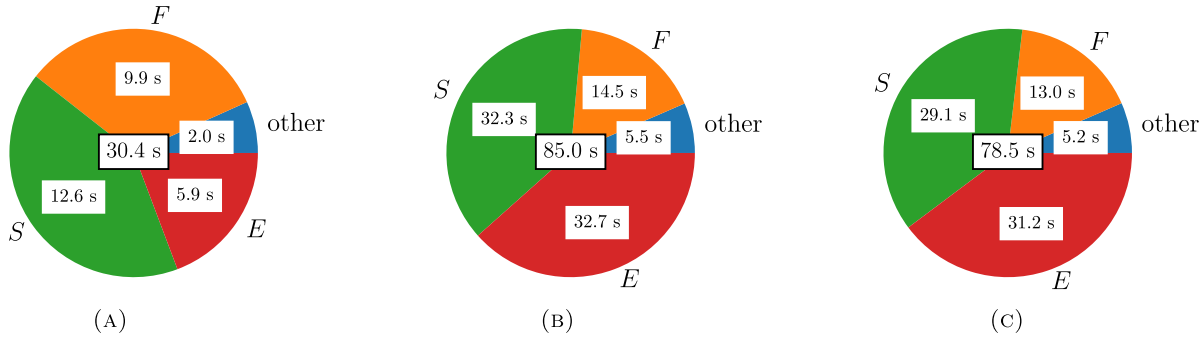


FIGURE 10. Left atrium geometry. Code profiling of the emRKC method. Wedges F, S, E indicate time spent evaluating f_F, f_S, and computing the exponential step, respectively. Remaining operations are represented by the "other" wedge. (a) HH model. (b) CRN model. (c) TTP model.

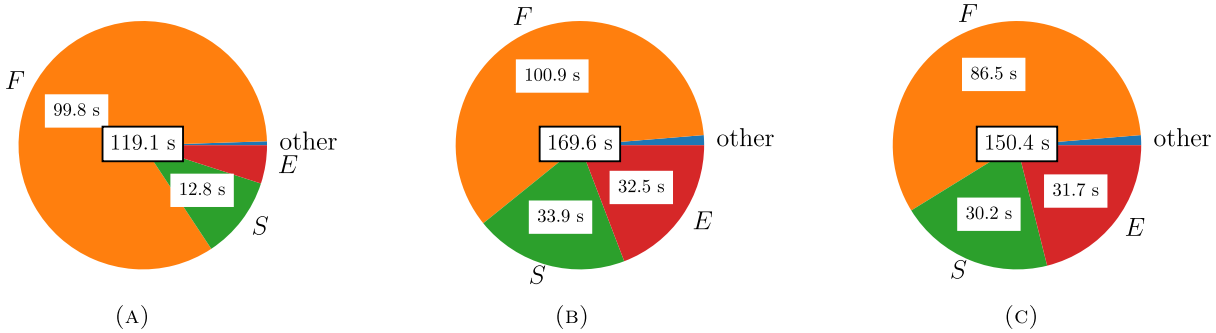


FIGURE 11. Left atrium geometry. Code profiling of the IMEX-RL method. Wedges F, S, E indicate time spent solving the linear system for f_F, evaluating f_S, and computing the exponential step, respectively. Remaining operations are represented by the "other" wedge. (a) HH model. (b) CRN model. (c) TTP model.

spent in solving the linear system in ALgorithm 4 Line 4, S the time spent in evaluating the slow components of the ionic model in Line 3 of Algorithm 4 , and E the time spent in evaluating the exponential for the gating variables in Line 2 of Algorithm 4.

As expected, both emRKC and IMEX-RL spend approximately the same amount of time in S and E, since the required operations are indeed identical. Due to the larger number of vector additions, the emRKC method spends more time in the "other" wedge. However, the total time required by IMEX-RL to solve the linear systems is about 6–7 times larger than the time spent by emRKC in evaluating f_F (compare the two F wedges). This significant reduction in execution time implies that the stabilization procedure of emRKC for the diffusion term f_F is much cheaper than solving a linear system during each implicit Euler step. Remarkably, we achieve this reduction in computational cost without sacrificing accuracy, as illustrated in Figure 7. Overall, emRKC achieves a speed-up factor of about 2 over IMEX-RL for the two most relevant ionic models CRN and TTP.

5.5. Left atrium with fibrosis

Finally, we repeat the numerical experiments from Section 5.4 using the same computational mesh but in the presence of fibrosis in the cardiac tissue – a common condition in patients with history of atrial fibrillation. Mathematically, fibrosis can be modelled by locally reducing the electric conductivity tensor and altering some

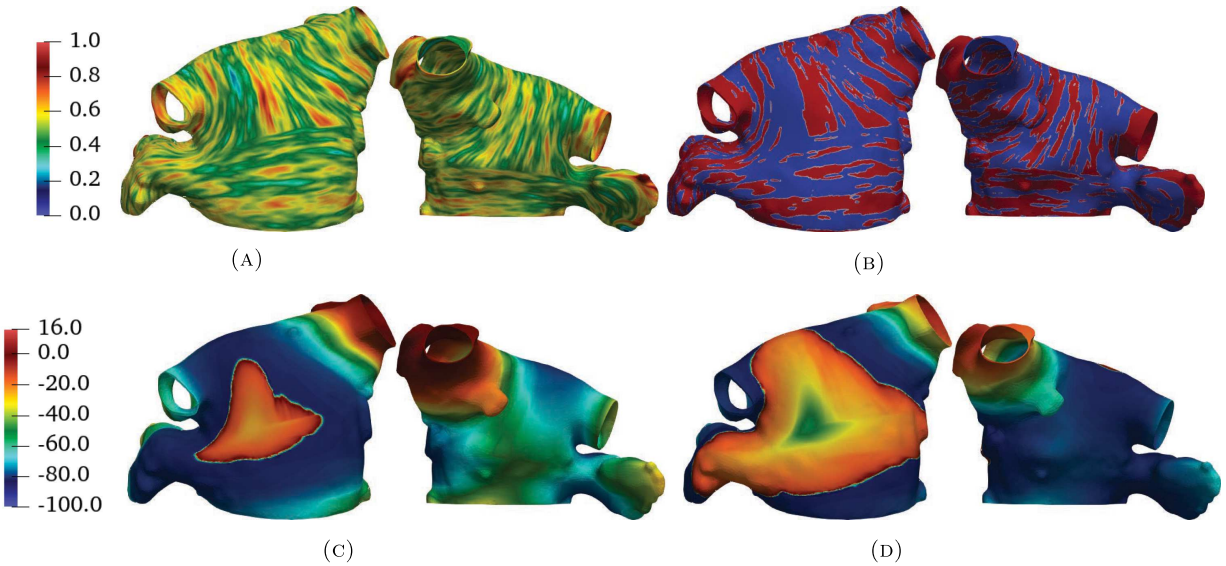


FIGURE 12. Left atrium with fibrosis. The fibrosis pattern (panel B), has been obtained by clipping at level 0.5 the random field (panel A). All elements within the red region in panel (B) are considered "fibrotic" and assigned a transverse conductivity of 0 (no transverse conduction). We also report the initial value (panel C) and solution at $t = 50$ ms (panel D) for the CRN ionic model. (a) Fibrosis random field $u(x)$. (b) Fibrosis pattern, in red where $\sigma_t = 0$. (c) Initial value. (d) Solution at $t = 50$ ms.

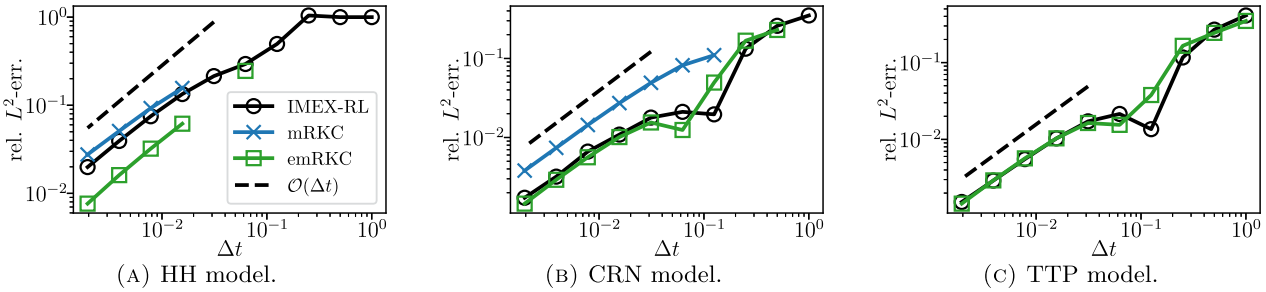
ionic currents in the membrane model. Here, we incorporate fibrosis only in the conductivity tensor by decoupling cells in the cross-fiber direction [23, 25, 26]. The spatial heterogeneity is obtained from a spatially-correlated random field $u : \Omega \to [0, 1]$, where we set $\sigma_t = 0$ if $u(\boldsymbol{x}) > q$, where q is some quantile value; in our simulations, we use the median. Following Section 2 of [46], we generate the random field u, shown in Figure 12a, by solving a stochastic differential equation. The corresponding fibrosis pattern is shown in Figure 12b.

The initial value is generated exactly as in Section 5.4 and is displayed in Figure 12c, whereas the solution at the final time $T = 50$ ms is shown in Figure 12d. Again in Figure 13, we verify first-order convergence for all three methods and models and also observe instabilities in mRKC but not in emRKC. The number of stages taken by the methods is omitted here since it essentially coincides with that in Figure 8. In Figure 14, we display the results from the efficiency experiments where we again observe that emRKC is slightly faster than IMEX-RL.

5.6. Parallel performance and scalability

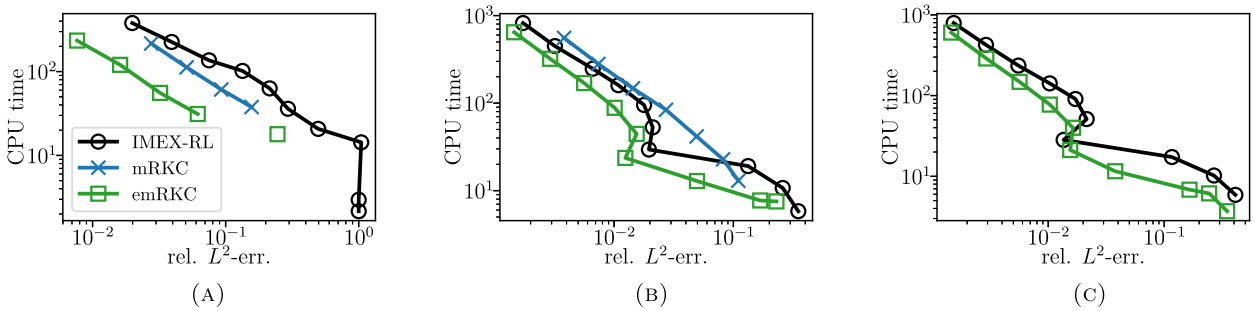
To illustrate the inherent parallelism of the explicit emRKC method, we now conduct a strong scalability experiment where for a fixed problem size we progressively increase the number of processors. Again we consider the left atrium geometry from Section 5.4 and use the CRN ionic model with 21 state variables (20 for the ionic model and 1 for the potential). Since the FE mesh has about 1.7×10^6 degrees of freedom (dof's), the resulting total problem size is about 36×10^6 dof's.

In Figure 15, we display for both the emRKC and IMEX-RL methods the total execution CPU time against the number of processors ranging from 16 to 1024; hence, we start with about 2.2×10^6 dof's per processor and end up with only about 35 000 dof's per processor. Although both methods yield an optimal speed-up up to 256 processors for this problem size, the parallel efficiency of the implicit IMEX-RL then declines. Although those preliminary computational results are limited to 1024 processors and a relatively modest problem size, they



(A) HH model. (B) CRN model. (C) TTP model.

FIGURE 13. Left atrium with fibrosis. Convergence experiments with different ionic models. (a) HH model. (b) CRN model. (c) TTP model.



(A) (B) (C)

FIGURE 14. Left atrium with fibrosis. Efficiency experiments with different ionic models. (a) HH model. (b) CRN model. (c) TTP model.

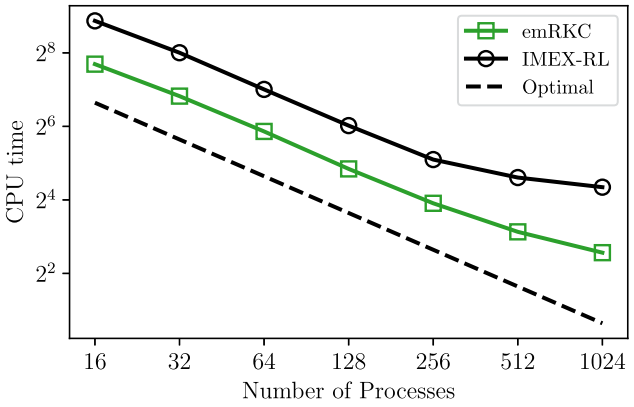


FIGURE 15. Strong scalability. CPU time *vs.* the number of processors.

already illustrate not only that the explicit emRKC method is faster than the IMEX-RL methods, as expected from Figures 10 and 11, but also that the explicit emRKC method exhibits better parallel scalability than the IMEX-RL method.

6. Conclusion

Explicit stabilized methods, such as RKC, strike a judicious balance between explicit and implicit methods in particular for the time integration of large-scale nonlinear parabolic PDE's. The monodomain equation from cardiac electrophysiology is in principle well-suited for explicit stabilized methods due to its moderately stiff diffusive part. However, in the presence of coupled stiff nonlinear ionic models, which are inherently multiscale in nature, conventional RKC methods can become ineffective. By adapting the number of stages locally to individual components depending on their respective degree of stiffness, mRKC methods [5] significantly mitigate this efficiency loss, yet without exploiting the particular diagonal structure of ionic models as in the standard Rush–Larsen approach.

To further improve upon the mRKC approach, we have proposed the emRKC method tailored specifically to the monodomain model. Drawing inspiration from the Rush–Larsen approach, the emRKC method leverages the special diagonal structure of ionic models for which it applies exponential integration. The combination of mRKC with exponential time integration results in a particularly efficient approach for solving the monodomain equations. Moreover, since it is fully explicit, the emRKC is particularly well-suited for implementation on GPU architectures in a parallel HPC environment. As existing fully explicit solvers for the monodomain model are always limited to structured grids [36], the emRKC thus removes this limitation and extends explicit time integration to unstructured grids, which are crucial for electro-mechanical [22, 48] and sub-cellular [35, 51, 52] models.

The stability and accuracy properties of the emRKC method have been analyzed in Sections 4.4 and 4.5. In particular, in Theorem 4.4 we show that for a linear scalar test equation the method is stable without constraint on the step size for sufficiently many stages. We also prove in Theorem 4.5 that the method is first-order accurate. Both stability and accuracy are also verified numerically. In particular, we show in Section 5.1 through a realistic nonlinear example that emRKC can take up to 1500 times larger time steps than a standard explicit method.

In our numerical experiments, emRKC outperformed or was on par with a standard baseline method, which combines IMEX integration for the parabolic component with exponential integration for the ionic model, as such a fair benchmark for comparison. All numerical experiments unequivocally showcased the good performance of emRKC, even in the presence of severe stiffness and nonlinearity.

The results of our study provide compelling evidence that explicit methods can indeed be exploited effectively for the solution of large-scale, stiff, nonlinear equations and can be a robust and efficient choice for simulating multiscale dynamics. Furthermore, the outcomes of this study provide a foundation for the design and implementation of explicit parallel-in-time methods specifically tailored for addressing the monodomain equation.

Appendix A. Nonlinear power iteration

Here we provide a pseudo-code for the nonlinear power iteration [37,38,57,58,64] used in the RKC, mRKC, and emRKC methods to estimate the spectral radii of the right-hand sides f, f_F, f_S. The function Compute_Rho below takes as input parameters the time t and state y, for which the spectral radius of $\frac{\partial f}{\partial y}(t,y)$ needs to be estimated, the nonlinear function f, and an initial guess v, ρ for the dominant eigenvector of $\frac{\partial f}{\partial y}(t,y)$ and its spectral radius. If no initial guess v, ρ is available, we simply choose an arbitrary random vector and set $\rho = 0$. The computed values v, ρ returned by Compute_Rho provide an initial guess for the next function call.

The function Compute_Rho indeed corresponds to a power iteration, as Line 8 of Algorithm 5 implies that

$$v \approx \tfrac{\partial f}{\partial y}(t,y)(z-y) = q\tfrac{\partial f}{\partial y}(t,y)v_{\text{old}}, \tag{A.1}$$

while Line 10

$$\rho = \frac{n_v}{\delta} = \frac{\|f(t,z)-f(t,y)\|}{\|z-y\|} \geq \frac{\langle f(t,z)-f(t,y), z-y\rangle}{\|z-y\|^2} \approx \frac{\langle \frac{\partial f}{\partial y}(t,y)q\cdot v_{\text{old}}, q\cdot v_{\text{old}}\rangle}{\|q\cdot v_{\text{old}}\|^2} = \frac{\langle \frac{\partial f}{\partial y}(t,y)v_{\text{old}}, v_{\text{old}}\rangle}{\|v_{\text{old}}\|^2} \tag{A.2}$$

Algorithm 5. Nonlinear power iteration.

function COMPUTE_RHO(t, y, f, v, ρ)
  $n_y = \|y\|$, $n_v = \|v\|$
  $\epsilon = 10^{-8}$, $\delta = n_y\epsilon$, $q = \delta/n_v$
  $tol = 10^{-2}$, $\rho_{\text{old}} = 0$
  while $|\rho - \rho_{\text{old}}| \geq tol \cdot \rho$ **do**
    $\rho_{\text{old}} = \rho$, $v_{\text{old}} = v$
    $z = y + q \cdot v_{\text{old}}$ ▷ $\|z - y\| = \delta$
    $v = f(t, z) - f(t, y)$ ▷ $v \approx \frac{\partial f}{\partial y}(t, y)(z - y) = q\frac{\partial f}{\partial y}(t, y)v_{old}$
    $n_v = \|v\|$, $q = \delta/n_v$
    $\rho = n_v/\delta$ ▷ $n_v/\delta = \|f(t, z) - f(t, y)\|/\|z - y\|$
  return v, ρ

approximately bounds the Rayleigh quotient. In practice, ρ is subsequently multiplied by a safety factor $\alpha > 1$, typically $\alpha = 1.05$.

The added computational cost from Algorithm 5 during time integration scheme is marginal. On the one hand, there is no need to update v, ρ at every time step, every so often, or even just once, is usually sufficient. On the other hand, subsequent calls typically converge very fast (1−2 iterations) thanks to the previous values of v, ρ now available as initial guesses.

Acknowledgements

The first author thanks the developers of FEniCSx and the Swiss National Supercomputing Centre staff for their support.

Funding

This work was supported by the European High-Performance Computing Joint Undertaking EuroHPC under grant agreements No. 955495 (MICROCARD) and No. 955701 (TIME-X), co-funded by the Horizon 2020 programme of the European Union (EU) and the Swiss State Secretariat for Education, Research and Innovation. We also acknowledge the CSCS-Swiss National Supercomputing Centre (project No. s1074).

References

[1] A. Abdulle, Fourth order Chebyshev methods with recurrence relation. *SIAM J. Sci. Comput.* **23** (2002) 2041–2054.
[2] A. Abdulle, Explicit stabilized Runge–Kutta methods, in Encyclopedia of Applied and Computational Mathematics. Springer (2015) 460–468.
[3] A. Abdulle and A.A. Medovikov, Second order Chebyshev methods based on orthogonal polynomials. *Numer. Math.* **18** (2001) 1–18.
[4] A. Abdulle and G. Vilmart, PIROCK: a swiss-knife partitioned implicit-explicit orthogonal Runge–Kutta Chebyshev integrator for stiff diffusion-advection-reaction problems with or without noise. *J. Comput. Phys.* **242** (2013) 869–888.
[5] A. Abdulle, M.J. Grote and G. Rosilho de Souza, Explicit stabilized multirate method for stiff differential equations. *Math. Comput.* **91** (2022) 2681–2714.
[6] P. Africa, M. Salvador, P. Gervasio, L. Dede' and A. Quarteroni, A matrix-free high-order solver for the numerical solution of cardiac electrophysiology. *J. Comput. Phys.* **478** (2023) 111984.
[7] P.C. Africa, R. Piersanti, M. Fedele, L. Dede' and A. Quarteroni, Lifex-fiber: an open tool for myofibers generation in cardiac computational models. *BMC Bioinf.* **24** (2023) 143.
[8] M.S. Alnaes, A. Logg, K.B. Ølgaard, M.E. Rognes and G.N. Wells, Unified form language: a domain-specific language for weak formulations of partial differential equations. *ACM Trans. Math. Softw.* **40** (2014) 1–37.
[9] S. Balay, S. Abhyankar, M.F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E.M. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, J. Faibussowitsch, W.D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M.G. Knepley, F. Kong, S. Kruger, D.A. May, L.C. McInnes, R.T. Mills, L. Mitchell, T. Munson, J.E. Roman, K. Rupp, P. Sanan, J. Sarich, B.F. Smith, S. Zampini, H. Zhang, H. Zhang and J. Zhang. PETSc Web (2023).

[10] R.H. Clayton, O. Bernus, E.M. Cherry, H. Dierckx, F.H. Fenton, L. Mirabella, A.V. Panfilov, F.B. Sachse, G. Seemann and H. Zhang, Models of cardiac tissue electrophysiology: progress, challenges and open questions. *Prog. Biophys. Mol. Biol.* **104** (2011) 22–48.

[11] M. Clerx, P. Collins, E. de Lange and P.G. Volders, Myokit: a simple interface to cardiac cellular electrophysiology. *Prog. Biophys. Mol. Biol.* **120** (2016) 100–114.

[12] G. Cohen, P. Joly, J.E. Roberts and N. Tordjman, Higher order triangular finite elements with mass lumping for the wave equation. *SIAM J. Numer. Anal.* **38** (2001) 2047–2078.

[13] P. Colli Franzone, L.F. Pavarino and S. Scacchi, Mathematical Cardiac Electrophysiology. Vol. 13. Springer (2014).

[14] Y. Coudière, C.D. Lontsi and C. Pierre, Rush-Larsen time-stepping methods of high order for stiff problems in cardiac electrophysiology. *Electron. Trans. Numer. Anal.* **52** (2020) 342–357.

[15] M. Courtemanche, R.J. Ramirez and S. Nattel, Ionic mechanisms underlying human atrial action potential properties: insights from a mathematical model. *Amer. J. Physiol. Heart Circulatory Physiol.* **275** (1998) H301–H321.

[16] M. Croci and G. Rosilho de Souza, Mixed-precision explicit stabilized Runge–Kutta methods for single- and multiscale differential equations. *J. Comput. Phys.* **464** (2022) 111349.

[17] L.D. Dalcin, R.R. Paz, P.A. Kler and A. Cosimo, Parallel distributed computing using Python. *Adv. Water Res.* **34** (2011) 1124–1139.

[18] T. Dumont, M. Duarte, S. Descombes, M.A. Dronne, M. Massot and V. Louvet, Simulation of human ischemic stroke in realistic 3D geometry. *Commun. Nonlinear Sci. Numer. Simul.* **18** (2013) 1539–1557.

[19] R.D. Falgout and U.M. Yang, Hypre: a library of high performance preconditioners, in International Conference on Computational Science. Springer (2002) 632–641.

[20] T.E. Fastl, C. Tobon-Gomez, A. Crozier, J. Whitaker, R. Rajani, K.P. McCarthy, D. Sanchez-Quintana, S.Y. Ho, M.D. O'Neill, G. Plank, M.J. Bishop and S.A. Niederer, Personalized computational modeling of left atrial geometry and transmural myofiber architecture. *Med. Image Anal.* **47** (2018) 180–190.

[21] T.E. Fastl, C. Tobon-Gomez, A. Crozier, J. Whitaker, R. Rajani, K.P. McCarthy, D. Sanchez-Quintana, S.Y. Ho, M.D. O'Neill, G. Plank, M.J. Bishop and S.A. Niederer, Personalized computational finite element models for left atrial electromechanics (2021).

[22] M. Favino, S. Pozzi, S. Pezzuto, F.W. Prinzen, A. Auricchio and R. Krause, Impact of mechanical deformation on pseudo-ECG: a simulation study. *EP Europace* **18** (2016) iv77–iv84.

[23] L. Gander, S. Pezzuto, A. Gharaviri, R. Krause, P. Perdikaris and F. Sahli Costabal, Fast characterization of inducible regions of atrial fibrillation models with multi-fidelity gaussian process classification. *Front. Physiol.* **13** (2022) 757159.

[24] S. Geevers, W.A. Mulder and J.J.W. van der Vegt, New higher-order mass-lumped tetrahedral elements for wave propagation modelling. *SIAM J. Sci. Comput.* **40** (2018) A2830–A2857.

[25] A. Gharaviri, E. Bidar, M. Potse, S. Zeemering, S. Verheule, S. Pezzuto, R. Krause, J.G. Maessen, A. Auricchio and U. Schotten, Epicardial fibrosis explains increased endo–epicardial dissociation and epicardial breakthroughs in human atrial fibrillation. *Front. Physiol.* **11** (2020) 68.

[26] A. Gharaviri, S. Pezzuto, M. Potse, S. Verheule, G. Conte, R. Krause, U. Schotten and A. Auricchio, Left atrial appendage electrical isolation reduces atrial fibrillation recurrences: simulation study. *Circulation: Arrhythmia Electrophysiol.* **14** (2021) e009230.

[27] M.J. Grote and J. Diaz, Energy conserving explicit local time stepping for second-order wave equations. *SIAM J. Sci. Comput.* **31** (2009) 1985–2014.

[28] M.J. Grote and T. Mitkova, Explicit local time-stepping methods for Maxwell's equations. *J. Comput. Appl. Math.* **234** (2010) 3283–3302.

[29] M.J. Grote and T. Mitkova, High-order explicit local time-stepping methods for damped wave equations. *J. Comput. Appl. Math.* **239** (2013) 270–289.

[30] M.J. Grote, M. Mehlin and T. Mitkova, Runge–Kutta-based explicit local time-stepping methods for wave propagation. *SIAM J. Sci. Comput.* **37** (2015) A747–A775.

[31] E. Hairer and G. Wanner, Solving Ordinary Differential Equations II. Vol. 14. Springer-Verlag, Berlin (2002).

[32] E. Hairer, S.P. Nørsett and G. Wanner, Solving Ordinary Differential Equations I, 2 edition. Vol. 8. Springer-Verlag, Berlin (2008).

[33] M. Hochbruck and A. Ostermann, Exponential integrators. *Acta Numer.* **19** (2010) 209–286.

[34] A.L. Hodgkin and A.F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **117** (1952) 500–544.

[35] N.M.M. Huynh, F. Chegini, L.F. Pavarino, M. Weiser and S. Scacchi, Convergence analysis of BDDC preconditioners for composite DG discretizations of the cardiac cell-by-cell model. *SIAM J. Sci. Comput.* **45** (2023) A2836–A2857.

[36] D. Krause, M. Potse, T. Dickopf, R. Krause, A. Auricchio and F. Prinzen, Hybrid parallelization of a large-scale heart model, in Facing the Multicore-Challenge II. Springer (2012) 120–132.

[37] B. Lindberg, IMPEX: a program package for solution of systems of stiff differential equations. Technical report, Department of Information Processing, Royal Institute of Technology, Stockholm (1972).

[38] B. Lindberg, IMPEX2: a procedure for solution of systems of stiff differential equations. Technical Report TRITA-NA-7303, Department of Information Processing, Royal Institute of Technology, Stockholm, Sweden (1973).

[39] L.P. Lindner, T. Gerach, T. Jahnke, A. Loewe, D. Weiss and C. Wieners, Efficient time splitting schemes for the monodomain equation in cardiac electrophysiology. *Int. J. Numer. Methods Biomed. Eng.* **39** (2023) e3666.

[40] C.M. Lloyd, J.R. Lawson, P.J. Hunter and P.F. Nielsen, The CellML model repository. *Bioinf. (Oxford, England)* **24** (2008) 2122–2123.

[41] C.D. Meyer, D.S. Balsara and T.D. Aslam, A stabilized Runge–Kutta–Legendre method for explicit super-time-stepping of parabolic and mixed equations. *J. Comput. Phys.* **257** (2014) 594–626.

[42] S.A. Niederer, E. Kerfoot, A.P. Benson, M.O. Bernabeu, O. Bernus, C. Bradley, E.M. Cherry, R. Clayton, F.H. Fenton, A. Garny, E. Heidenreich, S. Land, M. Maleckar, P. Pathmanathan, G. Plank, J.F. Rodríguez, I. Roy, F.B. Sachse, G. Seemann, O. Skavhaug and N.P. Smith, Verification of cardiac tissue electrophysiology simulators using an N-version benchmark. *Philos. Trans. R. Soci. A: Math. Phys. Eng. Sci.* **369** (2011) 4331–4351.

[43] D. Ogiermann, D. Balzani and L.E. Perotti, An explicit local space-time adaptive framework for monodomain models in cardiac electrophysiology. *Comput. Methods Appl. Mech. Eng.* **422** (2024) 116806.

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

[45] S. Pezzuto, J. Hake and J. Sundnes, Space-discretization error analysis and stabilization schemes for conduction velocity in cardiac electrophysiology. *Int. J. Numer. Methods Biomed. Eng.* **32** (2016) e02762.

[46] S. Pezzuto, A. Quaglino and M. Potse, On sampling spatially-correlated random fields for complex geometries, in Functional Imaging and Modeling of the Heart, edioted by Y. Coudière, V. Ozenne, E. Vigmond and N. Zemzemi. Vol. 11504. Springer International Publishing, Cham (2019) 103–111.

[47] R. Piersanti, P.C. Africa, M. Fedele, C. Vergara, L. Dedè, A.F. Corno and A. Quarteroni, Modeling cardiac muscle fibers in ventricular and atrial electrophysiology simulations. *Comput. Methods Appl. Mech. Eng.* **373** (2021) 113468.

[48] A. Quarteroni, L. Dedè and F. Regazzoni, Modeling the cardiac electromechanical function: a mathematical journey. *Bull. Am. Math. Soc.* **59** (2022) 371–403.

[49] G. Rosilho De Souza, *Numerical methods for deterministic and stochastic differential equations with multiple scales and high contrasts*. Ph.D. Thesis, EPFL (2020).

[50] G. Rosilho De Souza, mRKC: a multirate Runge–Kutta–Chebyshev code (2022).

[51] G. Rosilho de Souza, S. Pezzuto and R. Krause, Effect of gap junction distribution, size and shape on the conduction velocity in a cell-by-cell model for electrophysiology, in Functional Imaging and Modeling of the Heart. Vol. 13958 of *Lecture Notes in Computer Science*, edited by O. Bernard, P. Clarysse, N. Duchateau, J. Ohayon and M. Viallon. Springer, Cham (2023) 117–126.

[52] G. Rosilho de Souza, R. Krause and S. Pezzuto, Boundary integral formulation of the cell-by-cell model of cardiac electrophysiology. *Eng. Anal. Boundary Elem.* **158** (2024) 239–251.

[53] S. Rush and H. Larsen, A practical algorithm for solving dynamic membrane equations. *IEEE Trans. Biomed. Eng.* **BME-25** (1978) 389–392.

[54] F. Sahli Costabal, T. Banduc, L. Gander and S. Pezzuto, The fibrotic kernel signature: simulation-free prediction of atrial fibrillation, in Functional Imaging and Modeling of the Heart. Vol. 13958 of *Lecture Notes in Computer Science*, edited by O. Bernard, P. Clarysse, N. Duchateau, J. Ohayon and M. Viallon. Springer, Cham (2023) 87–96.

[55] M.W. Scroggs, I.A. Baratta, C.N. Richardson and G.N. Wells, Basix: a runtime finite element basis evaluation library. *J. Open Source Softw.* **7** (2022) 3982.

[56] M.W. Scroggs, J.S. Dokken, C.N. Richardson and G.N. Wells, Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. *ACM Trans. Math. Softw.* **48** (2022) 18:1–18:23.

[57] L.F. Shampine, Lipschitz constants and robust ode codes', Computational Methods in Nonlinear Mechanics, in Proceedings of the TICOM Second International Conference, edited by J.T. Oden. North-Holland Publishing Company, New York (1980) 427–449.

[58] L.F. Shampine, Diagnosing stiffness for Runge–Kutta methods. *SIAM J. Sci. Stat. Comput.* **12** (1991) 260–272.

[59] B.P. Sommeijer, L. Shampine and J.G. Verwer, RKC: an explicit solver for parabolic PDEs. *J. Comput. Appl. Math.* **88** (1998) 315–326.

[60] R.J. Spiteri and R.C. Dean, Stiffness analysis of cardiac electrophysiological models. *Ann. Biomed. Eng.* **38** (2010) 3592–3604.

[61] R.J. Spiteri and R.C. Dean, Erratum: stiffness analysis of cardiac electrophysiological models. *Ann. Biomed. Eng.* **40** (2012) 1622–1625.

[62] K.H.W.J. Ten Tusscher and A.V. Panfilov, Alternans and spiral breakup in a human ventricular tissue model. *Am. J. Physiol. Heart Circulatory Physiol.* **291** (2006) H1088–1100.

[63] P.J. Van der Houwen and B.P. Sommeijer, On the internal stability of explicit, m-stage Runge–Kutta methods for large m-values. *Z. Angew. Math. Mech.* **60** (1980) 479–485.

[64] J.G. Verwer, An implementation of a class of stabilized explicit methods for the time integration of parabolic equations. *ACM Trans. Math. Softw. (TOMS)* **6** (1980) 188–205.

[65] J.G. Verwer, Explicit Runge–Kutta methods for parabolic partial differential equations. *Appl. Numer. Math.* **22** (1996) 359–379.

[66] J.G. Verwer, W. Hundsdorfer and B.P. Sommeijer, Convergence properties of the Runge–Kutta–Chebyshev method. *Numer. Math.* **57** (1990) 157–178.